Subject Date No.

Table of Content

* المحاضرة رقم (1) — Introduction + complixity Analysis

* المحاضرة رقم (2) — Examples on complixity Analysis

* المحاضرة رقم (3) — Examples on complixity Analysis

* المحاضرة رقم (4) — Introduction to Sorting Algorithms

* المحاضرة رقم (5) — selection sort + Insertion Sort

* المحاضرة رقم (6) — Bubble sort

* المحاضرة رقم (7) — Merge sort

* المحاضرة رقم (8) — Quick sort

Subject Date No.

* المحاضرة رقم (9) Binary search Tree Part one

* المحاضرة رقم (10) Binary search Tree Part Two

* المحاضرة رقم (11) Operation on BST

* المحاضرة رقم (12) Greedy Algorithms

* المحاضرة رقم (13) Heap Sort

~~* المحاضرة~~

* المحاضرة رقم (14) Huffman Code

* المحاضرة رقم (15) Introduction to Graphs

* المحاضرة رقم (16) Special type of Graphs

* المحاضرة رقم (17) Graph Implementation + Graph Traversal + shortest Path Algo

NOTEBOOK

Subject Date No.

* المحاضرة رقم (18) Dijkstras Algorithm + MST

* المحاضرة رقم (19) Topological sort + Linear Time sorting

NOTEBOOK

Subject Algorithms *(1) المحاضرة الأولى Date Matmatical No. X

→ what is the meaning of Algorithms? → Methods X
→ computational X

↳ خطوات متتالية لحل مشكلة .

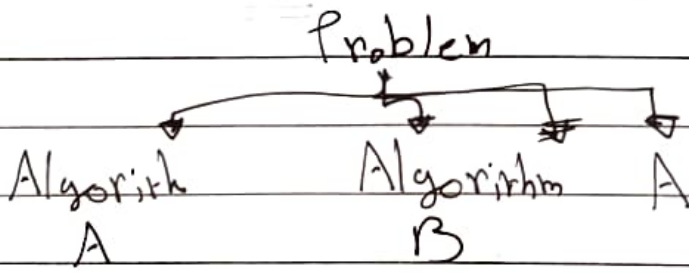
* وأكثر ما يميز دراستنا لـ Algorithms هو دراستنا بشكل كامل

(Abstract) مجرد ~~Astract~~

* وبالتالي لن يتم التطرق إلى تحليل الخوارزمية من وجهة نظر

ما يلي ← نوع الـ Data structure
كفاءة الـ pc المستخدم
نوع لغة البرمجة المستخدمة .

* غالبًا مثلاً أكثر من ~~خوارزمية~~ خوارزمية لحل مشكلة معينة



Subject Date No.

→ But, which of Them is a solution OR Best solution OR More effecient solution?

↳ ① All of Them is a solution.
②
③

→ what is the differance between Best And Effeicent solution?

* The Most effecient Algorithm is the one That have the least COST

↳ what is the Cost?

The cost is Time

↳ How we measure The Time?
we measure The Time in Constant unit

Subject Date No.

→ We said that we will study the Algorithms in Abstract Manner

So, the measurment of time in any other unit is it's dependent on other factors → The speed of the Machine
→ The Programming Language That is used.
... etc.

↳ Back to determining The Most efficient Algorithm

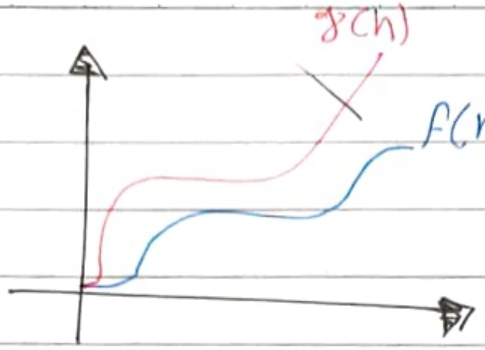
→ How to decide which Algorith Have The Least cost?
By comparing it with other situation or by other words with other CASE

↓ And That case is The (Worst case)

NOTEBOOK

Subject Date No.

Ex:-



Represents The running of The Algorithm

* The g(n) running in order of N^2

* The f(n) in The Worst case will run in order of N^2

* The Notations of the Wors case

① Big-O notation: The Algorithm That is running in big-O will guarantee ( ) that in The worst case will not exceed ( ) The order

Ex:- $O(n^2)$ ⇒ The Algorithm Won't Exceed The n^2 operation or step in The worst case

NOTEBOOK

Subject Date No.

② The little-o notation:

The ~~Algorithm~~ Algorithm will guarantee in the worst case ~~will~~ will run in order ~~less~~ little less than the worst case.

* How to work in the Exam?

* صلي كما النبي ←

① إذا طلب منك كتابة خوارزمية ولم تحدد طريقة الكتابة اكتب بأي طريقة

بأي ← انجليزي
انجليزي + عربي
عربي
Pseudocode

② ركز إذا كان المطلوب هو حل أو أفضل حل للخوارزمية

solution
Best solution
efficient solution

بينهم فرق

- - - انتظروا حلقة الزيادة :)

NOTEBOOK

Subject: Algorithms (2) Date: No.

How to compute the order (The Big-Notation) of An Algorithm?

→ Consider The following Examples:

① `cin >> X;` C_1 → C is an consitant Time (ثابت)

`cout << X;` C_2

→ $C_1 + C_2 = 2C \Rightarrow O(1)$

But why "1"?

We denote The consitant by "1"

Bec, There Two statment Each of Them Takes constitant Time to be excuted And Because We dont Know How the value of The constant The Sum of The costants equal To constant

② `cin >> A >> B >> C` C_1

`sum = A + B + C` C_2

`Avg = sum / 3` C_3

→ $C_1 + C_2 + C_3 = 3C = 3 \Rightarrow O(1)$

NOTEBOOK

Subject Date No.

* How to work in The Exam:-

* Note: ① في الامتحان إذا سألك أي من الخوارزميات التالية أسرع وكان كل خوارزمية تحل مشكلة مختلفة يكون الجواب أنك لا يوجد مقارنة

مثال: المثالين (1) + (2)

(1) يقرأ رقم واحد وطباعة

(2) يحسب ويطبع خانات الأرقام.

② إذا سألك عن هذه الخوارزمية ما هي وظيفتها أو ماذا تعمل؟

← التزم بالوجود يعني في مثال (1) X ليست رقم وهكذا تكلم ما هي.

↳ الخوارزمية (1) تقرأ رقم وتطبعه X خطأ.

تقرأ X أو متغير أو شيء لا مع

③

↳ Back To Examples:-

③ if (a > b)

    St; → Statment

else

    St;

→ O(1)

NOTEBOOK

Subject Date No.

(4) for (i 0 → n)
    st ;
→ O(n)

(5) if (a > b)
    ~~for (i 0 → n)~~
    for(i 0 → n)
else
    st;
→ O(n)

*How to work in The Exam:-

(6) for (i 0 → n)
    cin >> a;
    cout << a;
    n = n/2;
→ log(n) OR O(Log(n))

↳ * what is the (log)? العملية الحسابية التي تقلل من حجم الـ N بصورة كبيرة وغالبًا تكون خسرة.

n = n/4
n = n/8] == log(n)

Subject Date No.

* Note: ← المثال رقم (6) يكتب في الامتحان يكون الجواب في نموذج هو $O(n)$ وفي نموذج آخر هو $\log(n)$

↳ السبب ؟؟ هو صيغة السؤال

إذا كانت صيغة السؤال "As an C++ code" أي انها حددت لغة C++ يكون الجواب ← $O(n)$

إذا كان السؤال "As a segment of code" الجواب $\log(n)$ أي لغة حتى لو كانت كانت C++ ؟

If we rewrite the prev Example as C++ code

↳ for (i 0 → n)
    cin >> c;
    cout << c;
    n = n/2;

→ The code will Ends Here, so the complixity is $O(n)$

(7) for (i 0 → n) // O(n)
    St;
    for (i 0 → m) O(m)
    St;
→ n + m ⇒ O(n+m)

(8) if (a > b)
    for (i 0 → n) // O(n)
    else
    for (i 0 → m) // O(m)

→ O(Max(n,m))
↳ choose The Maxmium Number between Them.
But why?
Just one of the Two Loops will be excuted
so, The Max one is The worst case if it is excuted.

(9) for (i 0 → n) // O(n)
    St;
    for (i 0 → n) O(n)
    St;
→ n + n = 2n ⇒ O(n)

Subject Date No.

*Notes:

[1] We compute The exact order of an Algorithm Based on the Number of statment

[2] عند الأشخاص رح تكون النماذج مختلفة عادًا وبالتالي

[3] ويكتب عطيل خوارزمية A مثلاً ويكون المطلوب كالتالي

① ما هي وظيفة A؟

② ما حسب الـ Big-O لـ A؟ ولماذا؟

③ اكتب خوارزمية أفضل من A؟

Subject Algorithms (3) المحاضرة ✱ Date No.

↳ Some Examples on Finding The big-O Notation :-

(1) ~~for(~~

```
if a>b
    for (i 0 → n)
        sum (even)
else
    for (d 0 → m)
        sum (odd)
```

⟹ The Big-O is :- O(Max(n,m))

⟹ What does The Prevous Code do?

↳ Note :- في الامتحان إذا طلب منك ما هي وظيفة الخوارزمية أو شرح خطوات الكود إذا شرحتي بكون خطأ

إيجاد مجموع الأعداد الزوجية إذا كانت الرقم الأول (a) أكبر من (b) أو إيجاد مجموع الأعداد الفردية عدا ذلك.

↳ الشرح يكون كالتالي :-

NOTEBOOK

Subject Date No.

② while (i ≠ 4)
{
St;
}

↳ Big-O :- O(n) / Not O(1) ✗
↳ why?
* How many Times The statment inside the loop will be excuted?
~~we~~ we don't know because it's based on the value of (i) and we don't know the value of i
* So, ~~Again~~ what is the meaning of (n)?
(n) times
↳ How much (n)?
Maybe infinite.

Subject Date No.

③
```
if( a >b)
    for( i 0 → n)
else
    for (i 0 → m)
for (i 0 → k)
```
Max(n,m) + O(k)

↳ Big-O :: O(Max(n,m)+k)

④
```
for ( ; ;)
    St;
```
↳ Big-O :: O(n) ⇒ why? As same as the Answer in *example (2)

⑤
```
for (i 0 → n-m)        // O(n-m)
    St;
for (i 0 → k)          // O(1)
    St;
```
↳ Big-O :: O((n-m)+k)

Subject Date No.

⑥ Let The value of m = 10000, by considering The following Algorithm:

```
for (i 0 → n-m) // ليه O(n) ؟؟ why Not O(n-m)؟؟
    St;
for (i 0 → k) // O(k)
    St;
```

Because, The (m) becomes constant.

↳ Big-O :- O(n+k)

How To ~~work~~ work in The Exam?

① انتبه للفرق بين وظيفة ~~الـ~~ الحواديه وشرح الكود:

② إذا ما حدد نوع الكود (C++ أو Psedo code) إذا ما حدد نوع الكود ~~segment~~ segment of code

③ ~~ميه~~ مين Identation ال

```
for (i 0 → n)        for (i 0 → n)
    St;      ≠           St;
```

Subject Date No.

* Quiz:
(1) Write An Algorithm ~~That~~ To find The Max number in Array of size N?

(2) find The Big-O Notation?

(3) Do you Thinks That There is a better Algorithm than The one you wrote?
هل تعتقد أنه يمكن كتابة خوارزمية أفضل من التي كتبتها؟

Answer:-

(1)
```
arr[N]
Max = arr[0]
for (i = 1 → n)
    if ( arr[i] > Max
        Max = arr[i]
```

(2) Big-O = O(n)

(3) No, There isn't ⇒ why? لأنه أي خوارزمية يجب أن تجد العنصر الأكبر لابد لازم أن تنظر كل العناصر وبالتالي O(n)

Subject Algorithms (4) المحاضرة # Date No.

* Sorting Algorithms:

① simple sort
- insertion sort
- selection sort

The Big-O is: $O(N^2)$

- simple As Implementation (سهلة)
  - could be written in Any Programming Language

② Efficient sort
- Merge Sort
- Quicks sort
- Heap sort

In general The Big-O $O(n \log(n))$

- Efficient in The Manner of The Big-O

③ Bubble sort
- Bubble sort
- comb sort
- shell sort

شي داخل ولكنو ح يبر كليم

لا نصحيلو وحدة شرح وحدة فيو الاستحام

$O(N^2)$

④ Distribution sort
- counting sort
- Radix sort
- Bucked sort

$O(N)$

((Linear Sorting Algorithms))

لا يوجد فيها مقارنة

Subject Date No.

Notes:

* The categorization (تصنيف) for the sorting Algorithms differentiate (تختلف) From one To Another
Based on The Point of View for The one Who Puts The category

↳ يعني كلو واحد صنف الخوارزميات ال sort بناءً على وجهة نظر أو بناءً على معيار خاص فيه

* what is sorting?
Reordering items (ترتيب)
* ordering what?
For Example Numbers.
* The sorting or ordering Numbers Based on What?
Based on Comparison (مقارنة)
↳ أصغر شيء + أصغر كمية

↳ So, computing The Big-O for The sorting Algorithms is The number of Comparisons
* But. Isn't The Big-O for computing The Number of Statments?
True, But Here The Statment is The Comparison.

Subject Date No.

⇒ A Part of The comparison's May be discarded Based on The Problem

* There Are Two Types of sorting Algorithms Based on The Location of Excuting

- Internal Sorting :- The Algo And The Data Are in The Main Memory
- External Sorting :- A Part of The Algorithm And A Part of The Data Are in the memory And other Parts in The Virtual Memory (VM)

We will Study Just The Internal sorting Algorithms.

* The Distribution Sorting :-

↳ The Distribution Sorting is Not Based on The Number of comparisons Is based on The * ~~Number of~~ Number of (Digits * Number of items)

(عدد الأرقام * عدد العناصر)

Subject Date No.

Ex: comparing Two string

① Ali5 compare Ali7

A > < == A

l > < == l

i > < == l

5 > < == 7

② 123 compare 127

1 < > == 1

2 < > == 2

3 < > == 7

* The comparison compares The whole Item

→ But Distribution compares By Digits

↳ So, The Big-O is Basd on

d.n → O(n)

d: Number of Digits

n: Number of items

(d) is constant

Subject Date No.

* How To work's in The Exam? + Notes ⇒

↳ Question :- what is Algorithm is more efficient or best Radix sort with O(n) OR insertion sort with O(N²) ??

→ Answer :- I can Not compare between Them, because ~~Ther~~ They Are working in different way

insertion و Radix بين الطريقة ~~بين~~ ؟

~~وكيف كل واحد منهما يعمل بشكل مختلف~~

Distribution ← Radix

comparison ← insertion

طريقة كويسة في الفهم "حارة" ← simple + Efficient + Bubble ← يعني يمكن اختار الـ

~~طريقة كويسة ومختلفة~~ ← Distribution ← ~~واختار الـ~~

Subject Date No.

* The order of ~~out~~ our study of Algorithms:-

~~1~~ ~~insertion~~
~~selection~~

1) insertion
2) selection
3) Bubble
4) Merge
5) Quick

$O(N^2)$

$O(n \cdot \log(n))$

↳ The first Three Algorithms Are The Fundemental Algorithms for ~~the~~ faster sorting.

↳ * تعرضنا الخوارزميات الأساسية منه حيث سرعة الترتيب يعني إذا بدي ارتب Data أو ما أعرف هو أول ثلاثة خوارزميات

→ But, ~~There complixity~~ They Are $O(N^2)$? How To use Them First And For faster ~~Algorithm~~ Algorithm?

For Two Main Resones
- Problem size
- They Don't Need An extra data structure
زيادة

NOTEBOOK

Subject Date No.

Explain:

(1) Problem size: if The Problem size is Not large And can be Handled ( ) So, There is no need to run The other Algorithm with $O(n \log_2(n))$

(2) Extra Data Structure:
The Merge and Quick sort Needs More Data structure (DSA) To Run, And The Extra DSA Requires More Time for Handling them, So if There is No Need To use Quick and Merge sort, e.g.: Huge Problem size Is Not efficient To use Them.

NOTEBOOK

Subject Date No.

* Q:. Write (design) an Algorithm to sort n-numbers? (3 Points)

← Grading scale الأفضلية: التصحيح رح يعتمد على مدى
يعني كل ما كانت في إجابات أفضل للإجابات الأقل أفضلية
سوف تخسر جزء من العلامة - أو العلامة كلها :(-
حتى لو كانت صح

← إذا كانت الإجابات:
(1) ---- insertion
(1) --- Silection
(3) --- Merge
علامات

← أو إذا كانت:
1 ___ insertion
1 ___ selection
2 ___ Merge
3 ___ Quick

← مرحلة لزيادة العلامة هو اعطاء تبرير للإجابة
طيب بتمنى اعرف انه مناسب اعطي تبرير؟ أسأل حالي السؤال
بتمنى أتذكر كل تفاصيل الخوارزمية

← إذا كانت الإجابات كالتالي:
~~إذا أحد الطلاب كانت إجابته~~:
(3) ___ insertion, Because n is small and if the data is sorted will become O(n)
(1) ___ selection, Because n is small.

Subject Date No.

(3) ~~Heap~~ Merge sort, Because n is large.

(0) Merge sort .

* اختار أفضل خوارزمية بالنسبة للحالة التي لديها وليس عشوائيا ← Efficient sort

*Q:- write An efficient Algorithm for sorting n-numbers?

*A:- Quick or Merge sort X

↳ why wrong?

if you chose The Merge or Quick You Have To Justifies (يبرر) your Answer by saying "N is too large"

And if N is too large

Then it will need more DSA Based on Merge or Quick sort

And The More DSA you will use The More Time you need

Then It is Not Efficient To use Them.

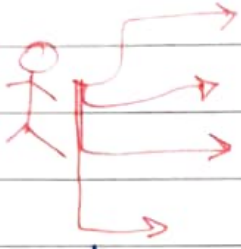
* Answer:

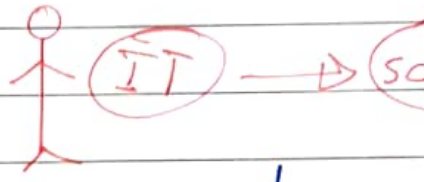
insertion OR Selection ← الحالة دي تكون عندك أنواع N صغيرة والتي لا أحتاج DSA إضافية وبالتالي هي Efficient بالنسبة للحالة.

NOTEBOOK

← الـ Bubble_Sort رح يكون منها نسختين وبالتالي
بدك تذكر أي نسخة ومع تبرير .

← ممكن تعبير بسيط عن السؤال وغير الجواب :





مثال :

اعط خوارزمية للذهاب من الـ IT الى الـ Science بأقصر وقت ؟
الإجابة : short spanning Tree

اعط خوارزمية للذهاب إلى أماكن محددة بأقل cost ؟
Minimum Spaninng Tree

---

* Note :- The DSA Requires More Time for Handling it
Ex :- OS File Mangment

← نقطة : إذا كانت الخوارزمية تعتمد على DSA فهو

← تم استثناء الـ Heap من المقارنة لأنها تحتاج DSA بأضافات وجبر أكثر
complete Binary Tree

Subject Algorithms (5) ~~غير كامل~~ * Date No.

* outlines :-

* selection sort
  - code
  - Tracing
  - complixity ~~Analy~~
  - comparission (ق، لعن)
* insertion sort

* selection sort :- why? To Run on Any Data Type

↳ code :-

```
template <class item>
void selection_sort (item a[], int n)
{
  for (int i=0 ; i<n-1; i++) {          O(n^2)
    int min = i;
    for (int j = i+1 ; j<n ; j++)       // O(n)
      if (a[j] < a[min])
        min = j ;
    swap(a[i], a[min]);
  }
}
```

we stored the index → why?

NOTEBOOK

Subject Date No.

* How The selection sort works?

From The Name of The Algorithm "selection"

↳ we will select The min value (or Max, it's depends on the order of sorting)

→ After the selection of needed item we swap it

→ Then we will logically devide the Array into → sorted
→ unsorted

→ The Relationship between sorted and unsorted IS:

* in Each step The sorted increases By one item

* And The unsorted part dicreases By one item

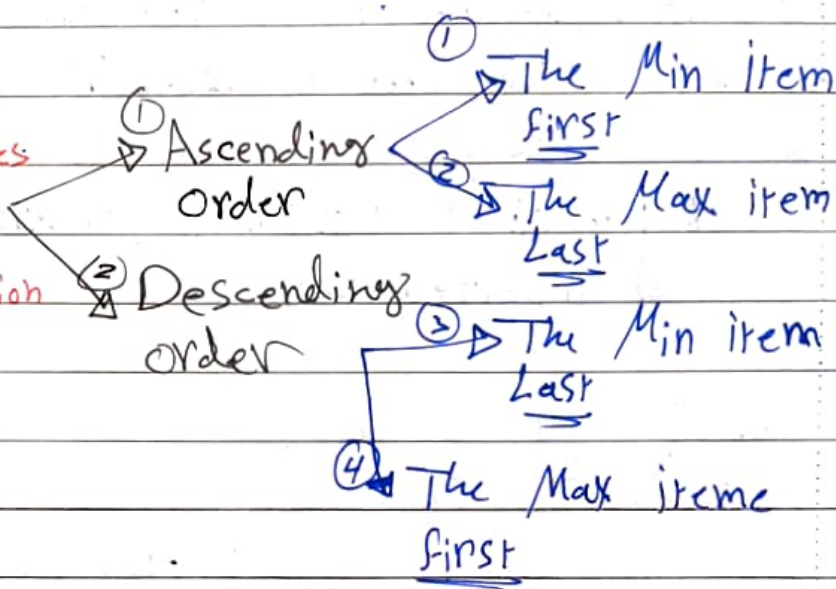
→ While we reach the "n-1" item we dont have to swap it.
→ why? Because, we have already choosen The Min item → So, The Remaining item is Already sorted

NOTEBOOK

Subject Date No.

* Note:- The dividing Process Doesn't Need An Extra data structure
→ we divide on The same Array.

↳ There Are Four Probabilites for sorting by using selection



↳ we will Trace one of Them
The first one it will be.
→ The Remaining 3 is for you :) or):

* comments on The Code:

① " i < n-1 " ⇒ why n-1 ?
The Last will be Already sorted.

② " int min = i; " ⇒ why " i " ?

↳

Subject Date No.

Bec. We store The index of The First value of The unsorted part

⟹ why?

To swap it, And bec the sorted part is Already sorted And we have been choosen The last

e.g

Note: The Selection sort cares about the index The Right Position is The first Element of unsorted part.

(3) " int j = i+1; " ⟹ why ?

we supposed That The first item is sorted Next And we will start compare it with The second item.

(4) " j < n " ⟹ why " n " Not " n-1 " ?

Bec, in The inner Loop we need To search for The Min Value

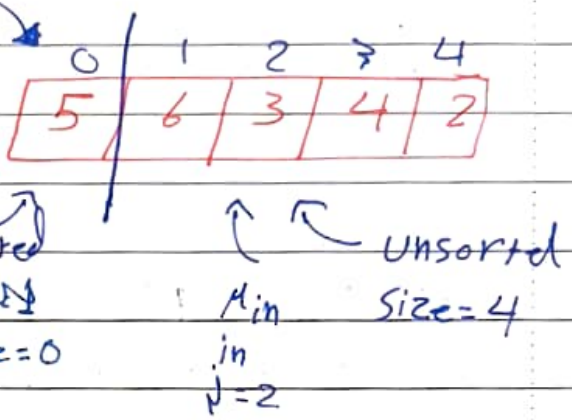
But, in The outer Loop

Subject Date No.

* Tracing :- Consider The Following Array :- 5, 6, 3, 4, 2

→ Phase 1 :- i = 0 → will find the min item Put it in #Min Position

i	j	min
0	~~1~~	~~0~~
	~~2~~	~~2~~
	~~3~~	4
	4	



j = 1

a[j] < a[min]

a[1] < a[0]

6 < 5 = false

j = 2

a[j] < a[min]

3 < 5 = T

min = 2

j = 3

a[3] < a[2]

4 < 3 = F

j = 4

~~a[4] < a[2]~~

2 < 4 = T

min = 4

j = 5 → breals the loop.

→ i = 0

swap(a[i], a[min])

swap(~~a[0]~~, a[4])

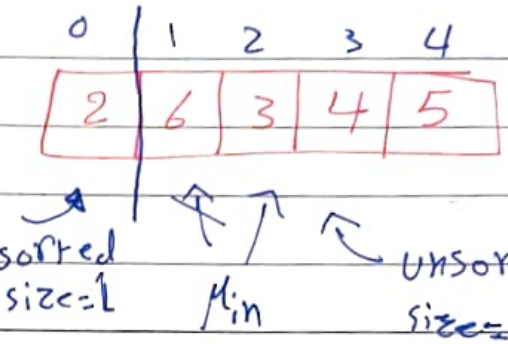
2	6	3	4	5

NOTEBOOK

Subject Date No.

*Phase 2 :- i = 1

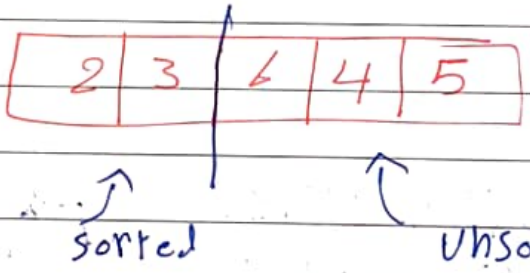
i	j	min
1	2	1
	3	2
	4	
	5	



j = 2

* a[2] < a[1]
  3 < 6 = T
  min = 2

→ swap(a[i], a[min])
swap(a[1], a[2])



j = 3

a[3] < a[2]
4 < 3 = F

j = 4

a[4] < a[2]
5 < 3 = F

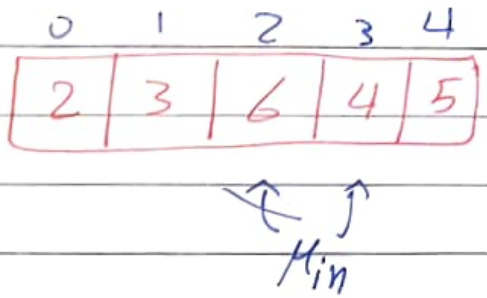
j = 5 → break
The Loop

NOTEBOOK

Subject Date No.

*Phase 3 : i = 2

i	j	min
~~2~~	~~3~~	~~2~~
~~3~~	~~4~~	3
2	5	



j = 3

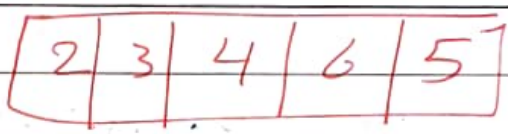
a[3] < a[2]

~~4~~ 4 < 6 = T

min = 3

swap (a[i], a[min])

swap (a[2], a[3])



j = 4

a[4] < a[3]

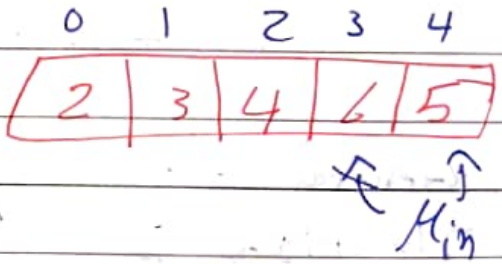
5 < 4 = F.

j = 5

breaks ;

* Phase 4 : i = 3

i	j	min
3	~~4~~	~~3~~
j = 4	5	4

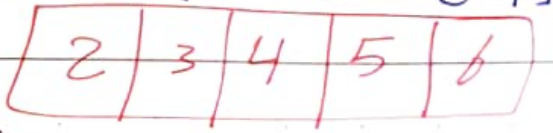


a[4] < a[3]

5 < 6 = T

min = 4

swap(a[3], a[4])



j = 5

breaks;

NOTEBOOK

Subject Date No.

* Phase 4 :- $i = 4$
↳ $4 < 4$ = False
breals The outer Loop.

* The Big-O :-

~~There Are~~ The Big-O is $O(N^2)$

But, There are Two cases
① Asking ~~Ad~~ about The number of swaps + Big-O?
② Asking about Big-O?

→ case 1 :- Specific cases e.g.: sorting 5 numbers.
Answer:
The Number of swaps is The ~~Sum~~ summation of ~~The~~ The n
↳ Not The items
Ex:

10	7	4	2	5

Number of swaps:
$1 + 2 + 3 + 4 + 5 = 15$

→ case 2 :- General code
The Big-O is $O(N^2)$

Subject Date No.

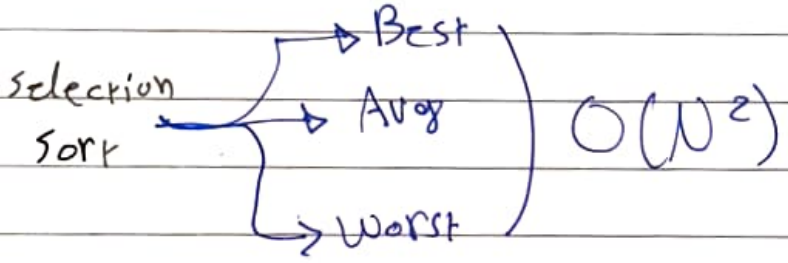
* Worst, Avarge and Best Case for The Selection Sort?

* The worst Case = Best Case = Avg Case

* There is No Best or Avg case in The selection sort

⇒ Why? The code will alwayse be running even The reorder ➤ it's sorted, or even if it's descending or

↳ Make some Tracing



NOTEBOOK

Subject Date No.

*Insertion Sort :-

↳ code :-

```
template <data item>
Void insertion.Sort (item a[], int n)
{
  for(int i = 1; i<n; i++)
    {
      item tmp = a[i];
      for(int j = i; j >0 && tmp<a[j-1];
          j--)
            a[j] = a[j-1];
      a[j] = tmp;
    }
}
```

And operator

NOTEBOOK

* How The insertion works?

↳ In The ~~selection~~ insertion sort we suppose That The first Element is Already sorted

↳ Then we ~~insert~~ Move The first Element from The unsorted Part To the sorted Part in the Proper ~~Posis~~ Position

* Selection sort

- The First index is Reversed (~~...~~) for The Min Element
- We ~~ser~~ search in The unsorted Part for The Min value
- The Last Element is Already sorted
  ↳ why? See Page 2

* insertion sort

- The first Element is arbitrarily sorted
- we Take the first Element we Have in The unsorted Part And Put in it's Proper ~~of~~ Position in The sorted Part
- we Have to checks about The Last Element
  ↳ why? ~~we~~ we Don't know what is The Value of Last Element

NOTEBOOK

Subject Date No.

✱ comments on The code:

① " int i = 1 " ⇒ why ?

Because we supposed That The first Element is sorted

② " i < n " ⇒ why?

To checks for All Elements

③ " temP = a [i] " ?

To store The first Element in The unsorted part

④ " int j = i; " ⇒ why?

To compare The first element in The unsorted Part Awt Put it in it's Proper Position.

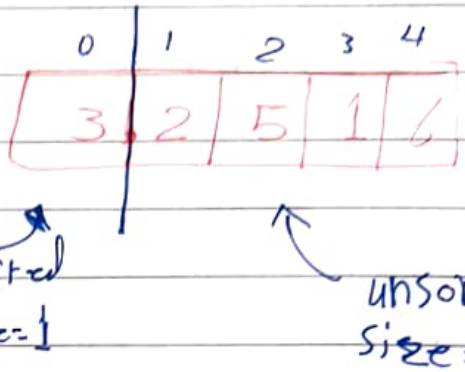
⑤ " a [j] = a [j-1] ; " ⇒ why?

For shifting Elements To The write

Subject Date No.

* Tracing :- suppose The Following Numbers: 3, 2, 5, 1, 6

* Phase 1 :- i = 1



i	j	tmp
1	~~1~~ 0	2

j = 1

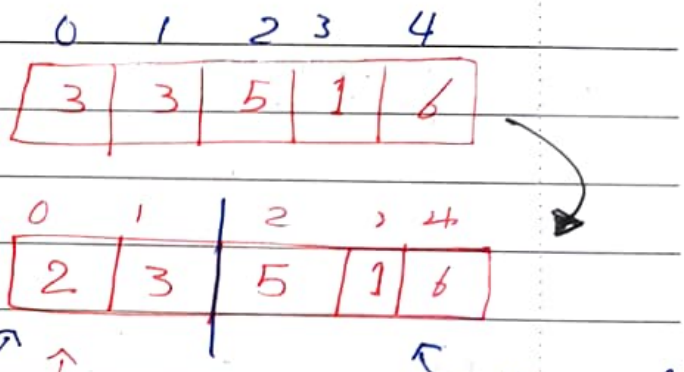
1 > 0 And 2 < 3 = T

a[1] = a[0]

2 = 3

0 > 0 And - - = F

break;



a[j] = tmp

a[0] = 2

After The End of The inner Loop

* Phase 2 :- i = 2

i	j	tmp
2	~~2~~ ~~1~~ 0	5

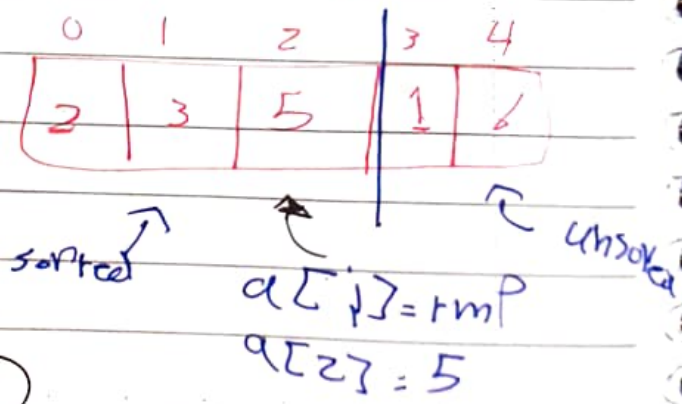
j = 2

2 > 0 and 5 < 3 = False

~~1 > 0 and 5 < 2 = F X~~

~~0 > 0 and - - - = F X~~

break;

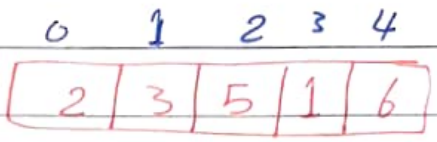


NOTEBOOK

Subject Date No.

***Phase 3 :-** i = 3

i	j	tmp
3	3 ~~2~~ ~~1~~ 0	1

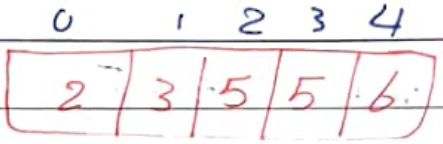


j = 3

3 > 0 And 1 < 5 = T

a[j] = a[j-1]

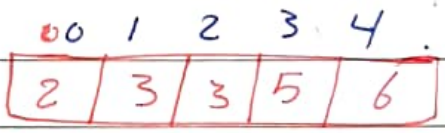
a[3] = a[2]



j = 2

2 > 0 And 1 < 3 = T

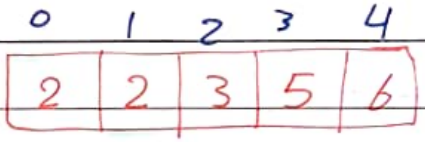
a[2] = a[1]



j = 1

1 > 0 And 1 < 2

a[1] = a[0]



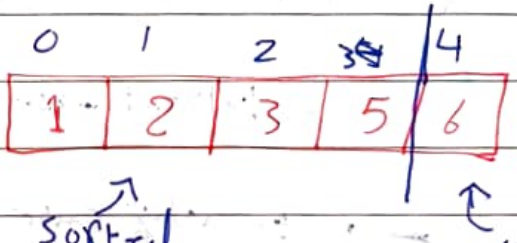
j = 0

0 > 0 And ... = F

breaks;

a[j] = tmp;

a[0] = 1;



Subject Date No.

*Phase 4 :- i = 4

i	j	tmp
4	4	6

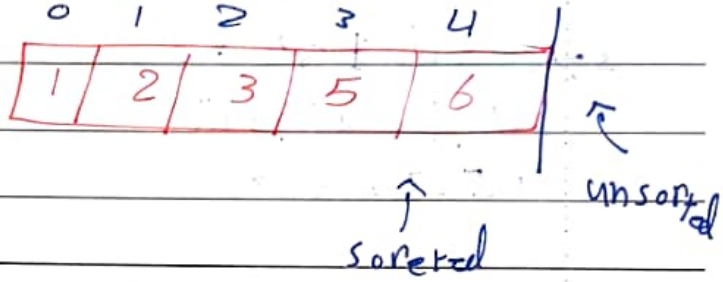
j = 4

4 > 0 And 6 < 5 = F

breaks;

a[j] = tmp

a[4] = 6



*Phase 5 :- i = 5

i < n

5 < 4 ? = F

breaks;

* Best, Avg, And Worst cases for the insertion?

- insertion sort
  - Best case → if the n-numbers Are sorted → O(N)
  - Avg case → semi-sorted → $O(N^2)$
  - Worst case → unsorted → $O(N^2)$

Subject Date No.

* How To work in The Exam? :-

↳ suppose I The following Queistion :-

*Q :- You Have To sort n-numbers, what you will you choose selection or insertion and why?

*A :- insertion, bec if the items is Sorted it will take O(N)

↳ Note :- you Have to compare All Aspects of diffrent Algorithms To get The correct Answer

Q :- Give a case which The selection sort is better Than The insertion sort?

*A :- If The AABU Want to Put (n) Cameras on The (m) Buildings Then you Have To select only n buildings, so which Algorithms selects The Max or Min Buildings (items)?

↳ The selection sort selects The n items then stop.

But the insertion you have to iterate on All The items each Time

NOTEBOOK

Subject Algorithms (6) المحاضرة # Date No.

*Bubble sort :-

↳ Code :-

```
template <class item>
Void Bubble_sort (item a[], int n)
 {
    bool sorted = false;
    int last = n-1;
   for (int i=0; (i < last) && ! sorted; i++)
     {
     sorted = true;
     for (int j = last; j > i; j--)
      {
        if (a[j-1] > a[j])
         {
            swap(a[j], a[j-1]);
            sorted = false;
         } // End of if
      } // End of inner loop
   } // End of outer loop
} // End of The function
```

Subject Date No.

* How The Bubble sort works ?

The idea behinde The bubble sort is That I am dealing (ظلم) with Bubbles

* Sorting The smallest Bubble → In The Front / In The Back

* Sorting The biggest Bubble → Front / Back

↳ We have two types of Bubble sort

1. Bubble sort
2. Modified or Improved Bubble sort

① Bubble sort :-

The Main Idea is to compare Between Two Adjacent items To The End of The Array
↳ And in Each Step we Put The Biggest item to The End

Subject Date No.

*Simple Tracing for Bubble sort (Not The Code):-

↳ suppose you have 5 to sort e.g. (5, 2, 4, 6, 1)

*Phase 1:-

(5, 2, 4, 6, 1)

5 ? 2 : T

(2, 5, 4, 6, 1)

5 ? 4 = T

(2, 4, 5, 6, 1)

5 ? 6 = F

(2, 4, 5, 6, 1)

6 ? 1 = T

(2, 4, 5, 1, 6)

*Phase 2:- (2, 4, 5, 1, 6)

2 ? 4 : F

(2, 4, 5, 1, 6)

4 ? 5 : F

(2, 4, 5, 1, 6)

5 ? 1 = T

(2, 4, 1, 5, 6)

Subject Date No.

* Phase 3 :-

(2, 4, 1, 5, 6)

2 > 4 = F

(2, 4, 1, 5, 6)

4 > 1 = T → True Means swap

(2, 1, 4, 5, 6)

* Phase 4 :-

(2, 1, 4, 5, 6)

2 > 1 = T

(1, 2, 4, 5, 6)

#

(2) The Modified Bubble sort :-

The Modified Bubble sort supposes. That The Array is Already sorted After The first iterate (The outer Loop) → iterate
↳ And if There is an any swapping operations Then ~~The Modified Bubble sort~~ The Array is not sorted and the flag (The sorted bool variable) will be false

NOTEBOOK

Subject Date No.

*comments on The Code :-

① "sorted = false" ⇒ why false?
We suppose The Arr is sorted After The first iterate

② " ! sorted " ⇒ why?
Bec, we Have start The first iteration Then we Have to suppose That The Array is sorted.

③ "sorted = false" ⇒ why? (The inner Loop)
If we Make any swapping operation Then The Arr is not sorted ⇝ so, sorted Must be false.

NOTEBOOK

Subject Date No.

* Tracing :- suppose The following Numbers
(5, 2, 4, 6, 1)

* Phase 1 :- i = 0
j = 4

i	j	sorted	last
0	~~4~~	~~F~~	4
	~~3~~	~~F~~	
	~~2~~	F	
	~~1~~	~~F~~	
	0	~~F~~	
		F	

0 1 2 3 4
(5, 2, 4, 6, 1)
6 > 1 = T

(5, 2, 4, 1, 6)
4 > 1 = T

(5, 2, 1, 4, 6)
2 > 1 = T

~~(5, 2, 1~~
(5, 1, 2, 4, 6)
5 > 1 = T

(1, 5, 2, 4, 6)

* Phase 2 :- i = ~~2~~ 1
j = 4

i	j	sorted
~~2~~ 1	~~4~~	~~F~~
	3	~~F~~
	2	F

0 1 2 3 4
(1, 5, 2, 4, 6)
4 > 6 = F

(1, 5, 2, 4, 6)
2 > 4 = F

(1, 5, 2, 4, 6)
5 > 2 = T → swap

Subject Date No.

i	j	sorted
1	4	T
	3	T
	2	F

(1, 2, 5, 4, 6)

Phase 3 : i = 2

j = 4

i	j	sorted
2	4	T
	3	T
	2	F

0 1 2 3 4

(1, 2, 5, 4, 6)

4 > 6 = F

(1, 2, 5, 4, 6)

5 > 4 = T

(1, 2, 4, 5, 6)

Phase 4 : i = 3

i	j	sorted
3	4	T
	3	T

(1, 2, 4, 5, 6)

5 > 6 = F

Phase 5 : i = 4

i	j	sorted
4	4	F

i < last

4 < 4 ⇒ F

breaks;

Subject Date No.

* Best, Avg and worst Worst case for Bubble sort:

Bubble sort
- Best when The Array is sorted → Big-O: O(N) → The inner Loop
- Avg case → We can't compute The complixity very well so, it will be Big-O: O(N²)
- worst case → Big-O: O(N²)

* Note: Try to trace an examples for The best and Avg cases.

* Ex (5, 7, 7, 9) , (5, 2, 3, 6)

Best case sorted

Avg case semi-sorted

* How to work in The Exam?

* Q: suppose The Prev code (in Page 1)

1) what does the code do?

Sorting

Subject Date No.

2) what is The Big-O?

if The Array is sorted the Big-O is O(N)
and if the Array is Not sorted ⇒ O(N²)
and if the Array is semi-sorted ⇒ O(N²)
Worst case

*Note- لازم نتذكر كل حالات الخوارزمية عند سؤال از Big-O ⇐ لازم ايضاً نتذكر حالة الـ Avg

3) what is The Best Algorithm Among These Three ? selection, insertion, Bubble

insertion sort

→ Bubble sort, because if The Array sorted The Big-O is O(N)
And The Algorithm is simpler to implement

↳ why?

* The selection is Always O(N²) ⇒ So it's Excluded (هيتعب)

→ Both insertion And Bubble sort has a Best case, which one is better?

↳ you Have to look up for Athere Aspects, And Here The Bubble is a simpler insertion

Subject Date No.

*Q: suppose (1, 2, 4, 5) what is The Big-O?

O(N)

If The Algorithm sorting the number in Ascending order

↳ But if The Algorithm sorting the Numbers in Descending order The big-O is $O(N^2)$

↳ There Are several Technique for Designing The Algorithms, Such as:

1. Brute force → Try All Possible Solutions
2. Greedy → أكثر سريع وأقل خسارة
3. Branch And Bound
4. Dynamic Programming
5. Divide And Conquer

* stars what we will study

NOTEBOOK

Subject Date No.

* Divide And conquer

- Divide → How to divide The Problem? / Keep continued on The same dividing Technique in The First divide
- conquer → What is The solution?
- combine

* The sorting Algorithms That is Based on Divide And conquer
  - Merge sort
  - Quick sort

* All work is done in The Hard Seation
  - Merge sort
    - Easy Division
    - Hard combination
    - why? =>
  - Quick sort
    - Hard Division
    - Easy combination

why?

NOTEBOOK

Subject Algorithms (7) Date No.

* Merge sort :

↳ Code : To Review The implementation of Merge sort, See :

→ "www.geeksforgeeks.org/merge-sort"

→ "stackoverflow.com/questions/66284003/"

→ Merge sort Algorithms is a sorting Algorithms for sorting items by using The Divid-and-conquer Techinque

* What is The different Between The Merge sort And The Merge Technique ?

→ The Merge Technique is used for sdving Problem By

1. Divide The problem into subProblems
2. Find a solution for Eeach subProblem
3. combine (Merging) The subsolutions Together

↳ Note : Merge Technique as same as Divide and conquer Tech

NOTEBOOK

Subject Date No.

↳ OKs, We Have Three Options :-

① using Merge Tech for solving a Problem And using The Merge sort To find The subsolutions

② using Merge Tech for solving a Problem And using Any other sorting Algorithm such as (insertion, selection, ---) To find The subsolution.

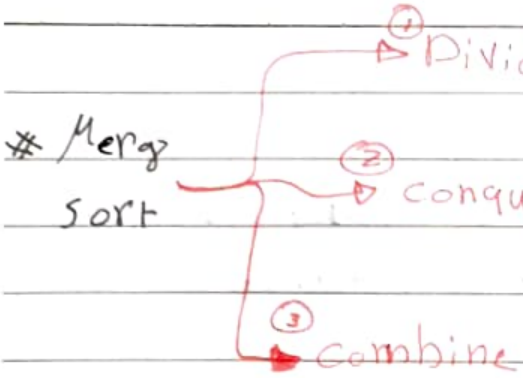
③ using ONLY The Merg sort
↳ which is based on divide And conquer :) Loopback

↳ keep in Mind Two Things :-

① Primal substructure :-
To Preseves ( ) The Main structure of The Problem.

② Primal subsolution :-
The subsolution is embeded in The final solution OR The final solution consists of subsolutions

NOTEBOOK



1 Divide :-

→ The Dividing Tech in The Merge sort is Alwayse Constant

→ We Divide By n/2 until we reach n/n for each subArray
↳ which is 1

→ The General Merge sort keeP dividing until n/n, But you can stoP dividing Before That in Any level.

→ The Dividing is working by using The Recursive structure.

→ How to divide?
[first, mid] → First sublist.
[mid+1, last] → second sublist

Subject Date No.

(2) Conquer :-

→ We compare Each ~~subl~~ Two sublists Together To determine The Min value ~~for~~ Between Them

→ The comparing operation is The Solution

(3) combine :-

↳ After finding The Min value between Two sublists, we Put it in The it's Position In The Previous sublist

~~→ We combine Recursir~~

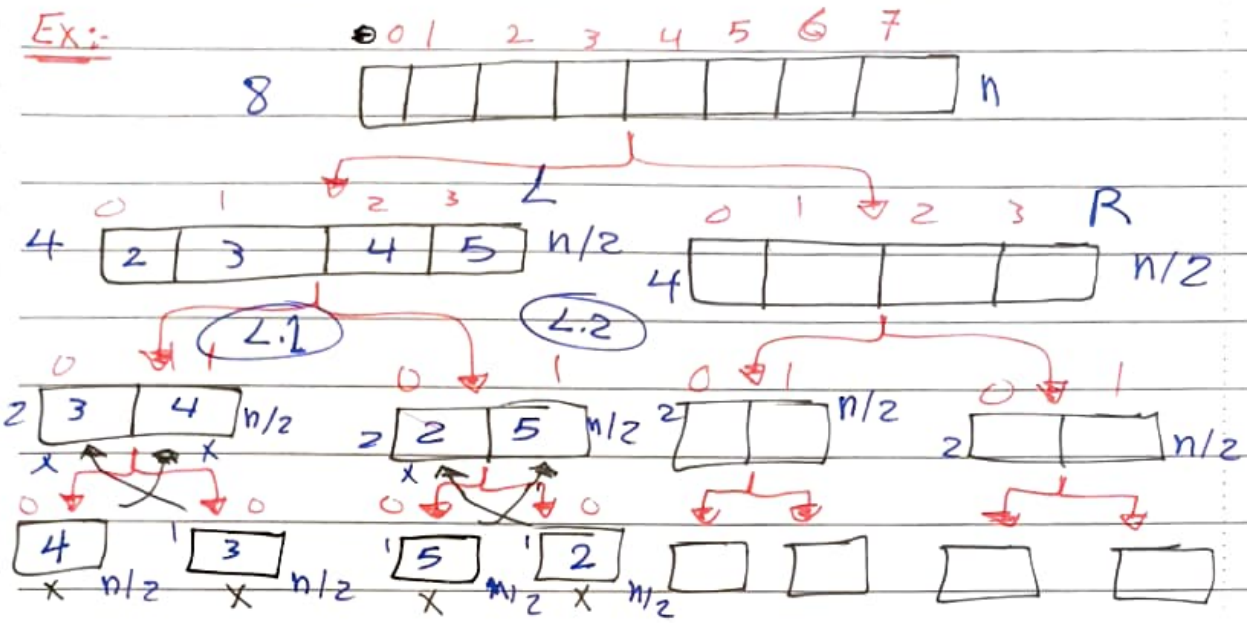
→ We Recursivly combine Each Two sublists untile we Reach The Main List.

Notes :-

↳ (1) There is no diffirent ~~between~~ between Greater than (>) or Less Than (<) in The conquer step.

Subject Date No.

Ex:-



① $4 < 3 = F$ Then 4 Greater Than 3

OR

$3 < 4 > 3 = T$

② $5 < 2 = F$

→ Then we will compare The L.1 and L.2

③ $3 < 2 = F$

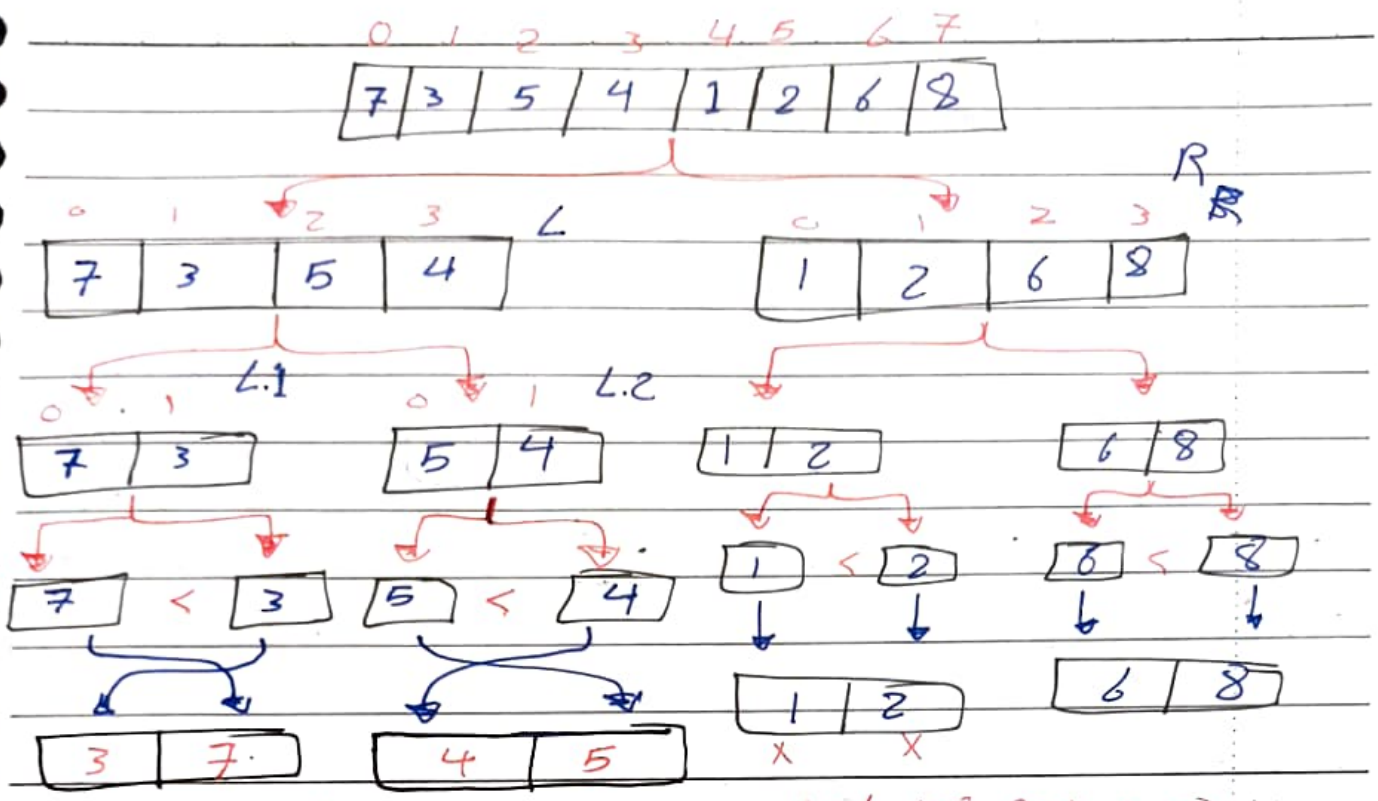
④ $3 < 5 = T$

⑤ $4 < 5 = T$

⑥ nothing To compare with 5 so it's Max one, And if there is More Elements will be The Same

* Note That we compare The L.1 and L.2 and They are Already sorted

NOTEBOOK

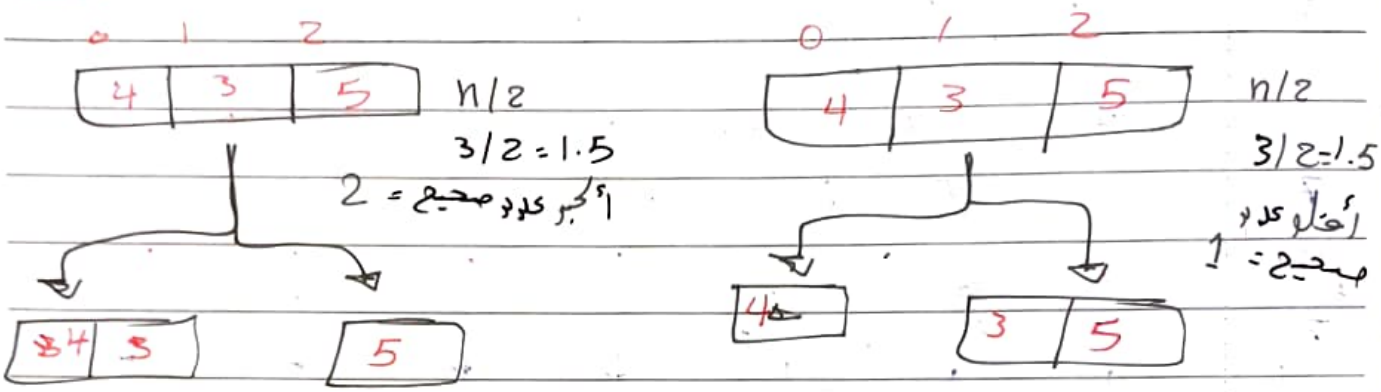


← باجي أول كل شي combine لا يعني انو كملنا DSA زيادة وإنتهى و للتوضيح

Subject Date No.

* What if The "n" is Odd?

Ex:-



↳ Are They The same?

Yes, How on Why?

↳ Bcc They Have The same cost? ⇒ How?

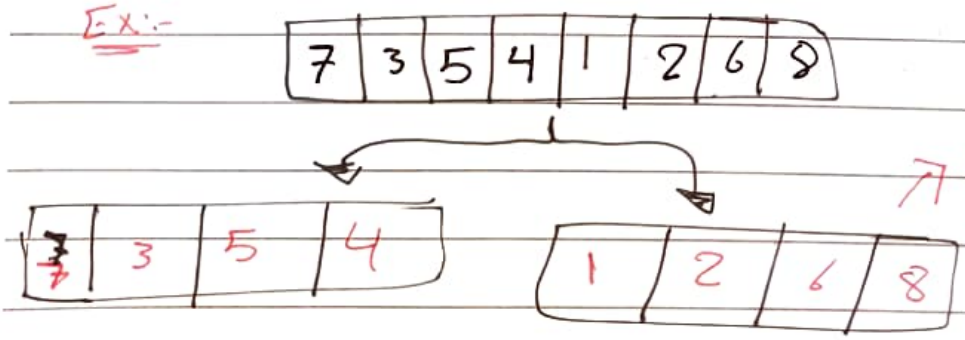
↳ The Poesrit Require An Extra DSA

So, They Have The same cost

So, They are The same.

* If one of The sublist is Already sorted Can We just not to divide it?

Ex:-



NOTEBOOK

Subject Date No.

↳ No, you Have to divide Them ⇒ Why?

*For Two Main Beasons:

① How can I Know if The sublist is sorted?
↳ we Have to iterate over The Elements of The sublist ⇒ which Takes More cost

② The Main idea of The Merge sort is To divide The DSA into sublists And if you Don't Do That Then Why Not to use other sorting Algorithms.

* Case Study:

①

3	7	9	11	12

↳ what is The Big-O?
O(n log(n)) ⇒ why?
↳ we have To divide conquer combine But, we don't make any comparsions so, it will be An Restoring operation

②

12	11	9	7	3

↳ what is The Big-O?
O(n log(n)) ⇒ why?
↳ Divide-conquer-combine

* Note: There is No Best case in The Merge sort

NOTEBOOK

Subject Date No.

*suppose The Example in Page 8, if in The Before last step insted of storing in The Array We need to store in Queue Stacks, Linked-list.... will still The same cost, less or greater?

يمكن يكون سؤال في الاختبار ← الجواب بعد الدرس أو في الفايينال ← الجواب بعد الفايينال.

← الجواب: دور على سؤال مشابه للدرس

*Notes:

① ← أنت غير مضطر تكتب Psedcode وإذا كتبت وكان في أخطاء برمجية سوف تحسب عليها ← ممكن أكتب Psedcode إذا كانت الفكرة كاملة بأسلوب code أو كتابة

مثال: أكتب من الشرح خوارزمية لتحزين أسرع

② ← يمكن أغلب عمل الـ shift بدل الـ swap في Selection sort وج = غير خوارزمية أخرى + 8 = عمل swap في الـ Merge sort

NOTEBOOK

Subject Algorithms (8) المحاضرة # Date No.

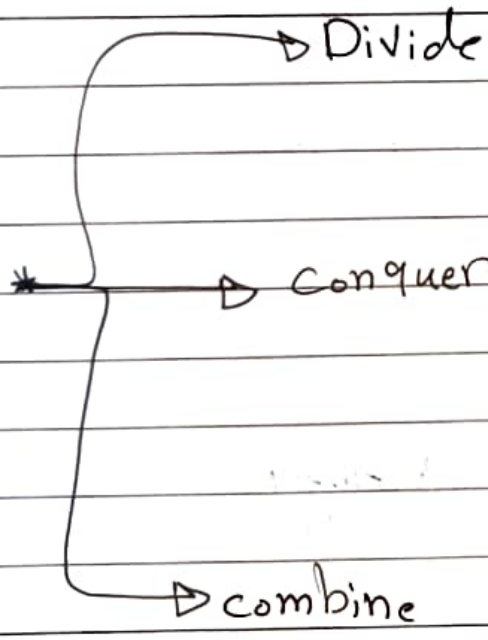
*Quick Sort:

↳ code: Review The Following Resources:

→ See Pages 170-172 from The Book "Introduction to Algorithms - MIT Press"

→ "www.geeksforgeeks.org/quick-sort"

↳ Quick sort is an Sorting Algorithm Based on Divide-and-conquer ~~Tech~~ Tech.



NOTEBOOK

Subject Date No.

* Divide :

→ in The Merge sort we were divide ~~always~~ The list By using n/2 ⇒ and That was ~~as~~ constant for all ~~cass~~ cases

↳ → The divide is The Hard works in The Quick sort

→ We divide Here Based on some value

→ That value called Pivot

→ we will study the case where The Pivot is arr[0] (The first Element)

→ And Then we split The list into Two list Based on The Relationship between Element and The Pivot

Element in List 1	Pivot (X)	Element in sublist 2
≤	x	≥
<	x	≥
≥	x	≤
>	x	≤

Subject Date No.

→ The Relationship we will study is Either



* sublist 1:- less or equal (≤) Pivor

* sublist 2:- Greater Than Pivor (>)

* But, How To know either The Elements are less or equal equal or Greater Than Pivor?

↳ we Have to iterate on The Array to know That

↳ How much it takes?

↳ O(N).

↳ Is that cost will be added on The Algorithm or it will be Embedded in The Algorithm?

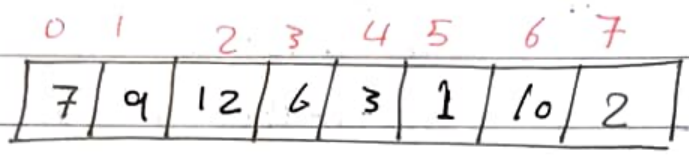
↳ it should be Embedded.

// a Part of The Algorithm Not an Extra oPreation //

in other words we will use That iterate for other things rather than dividing

Subject Date No.

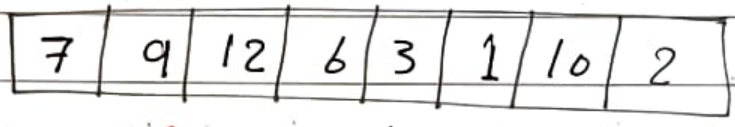
Tracing: suppose The following Numbers To sort Them
(7, 9, 12, 6, 3, 1, 10, 2)



Pivot = arr[0]
= 7

first ≤ 7 = T first++
7 ≤ 7

last > 7



first ≤ 7

last > 7

* 9 ≤ 7 = F

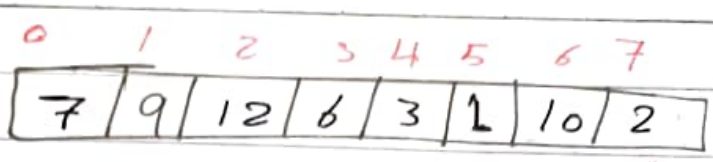
↳ So, because The 9 is bigger Than 7 we need to find a Place To Put The 9 in The seconde sublist

↳ But what is That Place?

↳ The Position of a number is less or equal To The 7

NOTEBOOK

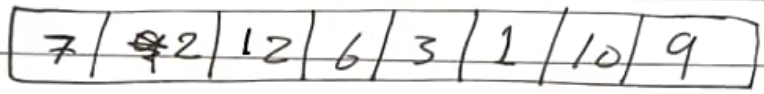
Subject Date No.



9 < 7 = F

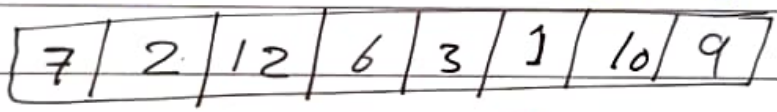
last > 7
~~last~~ 2 > 7 = F

swap(9, 2)
swap(arr[1], arr[7])
first ++, last --;



first < 7
12 < 7 = F
first ++

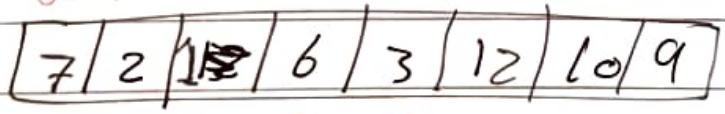
last > 7
10 > 7 = T
last --;



12 < 7 = F
first ++

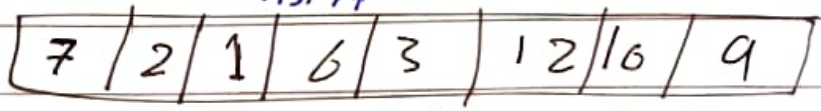
1 > 7 = F
last --;

swap(1, 12);



6 < 7 = T
first ++

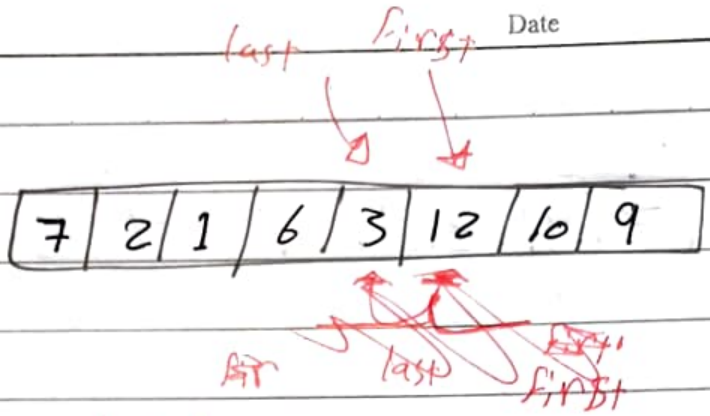
3 > 7 = F



3 < 7 = T
first ++

3 > 7 = F

Subject Date Noor: 29 Mi

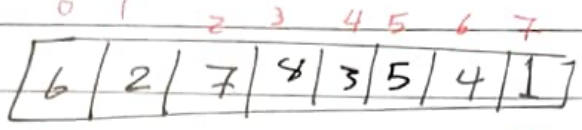


breaks:

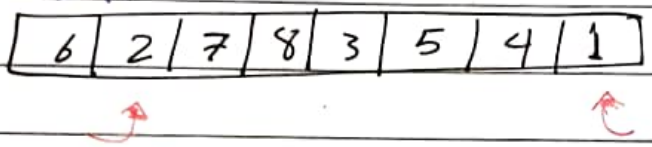
→ if one of the Two Pointers Exceeds (يزيد)
The on Another one we break

Subject Date No.

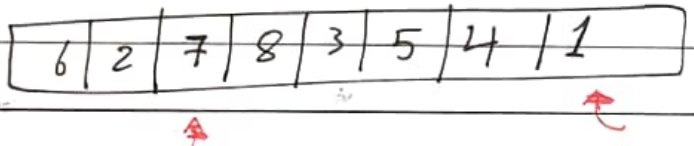
** Tracing 2 :



first → 6 ≤ 6 = T
first ++

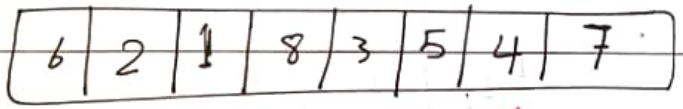


2 ≤ 6 = T
first ++



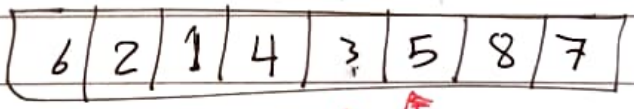
7 ≤ 6 = F
first ++

swap (7, 1)
last --

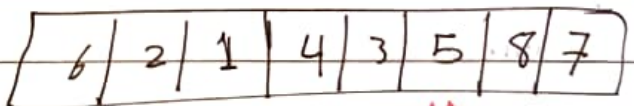


8 ≤ 6 = F
first ++

swap (8, 4)
last --

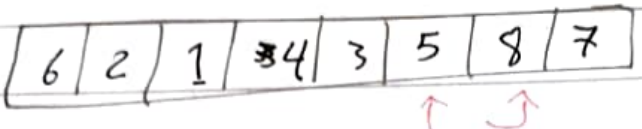


3 ≤ 6 = T
first ++



5 ≤ 6 = T

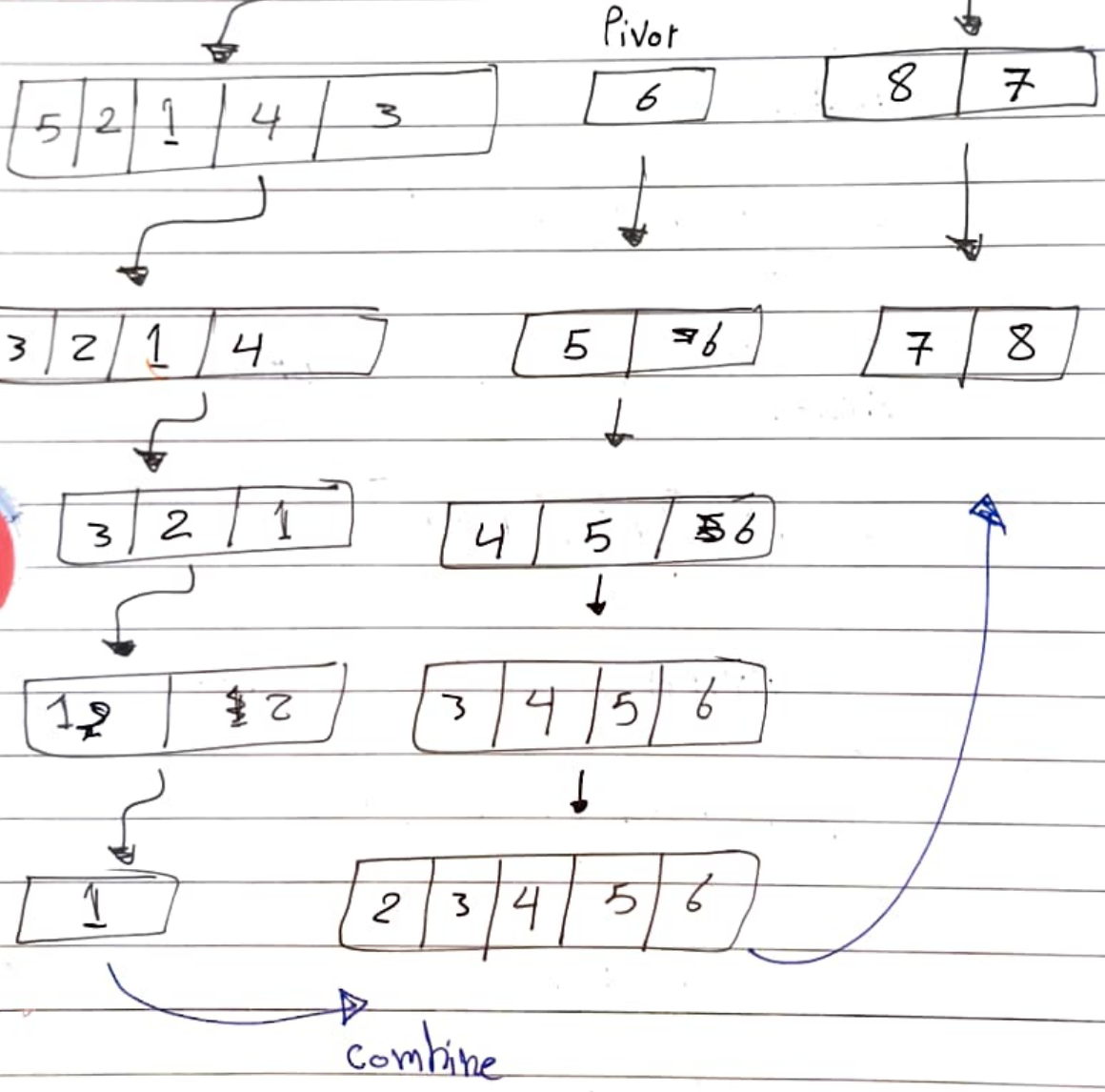
Subject Date No.



last first

* After we split the list into two sublist we swap the last Element with the first one

breaks



combine

Subject Date No.

* which of the Two Algorithms Merge sort And Quicksort is better Than The other?

↳ in General case The Merge sort is better Because we Don't Have a Worst case

⇒ Rather Than The Quicks sort

→ The Quicks sort Have a Worst case when The Array is Sorted

→ The Big-O of the ~~Worts~~ Worst case: $O(N^2)$

~→ But, if you Have a case where you can:

* determine The Pivot
* Have information About The data

↳ The Quick sort is better Than Merge sort

↳ But, why?

- The combining ~~of~~ operation is easier
- And we Have ~~in~~ information About The ~~Pivot~~ Nature of the Data

⇒ So, we can easily Determine The Pivot

NOTEBOOK

Subject Date No.

↳ if ~~The~~ we Don't Have information about The Data And - we choose The Quick sort we May Have The ~~worst~~ worst case which is $O(N^2)$

⇒ When we choose The Quick sort?
if we Have information About The Data
Ex: we sort The Marks in The Algorithms course Based on The Most ~~B~~ frequently Marks 35 ~~$~~ for Example :).

* Note: In The Exam if we Have an Sorting Algorithm sorting The Numbers ~~is~~ in Ascending order ~~And~~ And The Algorithm puts The Larger Numbers in The front And The smallest Numbers in The Backs,
⇒ The cost will be ~~that~~ ~~of~~ changed but not Totally

↳ Ex: The Merge Sort will still $O(n \log(n))$
But The cost ~~with~~ will be ~~different~~ ≠ diffrent

NOTEBOOK

Subject Algorithms (9) Date No.

* Binary Search Tree :

The BST (Binary Search Tree) is a data structure and Have some of Precision Among it.

→ Binary →

* True or false
* Two choices only

→ Tree : The Tree is special type of The Graph, what's is The Graph?

→ Graph : The Graph is Nodes and Arcs or Vertix and Edges But The Essential Part is The Node or Vertix

NOTEBOOK

Subject Date No.

*Types of The Graph:-

(1)  Graph

(2)  Graph Node
Isolated ~~Graph~~; The Most Essential
Part of The ~~Not~~ Graph
~~is Exist~~ ~~does~~ Exisits

(3)  Graph
Loop Graph or Loop Node.
↳ The Loop ~~is a graph~~ is a Node
referes to ~~it~~ it self
→ ~~It~~ Differs from The Cycle

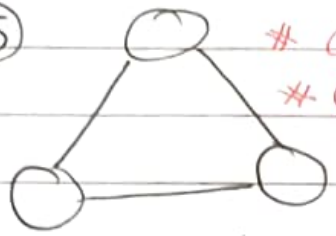
(4)  Not a Graph
↳ Why? The Most Essential
Part Does Not Exisit.

an Edge

NOTEBOOK

Subject Date No.

(5)

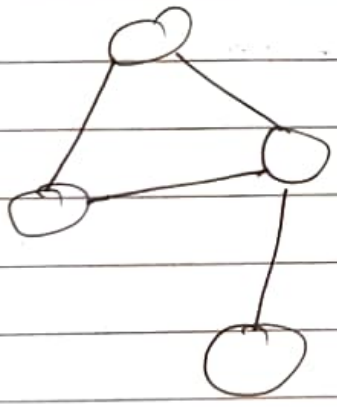


* Graph
* cycle Graph
  ↳ why? If we start in Any Node will back to it (The Start and The End Node Are The same)
* undirected Graph

→ And The degree of ~~Each~~ Each Node is 2

→ The degree is The Number of Edges in and out The Node.

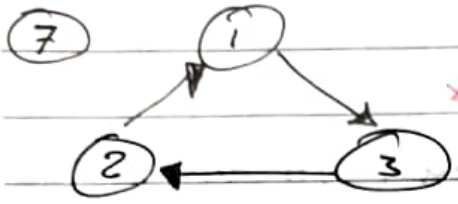
(6)



* Graph
* Undirected Graph
* Not cycle Graph

But, it is a Graph Has a cycle

NOTEBOOK



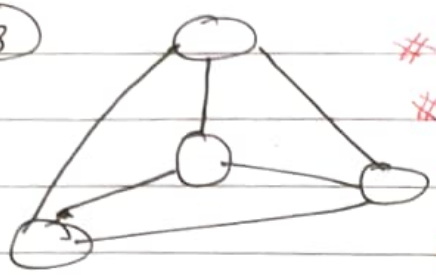
* Graph
* Directed Graph
↳ If we Have Directed Graph we Have Two chocises
1. Strongly Connected
2. Weakly Connected
* This Graph is ~~Weakly~~ weakly connected

↳ Graph
- Connected / Not Connected
- connected
  - Undirected → Always strongly connected
  - Directed
    - Strongly Connected
    - Weakly Connected

NOTEBOOK

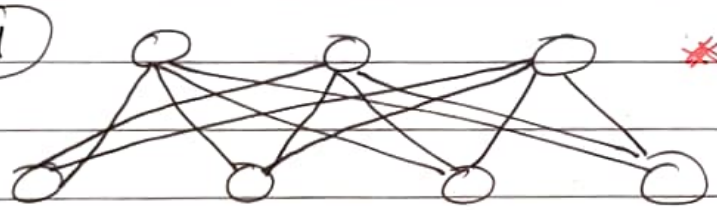
Subject Date No

(8)



* Graph
* connected Graph
* Undicted Graph
* Wheel Graph
  ↳ why? The degree of Each Node is 3
* unWheited Graph
  ↳ The Graph That it is Edges Doesn't Not Have a Cost or Wheight

(9)



* ~~Bipartite Graph~~
  ~~↳ we can't find~~
* complete Bipartite Graph
  ↳ The Graph is divided into Two Parts
  ~~↳ By using The coloring system~~
  → The Graph is divided By using The coloring System

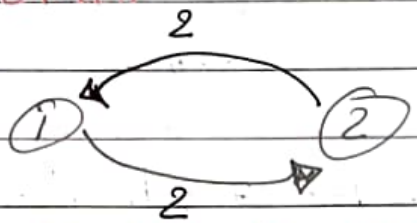
NOTEBOOK

Subject Date No.



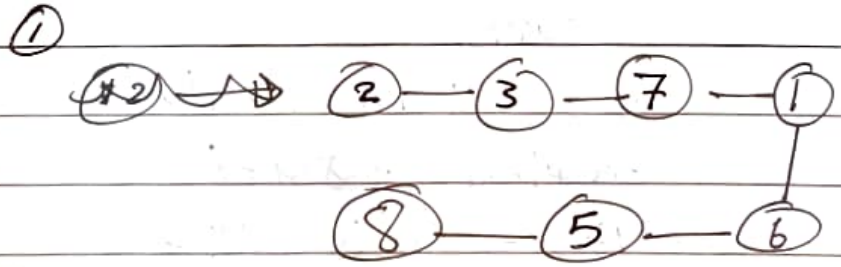
* Bidirectional
* Bidirectional Graph

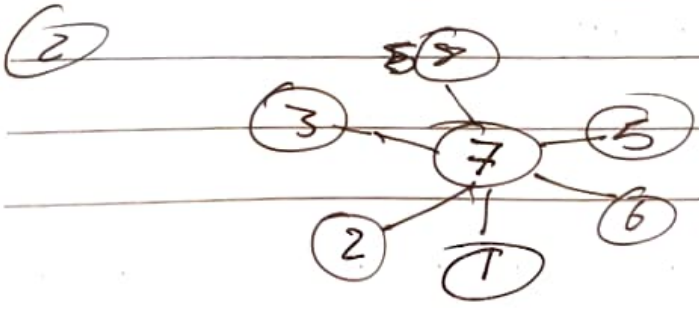
* Note The Bidirectional Graph is different from This Graph.



⇒ Why? The Example of The Two computers in The Network
If A, B are computers in The Network and The connection profled Between A → B
So, Router Have to find a Route Between A and B.

→ Example : Convert The following data into a Graph
(2, 3, 7, 1, 6, 5, 8)





6 → Page Number Not Node :)

Subject Date No.

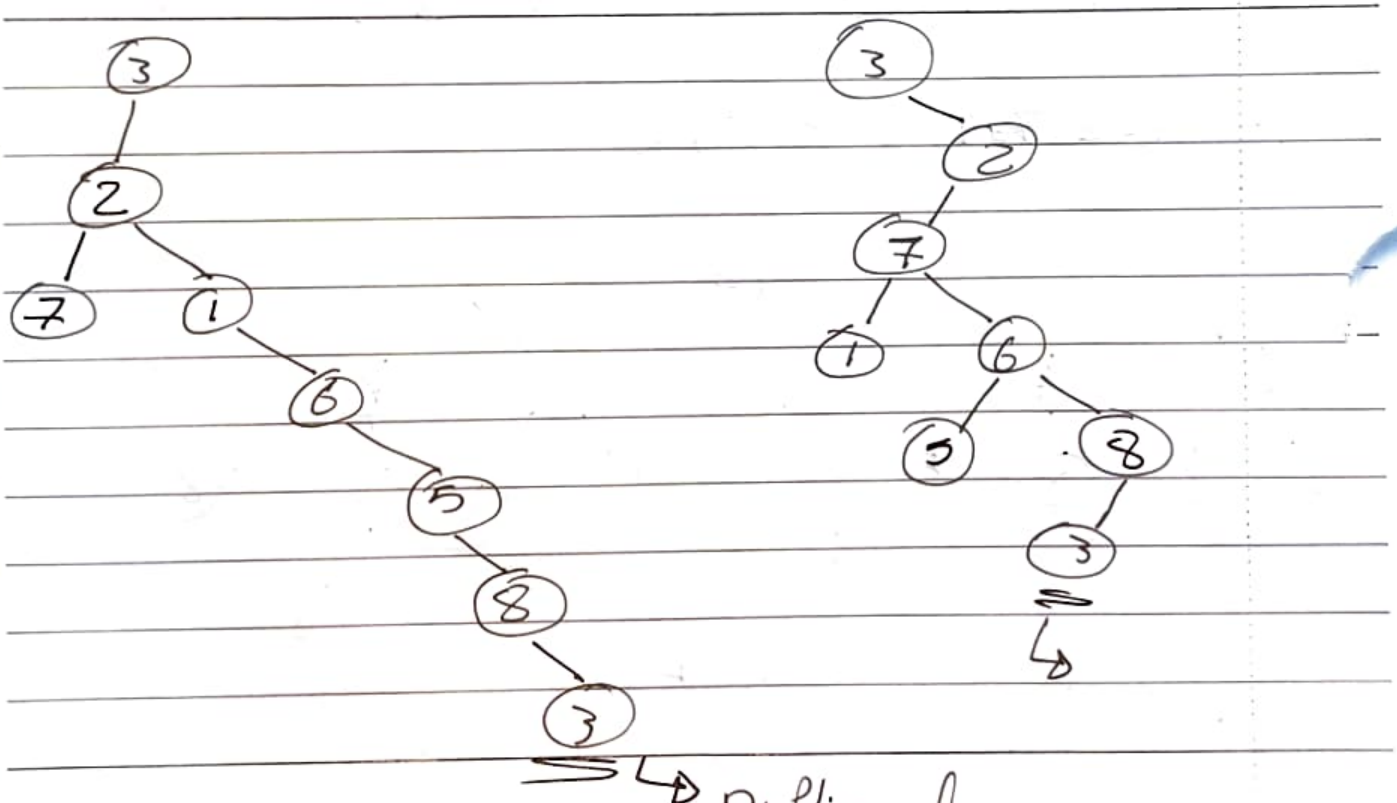
→ There are unlimited Number of Graphs

* How to convert a Graph into Tree:

① Add a Root Node which does Not has Any Parents

② Each Node Must Have only one Parent

*Ex:- conider the Prev Example:- (3,2,7,1,6,5,8,3)



* Note:- values can be Duplicated in The Tree

NOTEBOOK

Subject Date No.

↳ if Mentioned The Root and child Then we Move Moves from Graph to Tree.

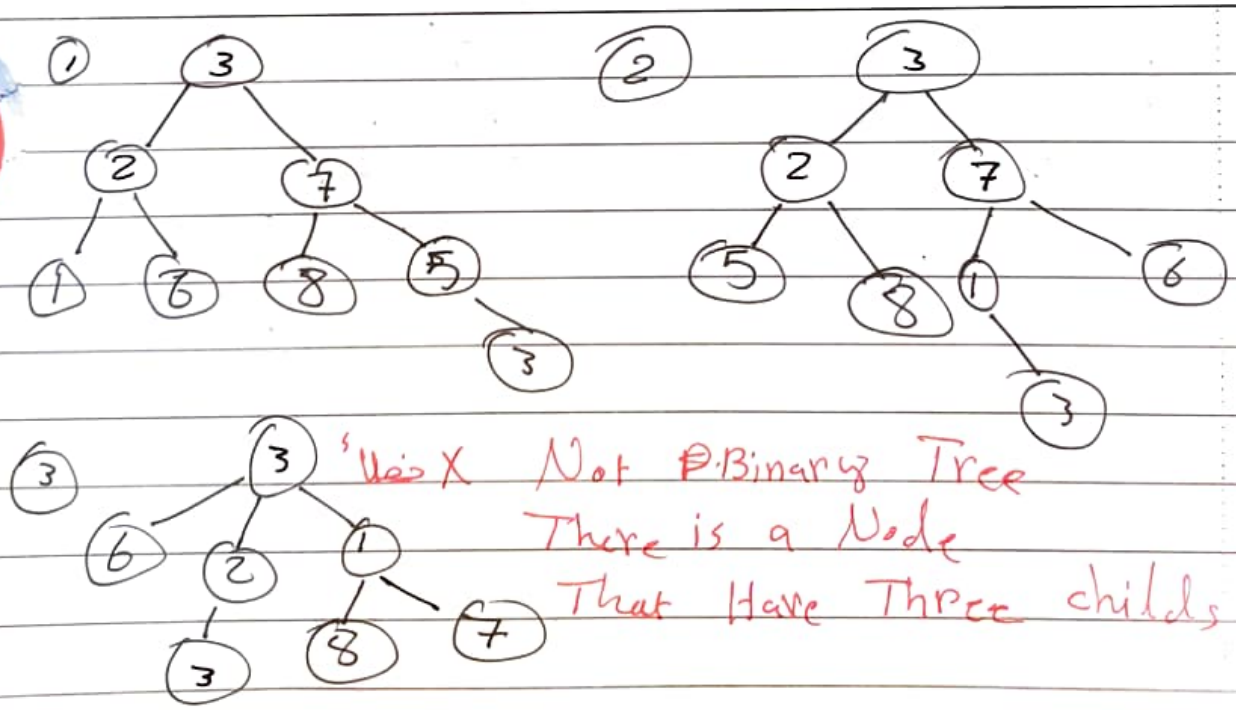
*Note:- We Make unlimited Number of Trees from a dataset

* How To convert The Tree into Binary Tree?

① The Root is The first Element ⇒ a[0]

② The childs for Each Node is ≤ 2 only.

*Ex:- (3, 2, 7, 1, 6, 5, 8, 3)



'Ha' X Not a Binary Tree
There is a Node
That Have Three childs

NOTEBOOK

Subject Date No.

Binary Tree ↖ ↗ Binary search Tree

* How to convert The BT to BST?

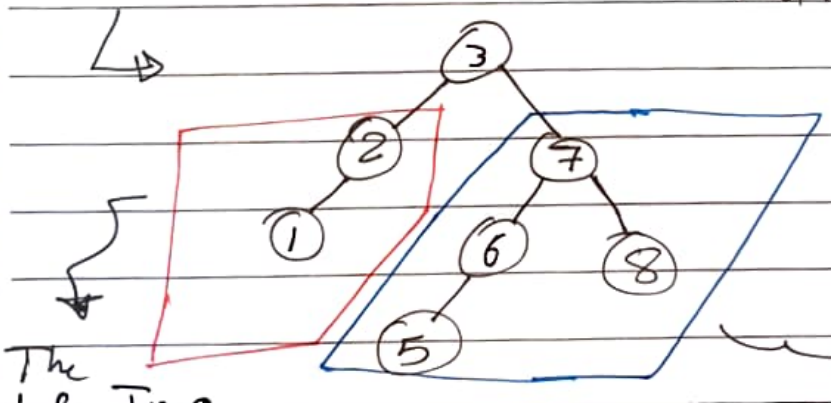
① No Duplicated Values in BST.
Remove

② All Nodes on the left of the Node (<) less than The Parent
And All Node on the Right of Node (>) greater Than the Parent

e.g.: All Nodes in the Left sub-Tree < Root
All Nodes in the Right sub-Tree > Root

* Note: We can Make only one BST from a dataset.

Ex: (3, 2, 7, 1, 6, 5, 8, 3)
Duplicated



The left-Tree is Also BST

The Right left Tree Also BST

Is BST a Graph? Yes

NOTEBOOK

Subject Algorithms (10) المحاضرة الأولى Date No.

* The Project :-

20 Grade for The Project
- 10 grade for Code
- 10 grade for Documentation

=> The Project should solve a real life Application
↳ So, if a student his Project was Just an Implementation for one of The Algorithms ~> it will Take Zero
- if No one of the students Make a Better Project Than he does => He will get a zero.

"But I guarantee he will get zero in The last of The semseter"
- Dr. Mazen Azzouni

=> The Project should solve a real life Application Not Case study

↳ How to get real life APP? Google it :)

NOTEBOOK

Subject Date No.

OR, There is Another choice That is:-

You find a Problem that is solved by a Algorithm and you optimize That solution by using Another Algorithm.

* Project's specifications:- **Code**

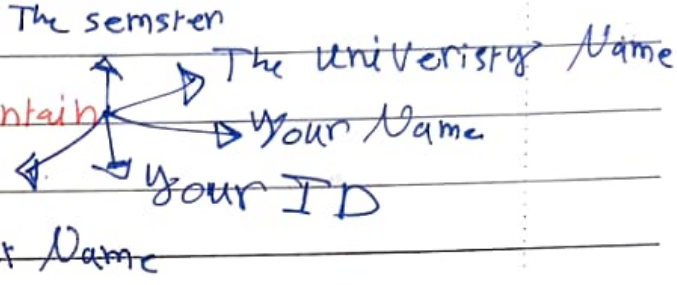
⇒ The code Must be written by using C++
↳ if you choose Another Programming lang Then you Must know:-
- what The functions That are used
- The function's Parameter, returned value and Implementation
↳ He May ask you "Remove a function and write a Alternative one"
- for Example a function take a parameter by reference ⇒ Q:- Why this Parameter is passed by reference in your code?
- or, delete calling by reference and Make it by value, And Modify your code.

Subject Date No.

* The full understanding of your code is a better chance to Get More Marks.

Documantiation

- start by creating a word file
- ~~Fir~~ 2-3 pages
- The first page should contain



- The seconde page
  - Contains The ~~Fabel~~ Table of Contents

- The Third page
  - Contains References
    - Min 3 references
    - The following ARE Not references:
      - youtube, wiki
      - Geeks for Geeks
      - stack overflow

- suppose you wrote The following text:

"The seleetion sort is an Algorithm for sorting numbers in ~~as~~ asending order and it's the ~~best~~ best Algorithm for sorting" → No Reference is Mentioned

-1 -1 -1 -1

NOTEBOOK

Subject Date No.

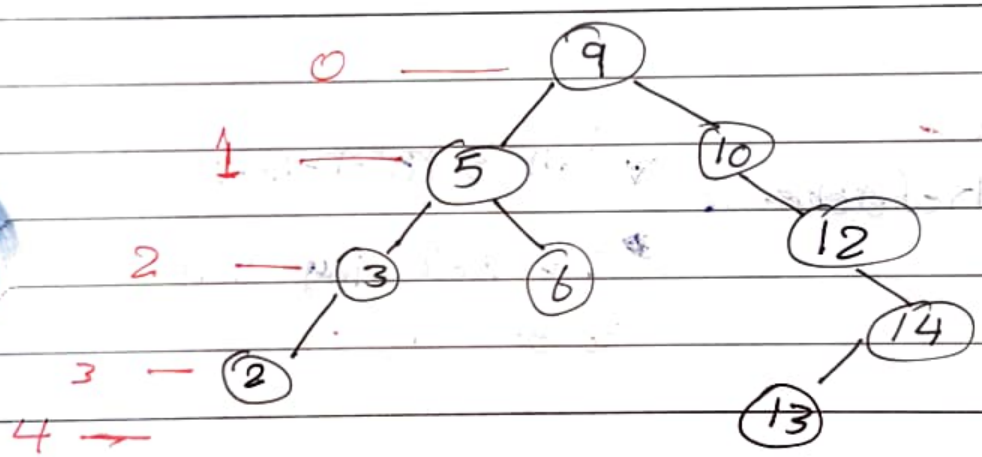
~~The Editor~~

⇒ The Earlier you submits your The Project The More you will Get Notes The More ~~your with the~~ get Marks.

* The BST :-

↳ The Huffmancode is Mentioned Here but it will be Explained Later

→ consider The Following ~~Tree~~ BST :-



* The Number of the Levels is = 5 starting from 0

* The Max number of Nodes in each Level is 2^n, where n the Level number

Ex: Max Number of Node in Level 2 = $2^2 = 4$

NOTEBOOK

Subject Date No.

↳ The Path :-

↳ In BST There is only ONE Path for Each Node

The Path leads From a Node To Another Node

↳ Why?

"Because The BST was build Based on Greater Than or Less The Root

Q: in The Exam:-

How Many Paths for a Node in BST and Why?

↳ Ex:- Consider The Prev BST, Which of The Following is Path for The Node "2"

(1) 13-14-12-10-9-5-3-2 X

(2) 6-5-3-2 X

(3) 9-5-6-5-3-2 X

(4) 9-5-3-2 ↘ ⇒ The Path

Q: Can We have a Path starts From "13", Why?

No, Because The BST starts from The Root which is The First Element in The array and The Node "13" is Not Root.

NOTEBOOK

Subject Date No.

~~Day~~ 8

* Depth :- The Number of Edges from The Root To a Specific Node
  → In The BST The Depth = Level

* Hieght :- The distiance from a specific Node To The Root
  ↳ in The BST The Hieght = Level

Level = Height = Depth.

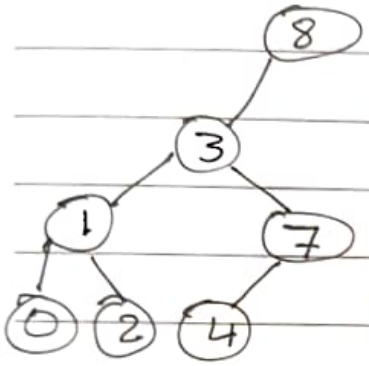
NOTEBOOK

Subject Algorithms (11) محاضرة ١١ * Date No.

* Operations on BST:-

⇒ Build a BST from the following data

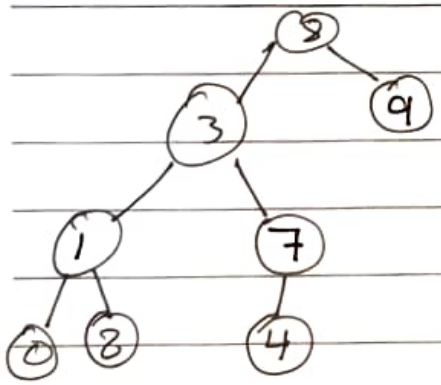
[8, 3, 7, 4, 1, 2, 0]



* start from selecting the Root which is a[0] = 8
* Then compare the Next element with the Root first > or <
and Then with Next Nodes until There is No Nodes to compare with
* what is the complixity for searching on the Prev Tree?

O(N) ⇒ Why?

↳ The BST has only one left side, so how we will get log(n) and there is No Partioning of the Tree



The complixity is O(log(N))

But if we Modify The Tree as following, it will be log(N)

NOTEBOOK

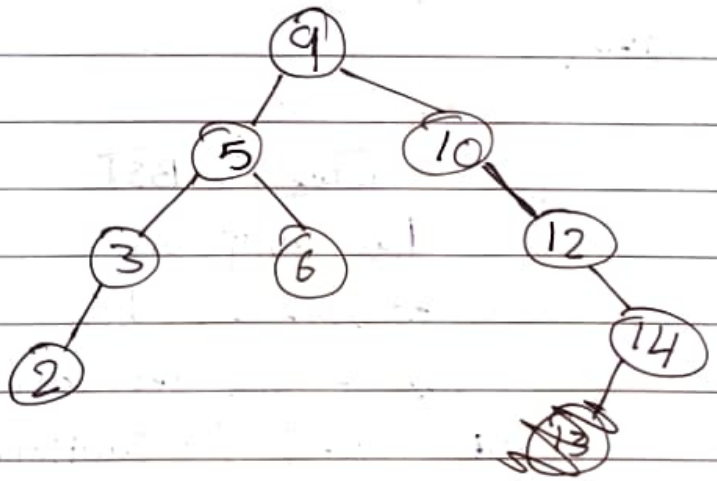
Subject Date No.

* operations on BST → Traversal, Search, Deletion, Insertion

* Traversal:-
  - How to visit all Node of The BST in specific order

Traversal → in-order, Pre-order, Post-order
The Three specifica orders To Travese The BST

* → consider The following BSTs.



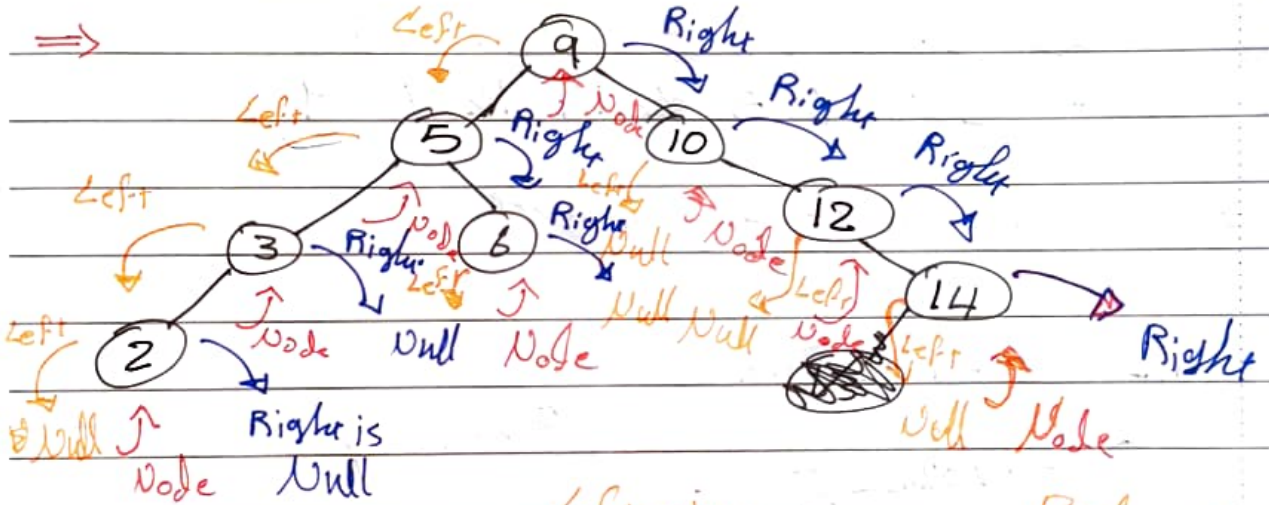
Subject Date No.

→ in-order Traverse :-

↳ LNR (Left - Node - Right)

* Visit the Left Node
* Then The ~~Root~~ or the Parent of the Node
* Then The Right Node of the Parent
* Apply this ~~Each~~ order on The Best of the Nodes until you reach the leaf

⇒



Left-subtree Right-subtree

⇒ In-order :- 2 3 5 6 9 10 12 14

↳ Based on the order That is used to Traverse The BST, We ~~can~~ can know where is The Root In-order The Left_subtree ~~is Left~~ is left the 9 and Right_subtree is Right 9

* The in-order on BST Gives me sorted elements in asending order (~~descending~~)
* The in-order IS Not a sorting Algorithms
* it Just an feature

NOTEBOOK

Subject Date No.

→ Pre-order Traverse :-

↳ NLR (Node - Left - Right)

PRE-order :- 9 5 3 2 6 10 12 14

↳ * Pre-order ⇒ The Root is on The first

→ Post-order :-

↳ LRN (Left - Right - Node)

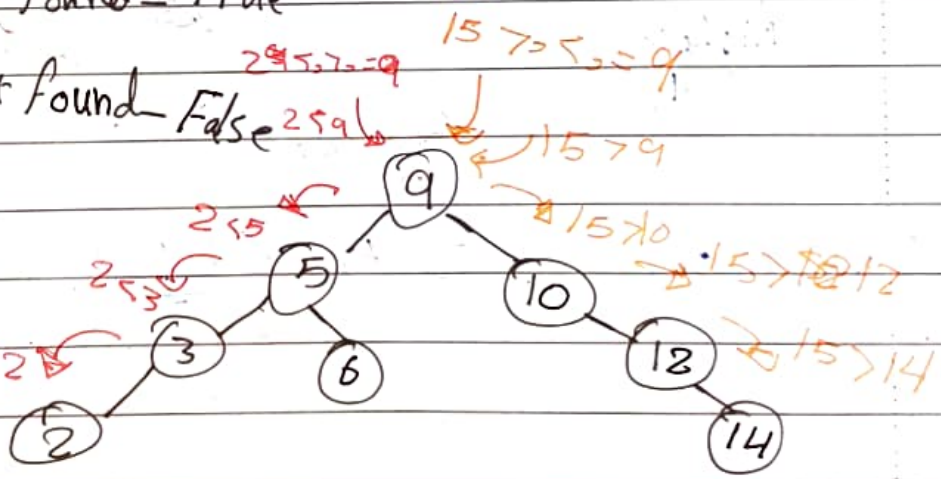
Post-order :- 2 3 6 5 14 12 10 9

↳ * The Root is The Last in order

* Search → Found - True

→ Not found - False

Serch for 2 and 15



founde it *

NOTEBOOK

Subject Date No.

*Deletion:

*Insertion:

↳ To insert a value in The BST
- serch the BST to find the Proper position
- insers the value

* if the value ~~is existe~~ existes Then No insertion will be completed, Why? Duplicated value

*Deletion:

Probabilites
* There Three choices to delete a node that have Either

① One child
② Two child
③ No child

① No child:
Just delete the Node, Because the deletion won't lead to Missing data

Subject Date No.

② One child :- → Left - child
→ Right - child

Ok, Lets devide The BST into → Left-subtree
→ Right-subtree

* We Need to delete a Node That has one child only

~~* If The Node on in The Left subtree for Left the Parent~~

* if The Node in The Left-subtree And it has one child → Left child
→ Right child

→ Let child Node be in The Left side of the Root of The deleted Node
→ Either is Left or ~~child~~ Right Node

* if The Node in The Right_subtree And it has one child → Left
→ Right

→ ~~Let child Node be in the Right side of the Root of The deleted Node~~
→ Let The child Node be in The Correct side by comparing it with The Parent of The deleted Node

NOTEBOOK

Subject Date No.

③ Two childs:

To

⟹ To Get The:

* Max ⟹ Go left until you get Null
* Min ⟹ Go Right until you get Null

*How To Delete The Node #who has Two childs:

① starting from a Node

② will Go Either:
- left ⟹ Max Element in The left of The Node.
- Right ⟹ choose The Min Element in Right of The Node

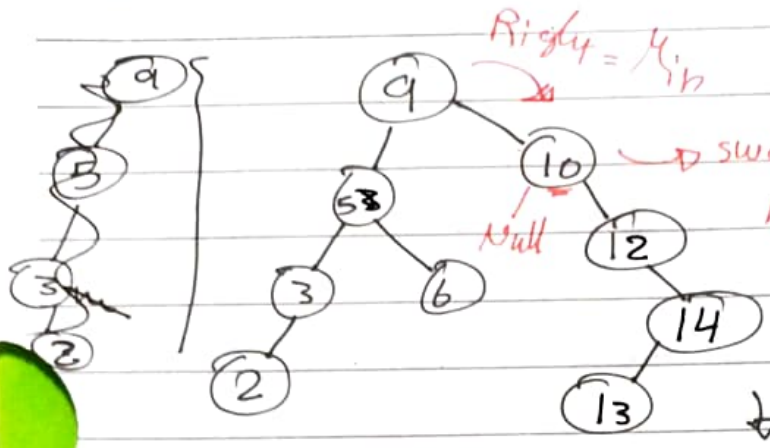
③ After ~~choosing~~ choosing The ~~Node~~ Max or Min Node we Need to swap or shift it ~~with~~ with The deleted Node

⟹ why? To convert The Node from Having Two child to ~~Node~~ ~~childs~~ One child

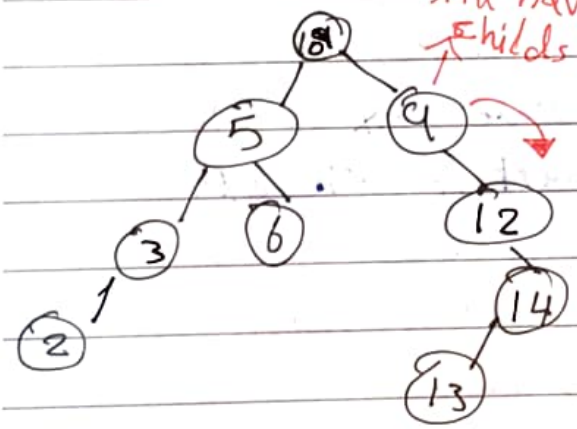
NOTEBOOK

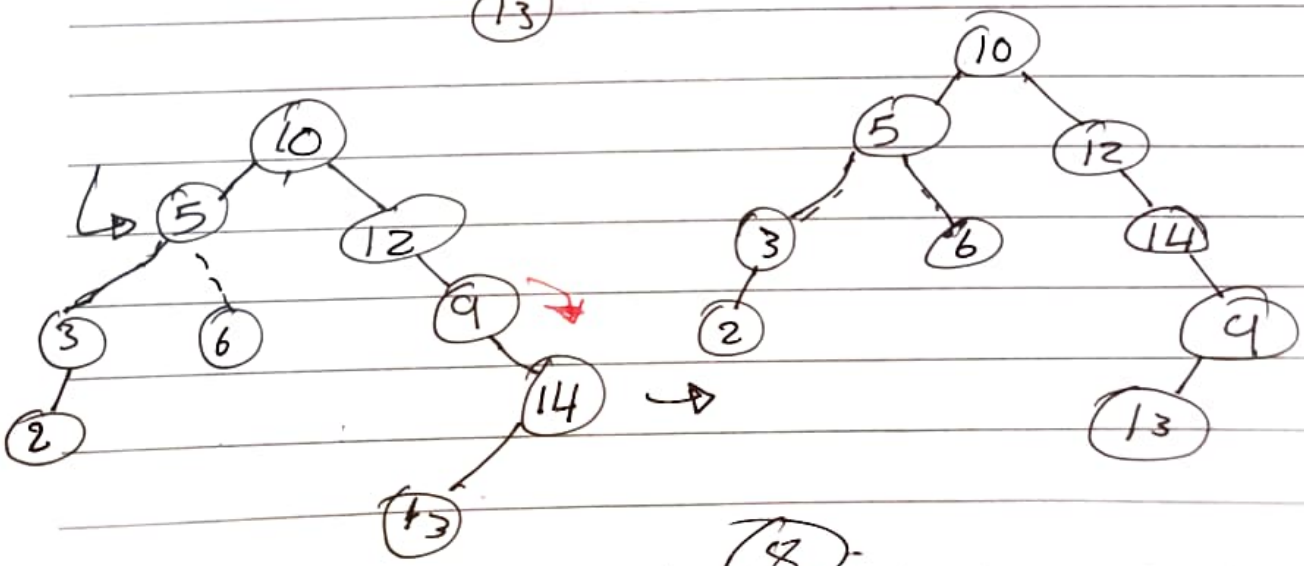
Subject Date No.

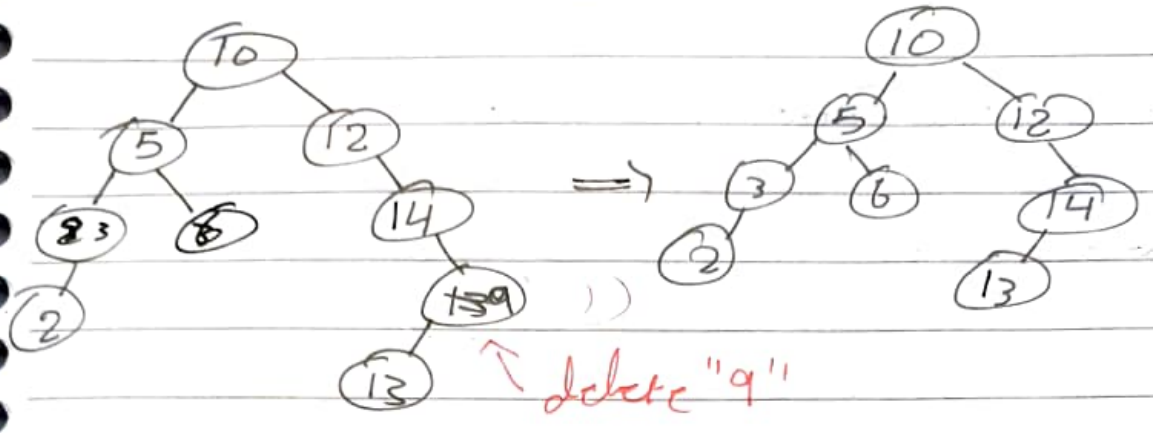
* Ex: considering The Prev BST, delete The Root "9"



If we ~~choose~~ choose The Right subtree Then we have to choose The Min Element → How?
↳ By Going to left of The subtree and so on.







✲ How to work in The Exam: And some Notes:-

① Traversal

↳ The cost for Traversing The BST is Payed Just once, The cost is Not Duplicated.

⇒ Proposal Question on Traversed

↳ What is The Root, what is Level, Path?

⇒ ~~There won't be~~

→ There is No question such as "Make Traversal and Give The output?"

→ The output of a traversed is given to You in The Exam, and you will use it to solve The question.

⇒ The cost of The Traversal is O(N)

NOTEBOOK

Subject Date No.

(2) search :

↳ simple Questions, ~~True~~ True or false ...

* No Question such as, find the value on The Given Tree.

(3) insert ⇒ [illegible]

(4) Delete ⇒ [illegible]

↳ Proposal Question :

↳ If we delete Node "5" The New parent for the Node "3" is 4, True or false And why?

Note :

O(log(N)) → search, insert, delete

if we start from The Root or ~~For~~ Not for The left or Right subtree

O(N) → Traversal

* And Not Duplicated

* if you Given a Binary search array and to convert it To BST what is The Root?

~~The~~ The is a[0] Not The Mid value

NOTEBOOK

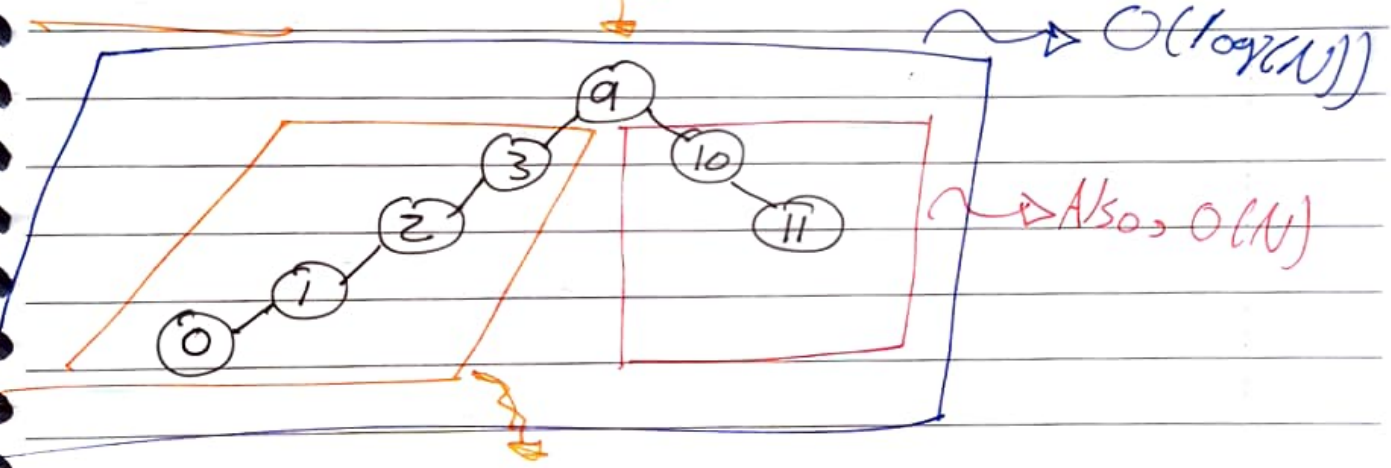
Subject Date No

⟹ Why, search, insert, and delete is $O(\log(n))$?

Because we decreasing The Number of The operations by Two ⟹ How? Greater Than OR Less Than

↳ The complixity for searching, inserting, deleting subTrees is $O(N)$ ⟹ Why?

↳ The oPposite of the Prev Answer



$O(\log(N))$

Also, $O(N)$

$O(N)$; No decreasing on The Number of operations

Subject Algorithms (12) محاضرة ١١ Date No

جشع / شره

* Greedy Algorithm:

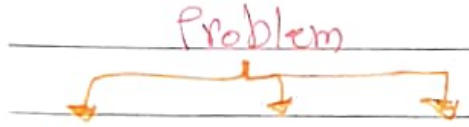
The Greedy Algorithm or The Greedy Techenique
↳ is a Technique for solving Problems of Type
Optimization Problems
↳ what is the optimization Problem?
↳ The optimization Problem is finding The
Optimal solution And
The optimal solution differs (يختلف)
from one case to Another
↳ Not the Problem, but why?
Bec, in The optimization
Problems we find
The Best of
The optimal solution
from a Group of Proposal
solutions
* if we change The
case The Group
of Proposal Proposal
solutions will
be different

* Note: The case
is described
by the specifications
of The Problem

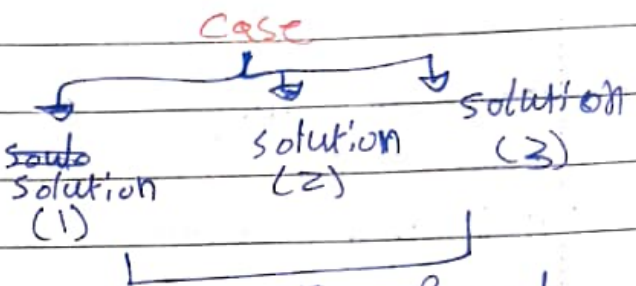
NOTEBOOK

Subject Date No.

* Problem (e.g.. sorting)



* Optimization Problem



finding The optimal solution Among.

These is The Greedy Algorithm OR Greedy Technique

Ok, we Need to find The optimal solution, But How?

↳ By choosing The least cost solution

→ But The Least cost of The solution is Based on The case of The Problem

→ So, Who determins- determins The cost? You, or The Question

* Note:.. The cost here Not The Time in The Big-O it was The Time but Here is dependent on The case

Subject Date No.

* How to determine The Least Cost solution?

↳ By determining The Greed choice

↳ Greedy choice ← Max / Min] The cost will be Paid

~~To simplify for you This~~

* will Take Three Problems solved by The Greedy Algorithm

1. Huffman Code
2. Counting Money
3. Process schedular

* Counting Money :

The Problem Statment :- We Have a banknote ( ) And we Need to exchange it ( ) To smaller banknotes with least Number of banknotes.

( ) ~~[illegible]~~

* So, The Best or Otimal sdution is to find The least Number of Bank Notes

Subject Date No.

* How?
  - By choicing choosing The Greedy choice
    - The Greedy choice is The Max
      - why? We Need To find the $ least Number of bank Notes by finding the Max Bank Notes

50 JD

20	10	20	5	1	0.5
20	10	10	5	1	0.5
10	10	10	5	1	
	10	10		1	
	10				

These Are Proposal solution we Need To find The optimal One

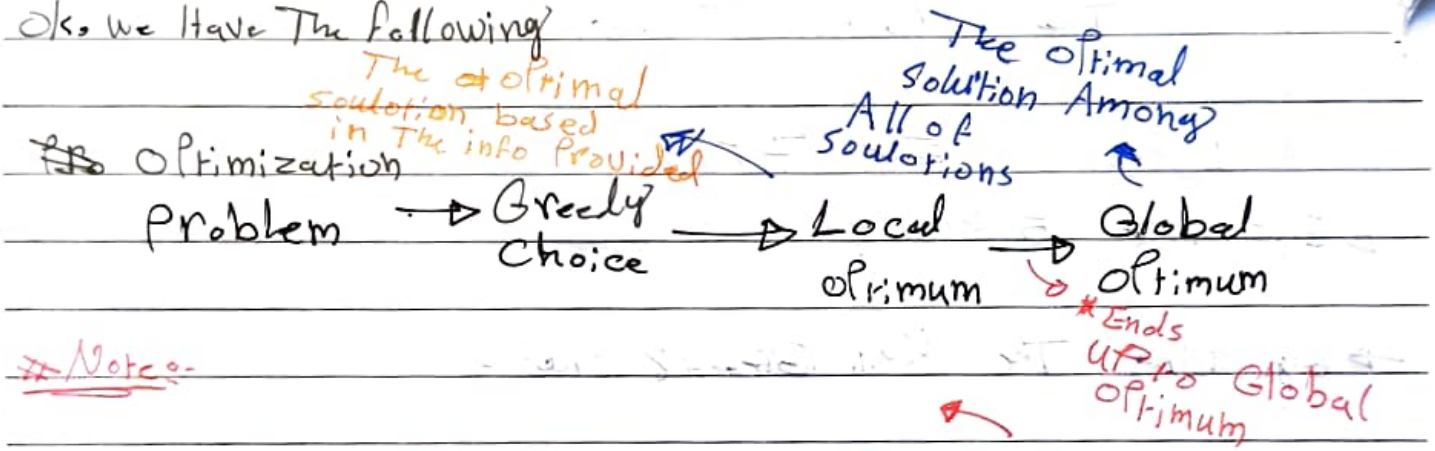
* choose The Max
  - 50 - 20 = 30 → local optimum
  - 30 - 20 = 10 → local optimum
  - 10 - 20 → can Not use The 20
  - 10 - 10 → 0 → solution founded
    - local optimum

NOTEBOOK

Subject Date No

* in our our case we Needed to find The least banknotes
↳ So we Have choosed The "Max" Greedy choice
* in other case we May to find The Maximum Number of coins (Like bus driver as Example)
↳ So, The optimal solution will be The Maximum Number of coins And The Greedy choice will be "Min"

Ok, we Have The following



* Note:-

→ The Greedy Algorithm Guarantee The local optimum

→ But it Dose Not Guarantee The Globale Optimum

→ So we find The Local find so it May be the Global optimum

Again we Have a Group of Proposed solution
Just one of Them is The optimum Global Optimum
→ By using The Greedy choice we: * MAY Get the Global optimum

Subject Date No

↳ Greedy Choice → Local Optimum → The Local is Global → Algorithm success

The Local is NOT Global → Algo failed

Ex:- By considering The kron BankNote with The following catagories

1 kron ~~5 k~~ 7 kron 10 kron

→ Exchange 15 kron ?

↳ By using The Max Greedy choice

15 - 10 = 5

5 - 7 → can Not be used

5 - 1 = 4

4 - 1 = 3

3 - 1 = 2

2 - 1 = 1

1 - 1 = 0

The Greedy Algorithm solution

↳ But, The Algo fails ⇒ Why?

↳ There is an better solution which is The Global optimum

This is a LOCAL Optimum

Subject Date No.

↳ Global optimum is:-

15 - 7 = 8
8 - 7 = 1
1 - 1 = 0

} global optimum → This solution is NOT Given By The Algorithm

Problem

* ~~scheduling process~~:

→ Problem Statment: we need schudle the running of The Processes in The ~~os~~ by The OS.

↳ we Have to define The case to find The optimal solution for That case

Every Time we change The case definition The Algo will be changed

* The case:

→ Does The Process starts and Ends in specific Time? ~~NO~~ No, if Yes

→ ~~Does~~ Is The Process is Primitive or Non Primitive?

} you Have to work with ~~other~~ other Algorithm

→ Does The Process Runs on ~~on~~ stricit Resources? No

→ for What we Need The scheduling? To ~~decreasing~~ increase The Throughput

NOTEBOOK

Subject الأنشطة Date No.

↑

Q: What is The Throughput?

The Throughput is running All Processes in The Least time

Ex: we have Three Algorithm to solve The case defined → ~~Reg~~ Longest
→ shortest
Better

① Longest

		Time		Total	
P_1/R_1	20	10	3	33	The Throughput is 35
P_2/R_2	18	11	6	35	
P_3/R_3	15	14	5	34	

② shortest

				Total	
P_1/R_1	3	10	15	28	Throughput is 40
P_2/R_2	5	11	18	34	
P_3/R_3	6	14	20	40	

NOTEBOOK

Subject Date No.

③ Better

				Total
P_1/R_1	20	14		34
P_2/R_2	18	11	5	34
P_3/R_3	15	10	6	34

Throughput = 34

⇒ Note :

shortest + longest ⇒ fails To find The Optimal solution

⇒ why?

Because There Are An Algorithm That finds The optimal solution which is The Better Algorithm.

Better ⇒ sucess ⇒ why? finds The optimal solution.

NOTEBOOK

Subject Date No.

⑤ Huffman code :-

The Main idea behind the Huffmancode is To Compress Data

* consider The following Example :-

•docx file

a a b a
b c a c
b a d a
c a c

→ This file contains 15 letters
→ Each letter is Represented by using 8 bit
→ 15 * 8 bit = 120 bit
→ fixed - lenght & Each letter represented by fixed lenght which is 8 bit.
→ How Many letters we have in The langauge?
↳ 4 letters

Not The ENG

* The Huffman code Treats The langauge Based on The letters exisits in the file.

→ The frequency of letters
a = 7, b = 3, c = 4, d = 1

Subject Date No.

→ The fixed lenght for Eeach letter is :-

a = 7 * 8 = 56

b = 3 * 8 = 24

c = 4 * 8 = 32

d = 1 * 8 = 8

* Remember That we Need To compress data

↳ Why? To save storage

→ Compressing data To save Storage ~~can~~ could be done By converting The Representation of the ~~Sinal~~ Singhe letter from fixed lenght → To Variable lenght code

→ The Variable lenght code :- The letter is Represented by one bit, Two Bit, Three Bit, etc

~~Intead~~

* Intede of 8bit for each Letter.

↳ a = 0
b = 1
c = 10
d = 11

* consider This code :-

0 0 1 1

a a b b

OR

a a d

NOTEBOOK

Subject Date No.

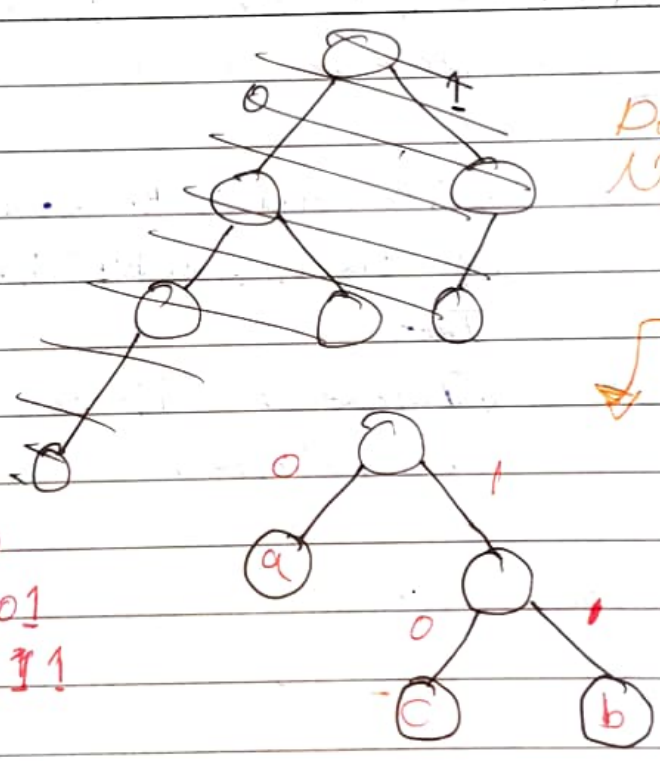
→ converting The code from fixed lenght To variable lenght ~~Reduce~~ produces a code That could Have more Than one Meaning

↑ This is NOT correct

→ How to Have a code with only one Meaning?

↳ By using The Binary Tree

Ex :



Details in The Next lecture

a = 0

c = 01

b = 11

* How Many Bits That I have been ~~Sa~~ saved for a? 56 - 7 = 49 bit saved

a = 1 * 7 = 7

it was a = 8 * 7 = 56

Subject Date

→ But, where is The use of The Greedy Algorithm?

↳ We use the Greedy to find the Min Number bits to assign it to the Most Frequent Letter.

→ There is a chance where a single letter Have Two codes but The same lenght → The Both is correct

⇒ How? By using different Algorithm

↳ So, The following code is used To Have The correct result in The Exam.

```
Huffman(int n, char C[1...n])
{
    Q = C; // Priority Queue
    for (i = 1 to n-1)
    {
        z = new node;
        z.Left = X = Q.extend.min();
        z.Right = y = Q.extend.min();
        z.Prob = x.Prob + y.Prob
        Q.insert(z);
    }
}
```

Subject Algorithms (13) جزء 1 ✱ Date No.

✱ Heap Sort :-

↳ what is Heap?

↳ The Heap Memory - Allocating Memory Dynamically

→ Allocating Memory Dynamically?

↳ Allocates Memory as Needed

→ unneeded Memory is released

↳ Heap + sort ⇒ Sorting items using Heap Memory

⇒ Recall The Previous Sorting Algorithms..

✱ selection - insertion Bubble sort
- No Extra DSA

✱ Merge - Quick sort
- less cost (Time)

Take The Advantage

Take The Advantage

Heap sort
- less cost
- No Extra DSA needed

⇒ The H

NOTEBOOK

Subject Date No.

Ok, we Need to use Heap Memory To use it in Sorting
↳ How to Manage The Heap Memory?
↳ By using Binary Tree.
↳ Each Node Represents a Dynamic Memory.

* The BT That is used in Heap sort :-

* Complete BT :-
→ The complete BT is a BT where Each We don't Move from one level to Another until we fill The Max Number of Nodes in That Level
→ Ex: The level 2 take $2^2 = 4$ Node a Node Don't inserted in level 3 until we fill The level 2 with 4 Nodes

* Left Justify BT :-
→ Start filling The BT from Left to Right ONLY.

* Why BT in Heap Sort?
↳ To Manage Dynamic Memory
* Why complete BT in Heap sort?
↳ To Avoid Missing Data
* Why complete left Justify BT in Heap?

Subject Date No

↳ To guarantee 100% That Each Node has it's own data; for Avoiding Missing Data.

---

* How To Build a complete left Justify BT?

↳ consider The following Data:-

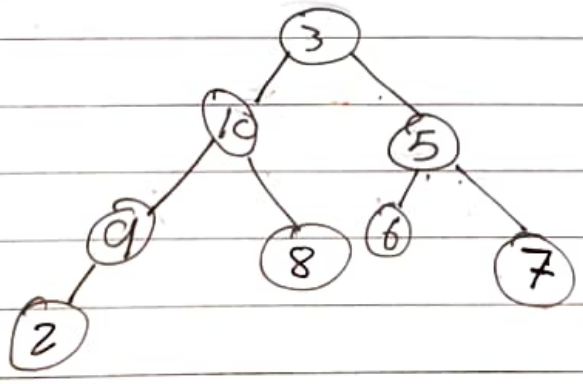
0	1	2	3	4	5	6	7
3	10	5	9	8	6	7	2

→ The Root is a[0]

→ For each Node we find The left and Right child By using The following formulaes:-

→ Left = 2A + 1 } for zero based index array
Right = 2A + 2 } (A = index)

→ Left = 2A } for one based index arrays
Right = 2A + 1 }



NOTEBOOK

Subject Date No.

⇒ How To Sort?

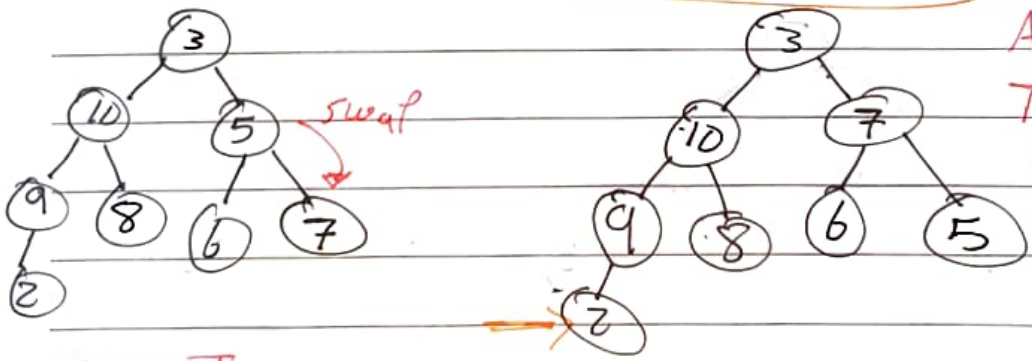
we sort by using
- Max Heap
  - Ascending
  - deascending
- Min Heap
  - Ascending
  - Deascending

↳ Max Heap: The Node must be bigger Than it is owh childs

→ To select select The Max Heap we start from bottom to Top or from The last level to The first one

↳ why?

↳ Bec. We Need The Max child Between The Two child and The childs of a Node is in one level down From it.



9 > 2 = T

10 > 9 , 10 > 8 = T

~~6 > 5 = F~~

5 > 6 = F, 5 > 7 = F

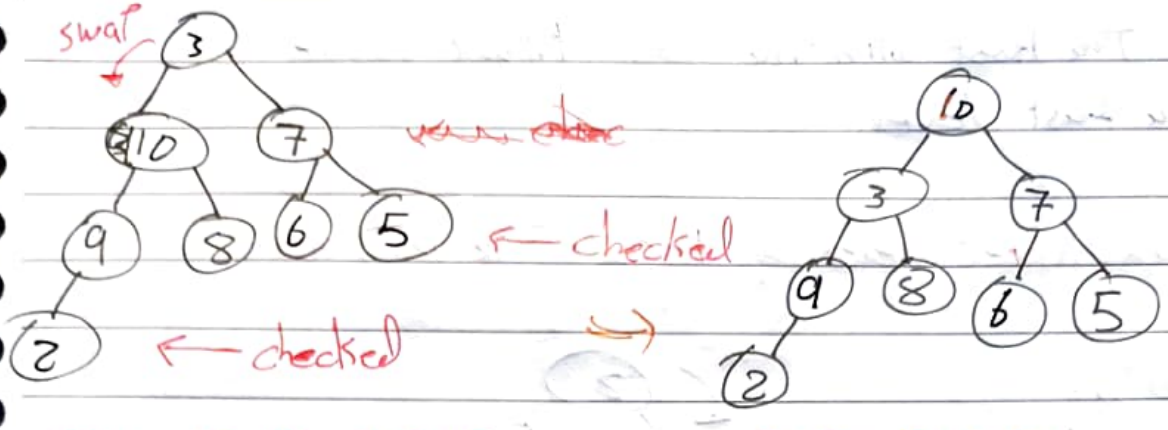
Max(6,7) = 7

swap(5,7)

After swaping ** Two Nodes We check Again if The Node is Max Between it's childs

in swap(5,7) we don't have to check

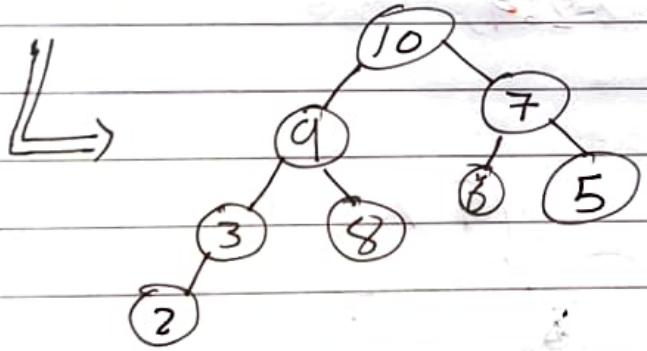
NOTEBOOK



3 > 10 = F, 7 > 3 > 7 = F
Max(7, 10)
swap(10, 3)

3 > 9 = F, 3 > 8
Max(9, 8)
swap(9, 3)

⇒ The swap was Applied on The left subtree and "3" has Two childs ⇒ Recheck



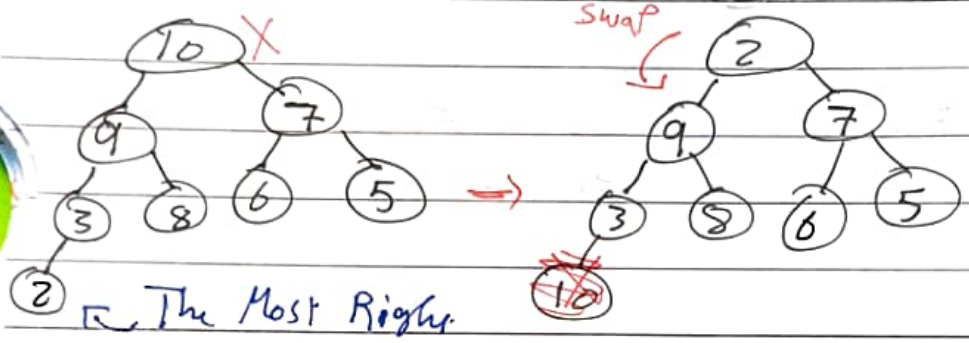
→ Now, We Need To Emptying (i.e.) The Tree into array
→ starting from The ~~Node~~ Root
Put → The Root in → index 0
→ index n-1

→ Then Delete The Root Node ⇒ why? To save Memory
→ How To Delete?

Subject Date No.

↳ swap The Root with the Most Right Node in The Last level.

→ Then, After deleting Resort The BT. insi-



↳ The Most Right

~~swap~~ swap(10,2)

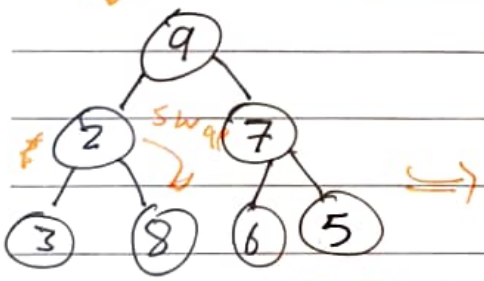
delete(10)

2>9=F, 2>7=F

Max(9,7)

swap(2,9)

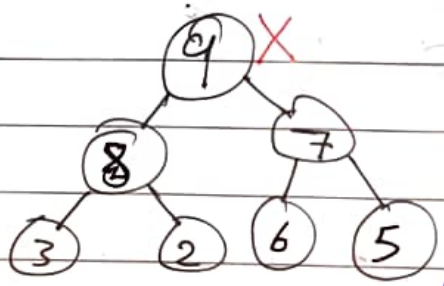
After swap check



2>3=F, 2>8=F

Max(3,8)

swap(2,8)



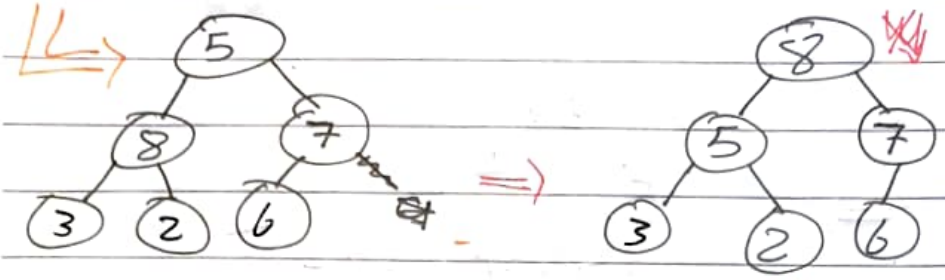
↳ The Most Right

swap(9,5)

↑ checks After swap

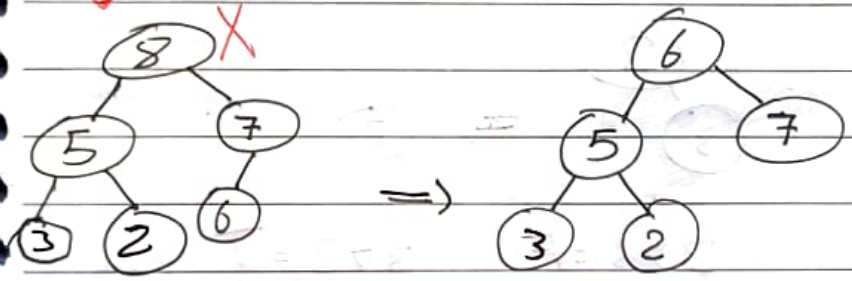
NOTEBOOK

Subject Date No.



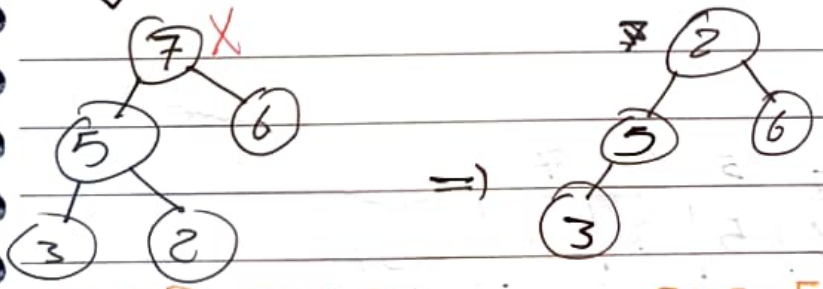
5>8=F, 5>7=F
Max(8,7)
swap(5,8)

5>3=T, 5>2=T
No swap



swap(8,6)

6>5=T, 6>7=F
swap(6,7)

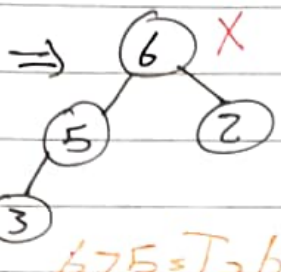


7>5=T, 7>6=T
No swap
~~delete(7)~~
swap(7,2)
delete(7)

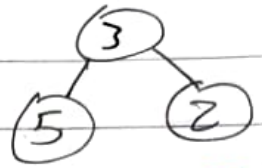
2>5=F, 2>6=F
Max(5,6)
swap(2,6)

7

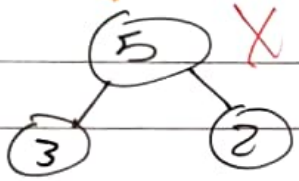
NOTEBOOK



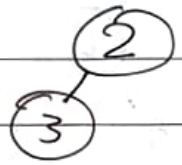
6>5=T, 6>2=T
delete(6)
swap(6,3)
delete(6)



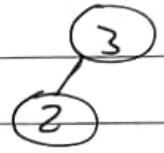
3>5=F, 3>2=T
swap(3,5)



5>3=T, 5>2=T
swap(5,2)
delete(5)



2>3=F
swap(2,3)



3>2=T
swap(3,2)
delete(3)

(2)

Delete(2)

0	1	2	3	4	5	6	7
10	9	8	7	6	5	3	2

NOTEBOOK

Subject Date No.

* The cost for Building BT?
  ↳ O(N) => why? Iterate over All Elements

* The cost for sorting BT?
  ↳ O(log(N)) => why? The Max child Either in The left or The Right.

The Total cost is ① O(N log(N))

* In The Exam?

→ * complete Tracing for Heap sort After 3 Phases?
→ why BT, complete BT, left Justify BT?

The Best case = worst case = O(N log(N))

← بكتب جنب سؤال بيقول إن Heap sort وبحكي هو الـ Big-O، هو الـ worst case لـ Heap sort إذا استخدم الـ

← هل يوجد خوارزمية لها Best case أفضل من worst case توجد الـ Big-O هو نفسه الـ O، Ω، Θ

NOTEBOOK

Subject :- Algorithms (14) Huffman ☆ Date No

☆ Huffman Code :-

↳ Why Huffman code? → To compress data by Reducing The # Number of bits

↳ How?

↳ By converting The Representation from fixed lenght to varaible lenght code

↳ How?

↳ ~~By using BT~~

By using The codeword

↳ How The codeword is implemented?

↳ By using BT

↳ which kind of BT?

↳ General BT.

↳ Why BT?

↳ The code of Each letter is Represented Either By "0" or "1" so, This is The Job of BT

NOTEBOOK

Subject Date No.

✱ How The BT For The Huffman code is Build?

↳ The BT for HuffmanCode is Build from The Leafs to Root

↳ why?

↳ To Have The Most frequent value in The Root or close to The Root

↳ why?

↳ To Have less depth or Height And so to Have less Number of Bits → To save Memory.

Example : what is The Huffmancode for The "e" letter?

التكرار

a : 5 5/100 0.05

b : 9 9/100 0.09

c : 13 0.13

d : 16 0.16

e 12 0.12

f : 45 0.45

100 - 88 = 12

→ How to find the Huffmancode for 'e'?

↳ By computing The frequency of "e"

↳ How?

Total - sum = x

إذا حددو عدد البتات في السؤال

إذا لا بتات

100 = Total

NOTEBOOK

Subject

Date

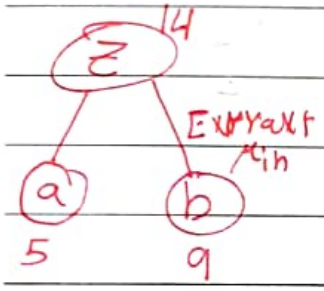
No.

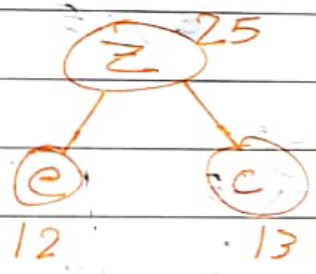
↳ Take These Probabilities or Frequencies (التكرار)

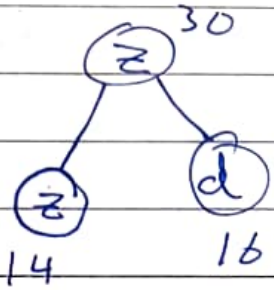
→ Put Them into Priority Queue

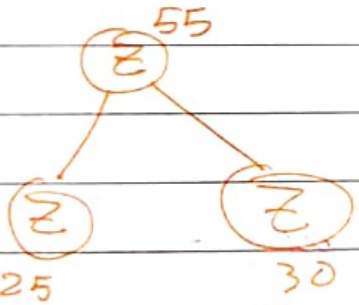
↳ from Min ⇄ To Max.

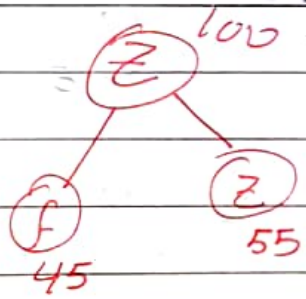
5	9	12	13	14	16	25	30	45	55
a	b	e	c	z	d	z	z	f	z
X	X	X	X	X	X	X	X	X	X











→ Create New Node → Extract Min from Priority Queue for Left and Right child.

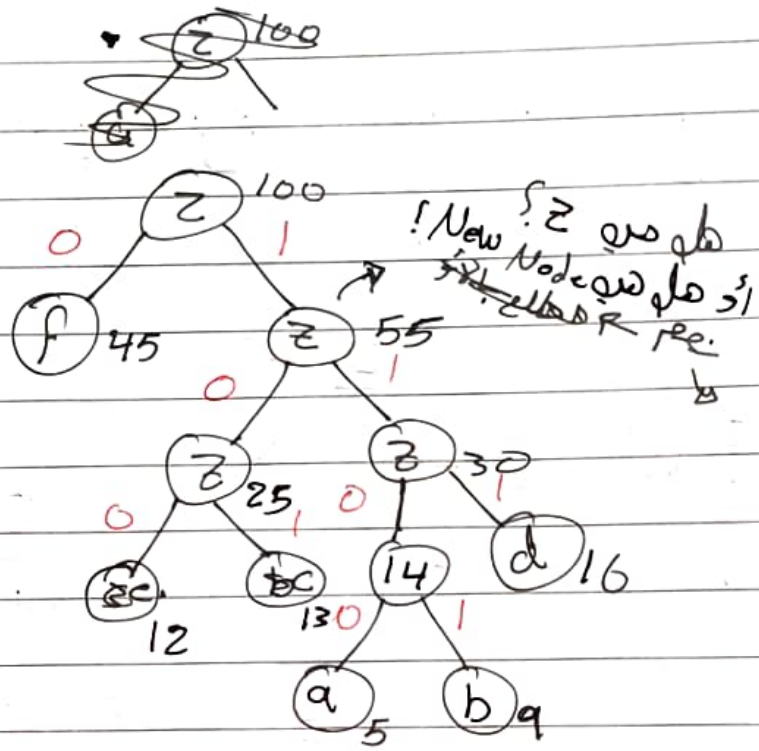
→ Take The sum of The child Node

→ insert The New Node into The Queue

→ Repeat until you will have 100.

Subject Date No.

* Now Build The BT - from starting from Root (which is biggest value)



** Notes :-

→ "z" is New Node Not a letter

→ Because The "z" it is a Node That we Have been created it should Have childs → Put Them

→ To get The code → Left side coded "0"

Right side coded "1"

other works in The Exam is Not Correct

Subject Date No

The code for Each Letter:-

f = 0

e = 100

c = 101

d = 111

a = 1100

b = 1101

* How To compute The saving Bits :-

Total Number of bits before Huffmancd / 8 * Frequency) − Total Number of bits After Huffman (N * Frequency)

↳ The Number of bit in Huffman code

f = 0 ⇒ one bit

(8 * 45) − (1 * 45) = 315 bit saved

e = 100 ⇒ Three bits

(8 * 12) − (3 * 12) = 60 bit saved

Subject Algorithms (15) المحاضرة # Date No.

* Graph:

→ What is Graph?

↳ The Graph cosists of Two set; set of Nodes and set of Edges

→ Types of Graph?

(1) (a) —— (b)

↳ ~~Hub~~ Number of Nodes? = 2

Number of edges? 1

→ This Graph is an undirected one

↳ Undirected Graph:- The Graph is undirected whe "a" is reachable from "b" and "b" is reachable from "a"

(2) (a) → (b)

↳ Directed Graph:-

Is "a" reachable from "b"?

↳ No

Is "b" reachable from "a"?

↳ Yes

Subject Date No.

(3) (a) —3→ (b)

↳ Directed ~~un~~weighted Graph

↳ what is The Meaning of The weight?

↳ The weight is cost That is Paid when Going from "a" to "b".

↳ what is The cost?

↓

~~The Graph~~

The Graph could be devided into Two Types ~~for~~ from a point of View of the weight

→ what ~~is~~ cost refreseuts?

↳ Time? No, Wrong X,

To know what is cost you have to know what the Graph refreseuts, and The Graph could be → General Graph

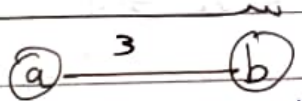
→ Application refreseuted by using Graph

NOTEBOOK

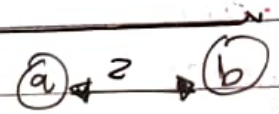
Subject Date No.

a Number in Math

→ General Graph
  └ The cost weight represents an Abstract Num
  → And The Graph represents Nothing.
→ Application Graph
  └ The Graph represents a case or real life Application, Such as:- Network, Map of cities, etc.
  → The cost Then represents some Value
  ⇒ for Example ⟹ The weight in The Network Graph represents The Bandwidth
  ⟹ in The Map The cost That is Paid between Two cities is it could be → Time, → Distance, → Gasoline

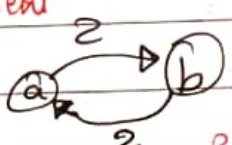
(4) 

└ Weighted Undirected Graph

(5) 

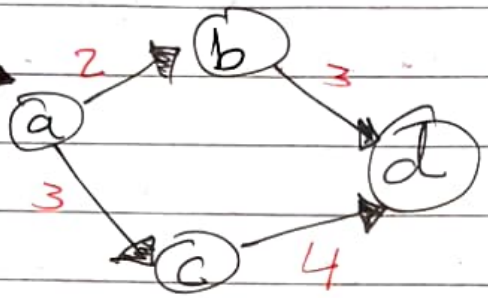
└ Bidirictional Graph
→ In This subject
→ in our course There is nothing such as bidirectional Graph

These Two are different
→ How?
└ Review Lect(9) Page(6)



Subject Date No

→ Consider The following Graph:



This Arrow Represents The starting Point of The Graph.

→ starting from "a", where will go Next?
↳ To the Min cost (weight); and The Min cost is the "2"
↳ why 2 is Min cost?
↳ Bec this is a general Graph; and if it is so, the weight is represented by using Abstract Numbers; and by using Abstract Numbers 2 is less Than 3

→ If the Prev Graph represents a Network, what is The Min cost?
↳ first Thing what cost Represents? Represents the bandwidth

Subject Date No

And The Bandwidth is The # Number of bits that is Transfered in same time

↳ if want to transfere a file 12 bit

$\frac{12}{3} = 4$ Packets

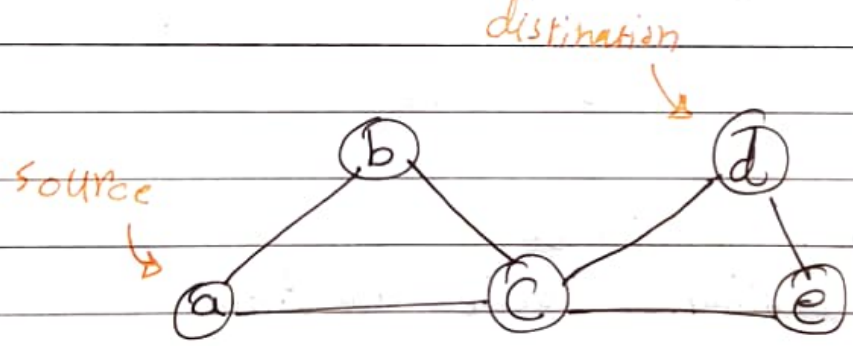
$\frac{12}{2} = 6$ Packets

So, The Number of Packets by using "3" is less Than using "2"

→ So, 3 is Min Cost Than 2

* Path :-

→ consider The Following Graph;



Path :- is a sequence of Edges leading from source Node to distination Node.

↳ Give a Path from "a" to "d"

1. * a - b - c - d
2. * a - c - d
3. * a - c - e - d
4. * a - b - c - e - d
5. * d - c - b - c - d

(5)

Subject Date No.

→ Question: Can we find The Max Number of Paths from one Node to another?
↳ look for The Answer

→ Question: what is the different between Path (1) and Path (5)?

Path (1) —called→ simple Path
↳ Visiting the Each Node only once

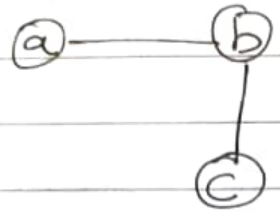
Path (5) —called→ Path
↳ A Node is Visited More than one.

→ when the Path is Mentioned
↳ source and distination Nodes are Also Mentioned
↳ source and distination Node we Need to find The shortest Path Between Them
↳ To find The shortest Path by Using Graph Algorithms (shortest path → BFS)

Subject Date No.

* connected, disconnected, Graphs:

(1)

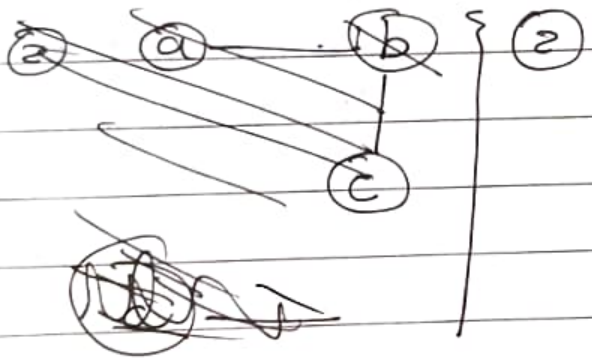


→ Undirected, Unweighted Connected Graph

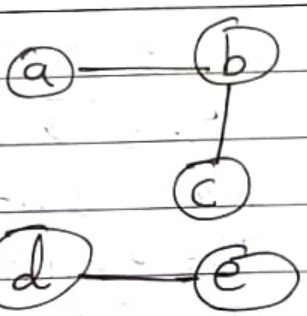
↳ in The undirected Graph There is No Problem with Paths from one Node to Another if the Graph is connected

There is a Path from Each Node to Another

(2)



(2)



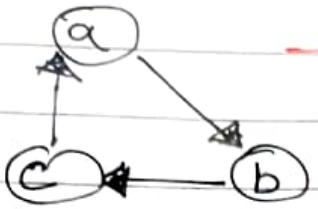
↳ Disconnected, Undirected Graph

↳ why?

↳ There is No Path from "a" to "d" or "e" to "c"

⋮ etc

Subject Date No.

(3) 

→ Directed ..... Then ?!!
↳ if the Graph is directed the Graph is Either

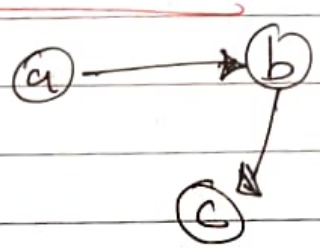
Weakly Connected

→ There is at least one Node is Not Reachable from other Nodes in The Graph

Strongly Connected

- There is a Path from a Node to another Nodes in The Graph

→ So, The Graph Above is Directed, Strongly connected Graph.

(4) 

→ Directed, weakly connected

NOTEBOOK

Subject Date No

→ We said That the Graph consists of Two sets → Nodes
→ Edges

↳ Question:- what is the relation between These Two sets.

→ To find The Answer back to discrete Math.

Subject Algorithms (16) محاضرة ١١ Date

*Special Graphs:

1- Complete Graph.
2- Cycle Graph.
3- Wheel Graph.
4- Bipartite Graph.

* How To ~~calculate~~ calculat the Number of Edges in a graph?

→ we have Two types of Graphs: Directed, Undirected

↳ UnDirected:
↳ for Each Node there is a degree where the degree is the Number of Edges connected with a Node.

↳ Directed:
↳ for Each Node there is: in-degree, out-degree

* # of Edges in a Node = in-degree = out-degree

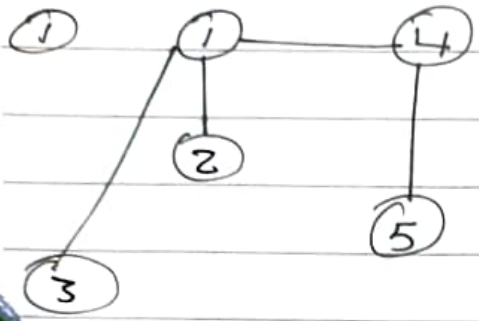
* Edges out from a Node

→ where: the Number of Edges = sum(degrees) / 2

Why?
↳ Each edge connect Two Node

Subject Date No.

*Example:



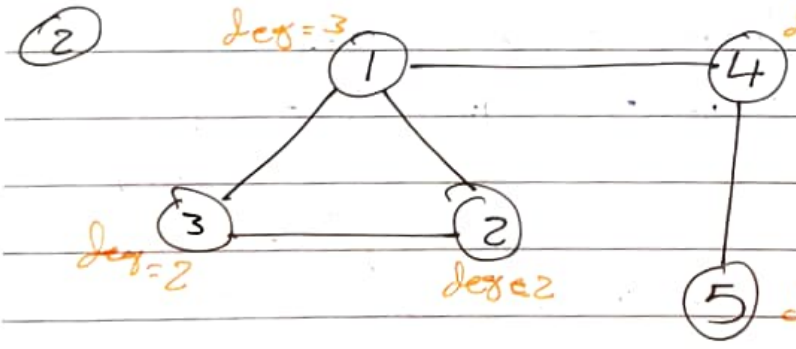
degree = Vertix Number

→ deg(V) = # of edges connected with V

↳ deg(1) = 3, deg(2) = 1
deg(4) = 2, deg(3) = 1
deg(5) = 1

→ sum(deg) = 3 + 1 + 2 + 1 + 1
= 8

# of edges = sum(deg) / 2
= 8/2 = 4



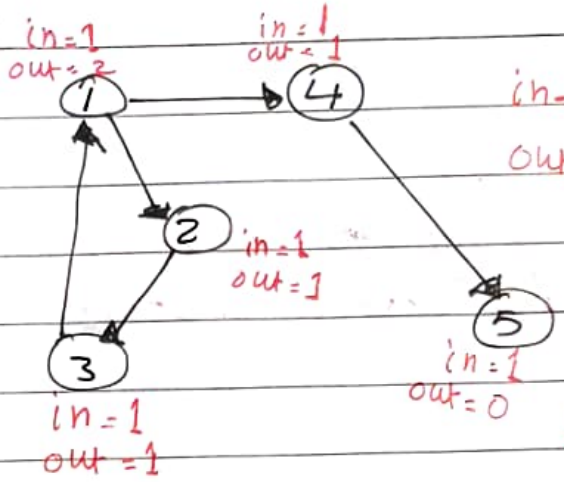
→ sum(deg)
= 3+2+2+2+1 = 10
→ # of Edges = 10/2 = 5

*Note: In The Graph Number (2) we added an Edge between "3" and "2"
↳ So, we can Notice That: Every Time we add an Edge on The Graph The sum(deg) will increase by 2

NOTEBOOK

Subject: Date: No.

(3)



in-deg(1) = 1

out-deg(1) = 2

→ sum(deg) = sum(in) + sum(out)

↳ sum(in) = 1+1+1+1+1

= 5

→ sum(out) = 2+1+1+1+0

= 5

→ sum(deg) = 5+5 = 10

→ # of edges = $\frac{10}{2} = 5$

* Note: The sum(in) Alwayse equal to the sum(out) → why?

↳ Because The Edge connectes Two Node, So the edge will be out from one Node and in to another one

Subject Date No.

→ Special Graph:

1 complete Graph:

↳ The Graph is complete if only between any Two Nodes: There is an Edge

Examples:

① ① K_1 ⟹ K_n → n = number of Nodes

↳ complete; How?

↳ There is only one Node and there is No Need to Have an Edge

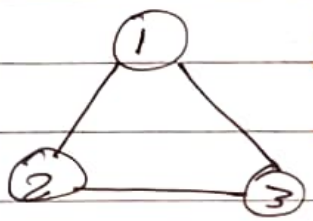
→ degree = 0

② ① —— ② K_2

↳ complete; between Any Two Node there is an Edge

→ degree = 1

③

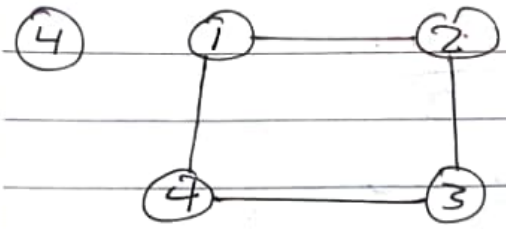


K_3

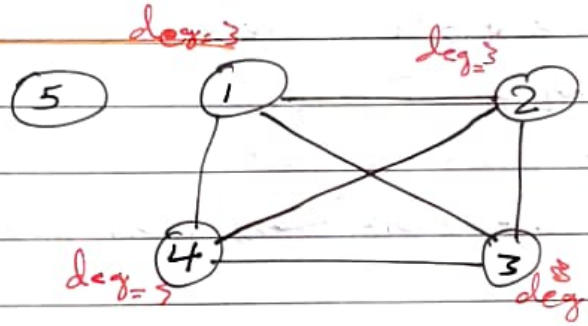
→ complete

→ deg = 2

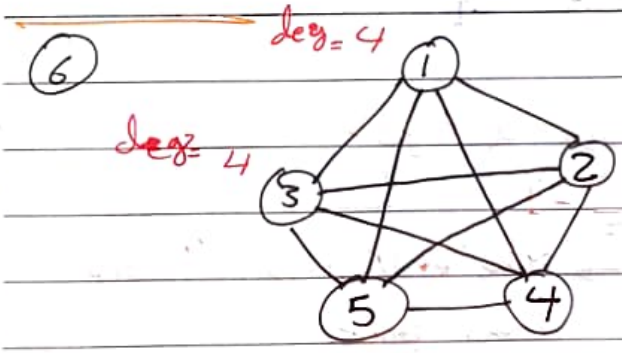
Subject Date No



X
↳ Not complete; There is
No Edge between
"1" + "3"
"2" + "4"



K4
↳ complete
→ degree = 3



K5
↳ complete
→ degree = 4
of each
Node

~~* We can Notice that~~

* Note: We can Notice The following:

→ The degree of each Node = Number of Nodes - 1
(n-1)

→ The summation of The degrees = n*(n-1)
↳ for Example → 5*(4) = 20 degree in The whole Graph

→ # of Edges = $\frac{\text{sum(deg)}}{2} = \frac{\sum n*(n-1)}{2}$

$= \frac{20}{2} = 10$

[2] cycle Graph: C_n

↳ The cycle Graph is a Graph where ~~you can~~ The starting and The End Node is The same Node without ~~visiting~~ Visiting a Node More ~~Than~~ Than once.

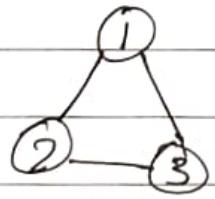
① ① → X Not a cycle C_1

② C_2 ① — ② → X Not a cycle

↳ 1-2-2-1

↳ ~~The~~ Visited More Than ~~other~~ Once

③ C_3



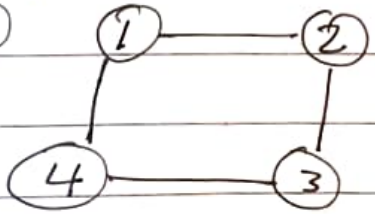
→ cycle Graph

↳ # Nodes = 3

degree = 2

C_4

④



→ cycle Graph

→ # Nodes = 4

degree = 2

Subject Date No.

→ we can Notice The following :-

↳ The degree in The ~~cyclay~~ cycle Graph it's always = 2

→ The Numbers of Nodes Must be ≥ 3

→ The sum (degrees) = $\Sigma = 2 * n$

The degree of Each Node ← = → The Number of Nodes

→ ~~The~~ # of Edges = $\frac{\Sigma * n}{2} = n$

*In The Exam :

→ a Proposal Question Could be as following

↳ Giving a description of a special Graph Give rules specifing The Number of edges , degree ... etc.

NOTEBOOK

Subject Date No.

[3] Wheel Graph :-

↳ The wheel Graph is a cycly Graph but There is a Node added in The center of the topology of the Graph.

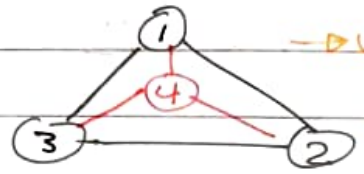
→ Wheel Graph is denoted by W_n
↳ where is n is the Number of Node in The cycle Graph
→ So, How Many Number of Nodes in The wheel Graph?
↳ n+1
↳ Why? why Not n = n+1 ?
↳ Bec The definiton said That The wheel Graph is a cycle Graph in The origin; Hence The Number of Node in The cycle Graph should Not change.

NOTEBOOK

Subject Date No.

Examples:-

W_4

① 

→ wheel Graph (W_3)

→ # of Nodes = $n+1$

→ # of ~~edge~~

→ degree = 3 ; ~~except~~

except the center Node

→ degree of the center Node = n

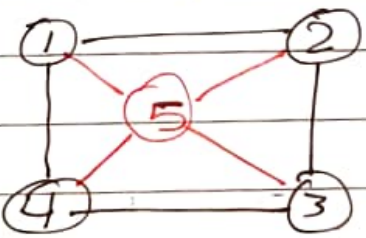
↳ # ~~edges~~ = sum(degrees)

Number of Edges for each Node

Number of Node

$= \frac{(3 * n) + n}{2} = \frac{4n}{2} = 2n$

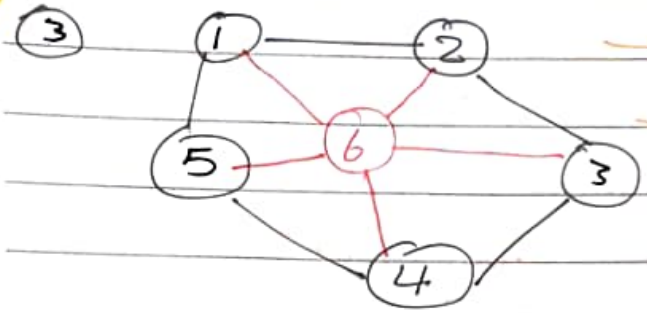
Number of Edges for the Center Node

② 

→ W_4

→ # of Nodes = 5 4

→ # of edges = ~~2*5~~ 2*4=8



→ W_5
→ # of Node = 6 2×n
→ # of edges = ≤ 10

4 Bipartite Graph:-

↳ Devi Dividing All Nodes in The Graph Into Two sets; The Nodes in each set Does Not have any edge Between Them.

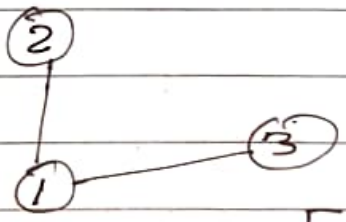
↳ edge Not Path

set 1 set 2

$$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix} \qquad \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_n \end{bmatrix}$$

* There is No edge between V_1 and V_2 in The Both sets

*Examples:-



→ is There an edge Between "1" and "2"?
Yes → Add to set 2

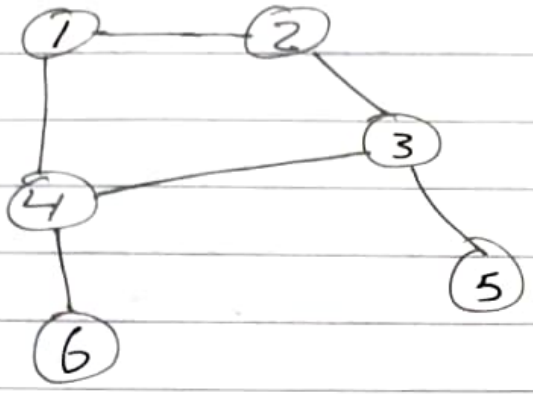
$$\begin{bmatrix} 1 \end{bmatrix} \qquad \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

→ . . . .

NOTEBOOK

Subject Date No.

(2)



$$\begin{bmatrix} 1 \\ 3 \\ 6 \end{bmatrix} \quad \begin{bmatrix} 2 \\ 4 \\ 5 \end{bmatrix}$$

~~→ One way to split The~~

→ One way could be used to determine if The Graph is a Bipartite Graph is Coloring system

↳ The Algorithm that is used from The coloring system is Two colors → Red Blue That will be used

→ Two colors works as The following :-

↳ Any Two ~~adjacent~~ adjacent-Node should Not have the same color

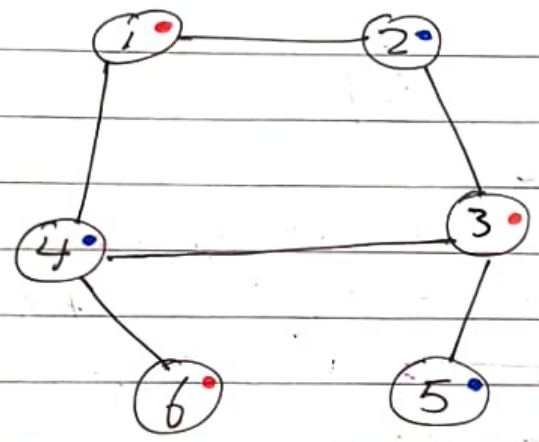
→ start from coloring a Node with red or blue and Then Color All Node connected with it with different color

NOTEBOOK

Subject Date No.

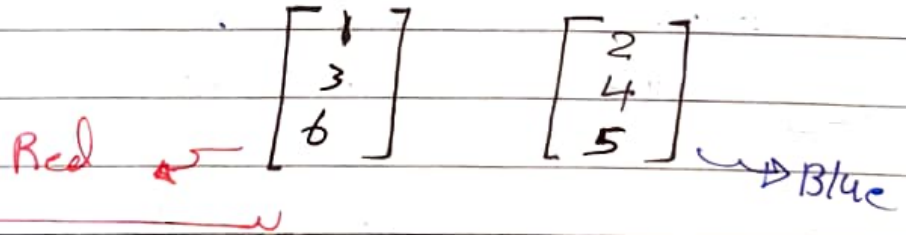
→ if There is a Two Adjacent Nodes & Have The same color Then The Graph is Not Bipartite obe.

* Note: In The Exam No colors is used, So use The Arthmetic operations e.g (+, *, -, /)



~~if condetion~~

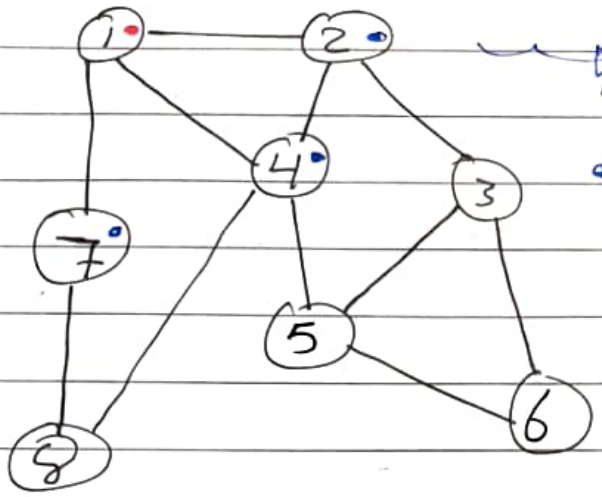
→ if The condetion was True, put The Node That have The same color in The same set.



* Adjacent Node: Two Nodes are adjacent if There is an Edge between Them

NOTEBOOK

3



→ Not Bipartite; "2" and "4" are adjacent Nodes Having The same color.

complete Bipartite Graph:

→ complete → Between Two Nodes There is an Edge

→ Bipartite → Two sets of Nodes!

→ complete Bipartite → Two sets of Nodes There is No edge between any TWO Nodes in The same set

AND

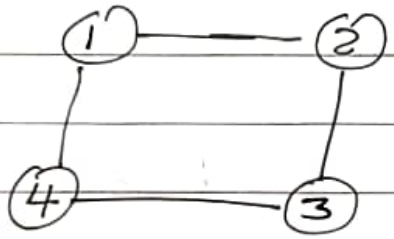
There Must be an Edge between a Node from "set1" and a Node from "set2"

→ K_n
↳ complete Graph Having "n" Nodes

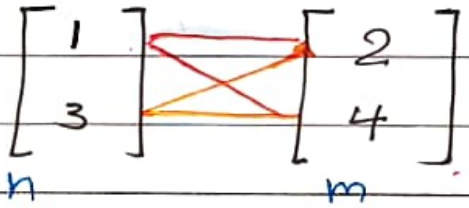
→ $K_{n,m}$
↳ complete Bipartite Graph Having Two sets → n → size
→ m → size

* Example :-

(1)



→ $K_{2,2}$
→ complete Bipartite Graph



* of Nodes in "n"

* of Nodes = 4

sum(degree) = (2*n) + (2*m)

(2*n) → degree of Each Node in The set "n"

(2*m) → degree of a Node in set "m"

* of edges = sum(degree) / 2

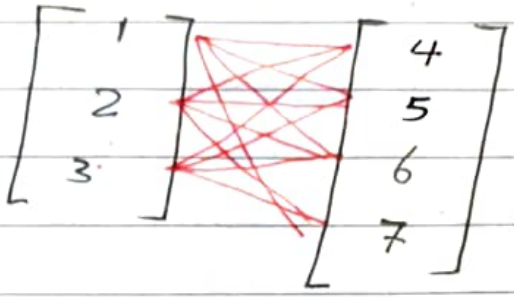
= $\frac{(2*2)+(2*2)}{2} = 4$

NOTEBOOK

Subject Date No

(2) consider a Graph Have The following Two sets

$K_{3,4}$
n m



→ complete Bipartite Graph
→ Each Node in "n" connected with Each Node in "m"

↳ # of Node: n+m = 7
→ degree of each Node in "n" = 4; why?
↳ complete Bipartite Graph
→ degree of each Node in "m" = 3
→ sum(degree) = (4*n) + (3*m)

→ # of edges = $\frac{(4*n)+(3*m)}{2}$

$= \frac{(4*3)+(3*4)}{2}$

$= \frac{24}{2} = 12$

* Note :- (: pass gün

Subject Date No.

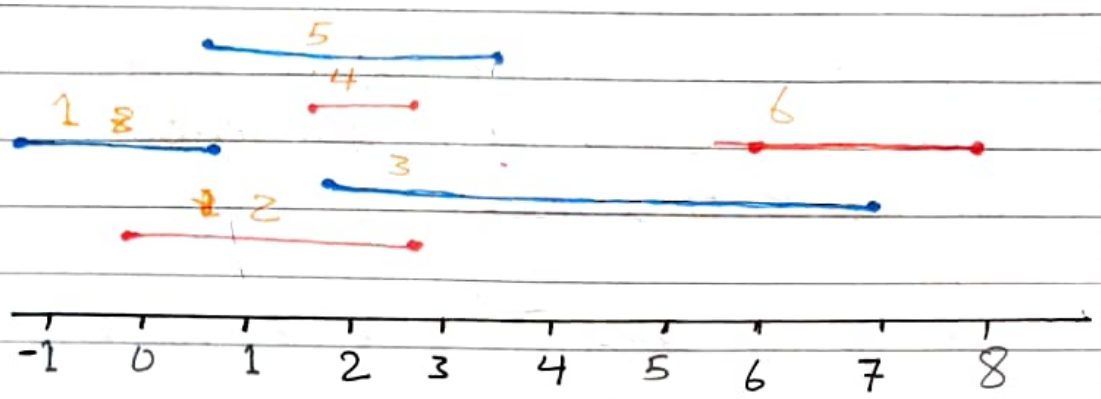
*Interval Graph:-

↳ *We can say That the interval Graph is a Graph where each Node Refresents an interval and The Edge between Any The Two Nodes Refresent The ~~interse~~ intersection Between Two intervals

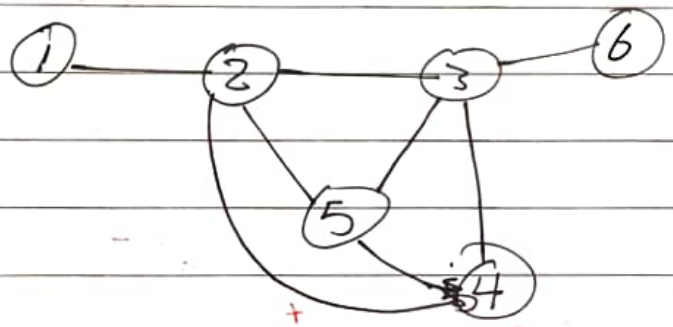
*Example:- suppose That you have The following Processes, and These Processes should be ~~schud~~ scheduled on The CPU ; Giving That Each Process have a start and End Time on The CPU

↳ sheduling is to Manage which of The Processes could be excuted Next on The CPU.

⇝ Processes ⟹ (0,3), (2,7), (-1,1), (2,3)
(1,4), (6,8)



→ Let's Represent Each Process as a Node



→ continue our Example: what is The Min Number of resources (CPU) Needed To Run These Prosses?

↳ The solution is by using The Coloring system

Coloring system ? why?

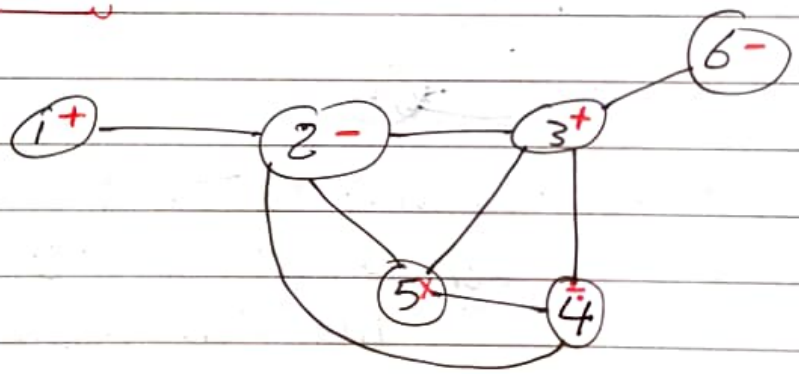
↳ Any Two Adjacent Node in The interval Graph ~~are~~ have intersection between Them , Then ~~can't~~ couldn't be ~~on The same~~ Processed on The same CPU

Subject Date No.

→ And

→ So; different colores for the Adjacent Node could be different CPUs for ~~these~~ the intersected Processes.

* How To use coloring system To solve The Problem?
  - ↳ starting from using one colore
    - ↳ if its possible to color All Nodes with one color ⇒ ~~easy~~
  - → else
    - ↳ Add Another color, and so on.



→ Min Number of resources = 4

→ Question:- if we colored the edges instead of coloring the Nodes; Is the Min Number of CPUs ~~will~~ will be still the same?
  - ↳ look for the Answer

NOTEBOOK

Subject Algorithms (17) الخوارزميات # Date No.

Graph Implementation:

~~→ it's Named by~~

→ The Meaning of Implementation of a Graph is How To Represents The Graph or How To store it in The computer Memory.

→ There 6 - 8 ~~ways~~ ways To implement The Graph, But we will use only Three of Them.

1. Adjacency List
2. Adjacency Matrix
3. Incidance Matrices

---

① Adjacency Matrix

↓

~~→ الجار~~

Two Nodes Are Adjacent only if There is an Edge Between Them

↳ is an 2-D Array where the number of rows equal to The Number of columns

↳ The Number of rows is equal to The Number of Nodes

Subject Date No.

* How The adjeancy Matrix works?

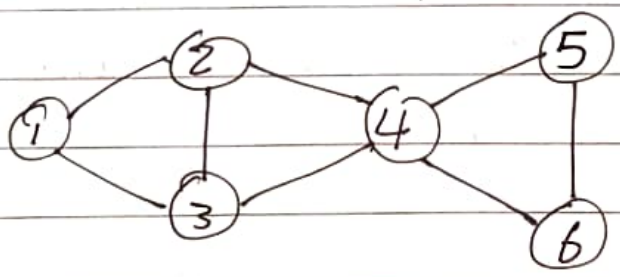
→ For Undirected Graph

↳ if there is an edge between Two Nodes where i = the first Node and j = the second Node we Put in the intesection between (i, j) in The Matrix ⇒ 1 → Refresent That there is an edge between i and j NOT Weight.

→ Else if there is Not Edge Between i and j we Put ∞ → Why Not "0"?

↳ 0 May indicate that there is a weight its value equal to 0.

→ consider The following Graph:-



i \ j	1	2	3	4	5	6
1	∞	1	1	∞	∞	∞
2	∞ 1	1 ∞	1	1	∞	∞
3	∞ 1	1	∞	1	∞	∞
4	∞	1	1	∞	1	1
5	∞	∞	∞	1	∞	1
6	∞	∞	∞	1	1	∞



→ How Many 1's in The row "2" ?

↳ 3

→ How Many 1's in The column "2"?

↳ 3

→ What is The degree for the Node "2" ?

↳ 3

→ We can Notice That in The Undirected Adjacency Matrix for The Undirected Graph That :-

→ Number of 1's in a row = column = degree

Subject Date No.

* In The Exam:

Proposed Question could be: Giving You a Adjacency Matrix of some Graph, and you have to Draw The Graph from The Matrix and Apply some Graph Algorithms such as: shortest Path, MST, etc

→ if you Draw

→ if you drew the Graph in wrong way All Answers dependent on The Graph will also be wrong.

* How to know if the Graph is directed or ~~Undirected~~ Undirected from The * Adjecency Matrix?

→ The Graph is undirected only if:

$(i,j) = 1$ and $(j,i) = 1$ ⇒ (i,j) and (j,i) Have the same Value

→ $(i,j) = (j,i)$

Ex: There is "1" between "1,2" and Also there is "1" between "2,1"

→ The Graph is directed if

$(i,j) \neq (j,i)$

Ex: $(1,2) = 1$

$(2,1) = \infty$

NOTEBOOK

Subject Date No.

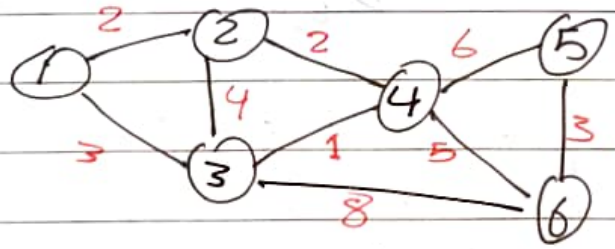
* How To know if the Graph is weighted or unweighted ?

→ Unweighted ? ⇒ The Graph is unweighted if it consistes only 1's and 0's
↳ special case :-
if All 1's in The Graph were weighted, How to differentiate it from unweighted Graph ?
↳ By asking the user who inserted the Graph To The Computer

In the Exam suppose That 1's are weights not just 1

→ Weighted ? ⇒ The Adjacency Matrix consistes of some Values and 0's

Ex :-

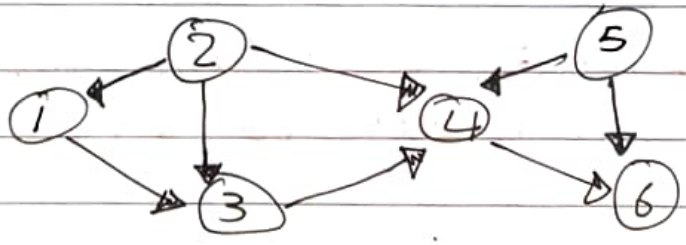


→ Represent This Graph by using adjacency Matrix.

Subject Date No.

i \ j	1	2	3	4	5	6
1	∞	2	3	∞	∞	∞
2	2	∞	4	2	∞	∞
3	3	4	∞	1	∞	8
4	∞	2	1	∞	6	5
5	∞	∞	∞	6	∞	3
6	∞	∞	8	5	3	∞

→ Representing Directed Graph by using Adjacency Matrix:-



i \ j	1	2	3	4	5	6
1	∞	∞	1	∞	∞	∞
2	1	∞	1	1	∞	∞
3	∞	∞	∞	1	∞	∞
4	∞	∞	∞	∞	∞	1
5	∞	∞	∞	1	∞	1
6	∞	∞	∞	∞	∞	∞

NOTEBOOK

→ From the prev Matrix, How to know if the Graph is directed or Not?

↳ By checking All Edges in The Matrix and if there is one edge where $(i,j) \neq (j,i)$ ⇒ Then The Graph is Directed.

↳ Else is Undirected Graph.

* Note: in real life There is No Full Bidirectional Graph
↳ where All edges in The Graph is Bidirectional

* Number of 1's in The row = out degree
* Number of 1's in one column = in degree
↳ Ex: Node "6" → indegree = 2 = # of 1's in row "6"
outdegree = 0 = # of 1's in col "6"

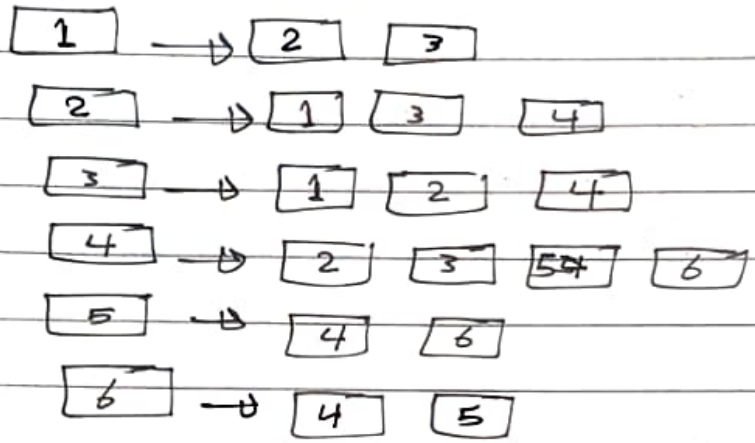
(2) Adjacency List:

↳ Representing The Graph by using a list, with size equal to the Number of Nodes; And Each Node has its own List consistes of the Adjacent Nodes with That Node.

Subject Date No.

Unweighted

~~Unweighted~~ Undirected Graph:-



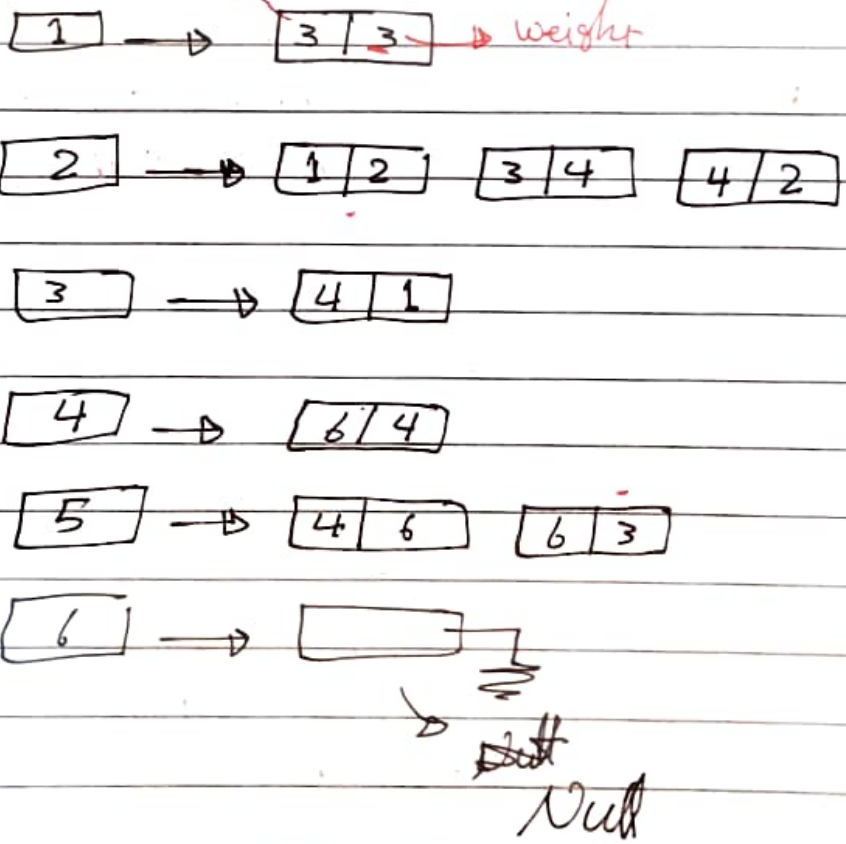
* The same Notes on The Adjacency Matrix could be Applied on The Adjacency list

How To know if The Graph is weighted or Unweighted, or directed Undirected?

→ Weighted ~~Undirected~~ Graph:-

Directed

Node



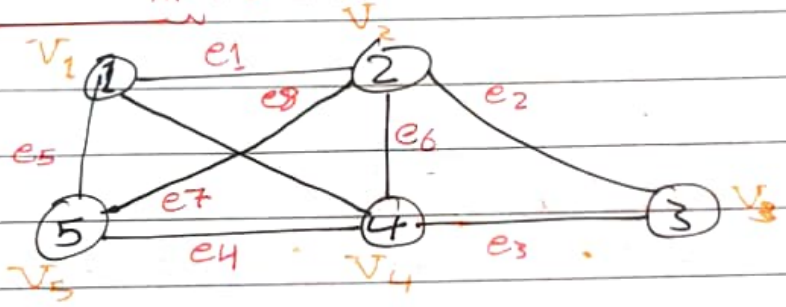
Subject Date No.

③ Incidance Matrices:

↳ 2-D Array; where The Number of Rowes = Number of Nodes, and Number of columns = Number of ~~En~~ Edges; ~~where~~

→ The incidance Matrix Refresents The Nodes in The Graph.

Ex:



	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
V_1	1	0	0	0	1	0	0	1
V_2	1	1	0	0	0	1	1	0
V_3	0	1	1	0	0	0	0	0
V_4	0	0	1	1	0	1	0	1
V_5	0	0	0	1	1	0	1	0

Rows in The incidance Matrix Refresents The Nodes And Each Row Refresent The connected Edges with The vertix (V_i)

→ Columns Refresents Edges

~~→ Note:~~

↳ And Each column Refresent The Nodes connected with an Edge (e_j)

Subject Date No.

* Note: The Number of 1's in Each column is exactly Two
  - why?
    - Bec: each edge ~~only~~ connect only Two Nodes

Exactly is Not Max or Min

- Adjacent Matrix —Represent→ Edges
  - The intersection Between Two Nodes Represents That there is an Edge between them.
  - How?
- Incidence Matrix —Represents→ Nodes
  - How?
  - "1's" in ~~The~~ a Row
    - Represents The existence of a Node V ~~on Edges~~ on The Edges from $e_1 \dots e_m$
      - Number of Edges
  - "1's" in The column
    - Represent's The Existence of Exactly Two Nodes in an Edge.

NOTEBOOK

Subject Date No.

*Note:-

→ The ~~in~~ out-degree in The Adjacency List could be calculated by:

↳ The size of the Adjacent List of That Node

→ And The in-degree in The Adjacency List could be calculated by:

↳ The frequency of The Node in The All Adjacent Lists

NOTEBOOK

Subject Date No.

* Graph Traversal :-

(1) Visiting All Nodes in The Graph
(2) The cost of revisiting The Node is Not Duplicated or repaid.

→ Two Main Algorithms for Graph Traversal :
(1) Depth-First search
(2) Breadth-First search

(1) Depth first Search (DFS) :-

* How DFS works?

Reachable + unvisited

→ starting from selecting a starting Node to start traversing from it.
→ Then we Proceed to all other Nodes that is reachable from The starting Point and its unVisited
→ we continue until there is No Any other unvisited Nodes in The Graph
→ But, based on the Prev, if we start traversing from the staring Node to Any reachable Node from the start Node until we reach the shallowest Node in the Graph different sequences of Paths will be Produced

Subject Date No.

The starting Node
or Any other Node

⇒ Why? Bec. it May have More than one reachable Node.

↳ So, we Need to Add one condition to Have The same sequence for the same starting Node.

→ The condition is → Alphabetical order

↳ Numbers = 1, 2, 3, 4, ---

Letters = A, B, C, D, ---

↳ So, in DFS we will Move from one to Another if it

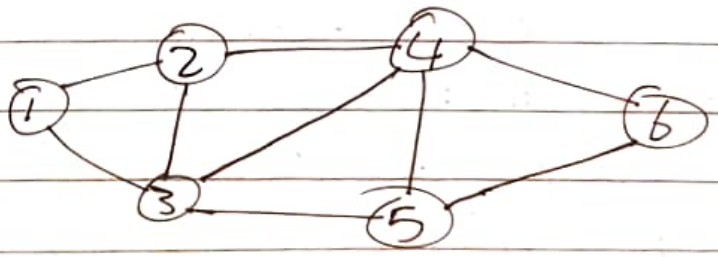
- Reachable
- Unvisited
- Alphabetical ordered.

*In The Exam:

The Alphabetical order May be Given for you by a word such as → CADK.

↳ The "C" becomes before "A" and so on.

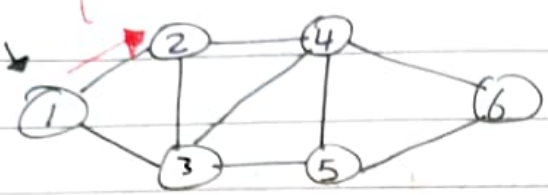
Tracing DFS by using The following Graph:



NOTEBOOK

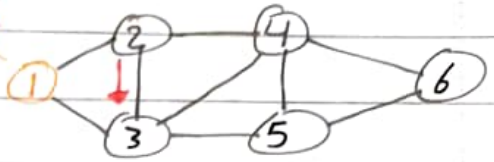
Subject Date No.

* Tracing Direction
Not an Edge

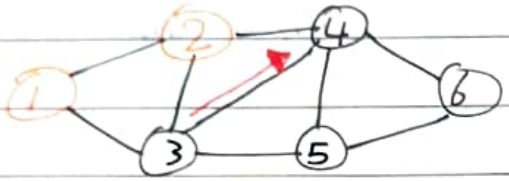


Step (1)

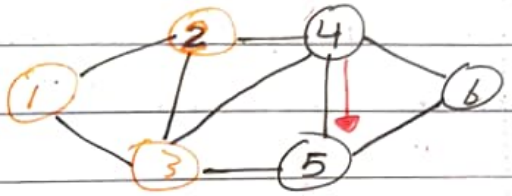
Visited Node



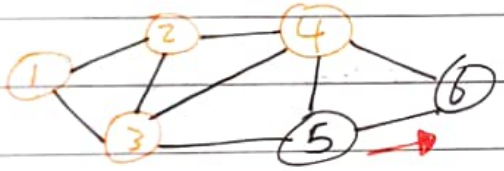
Step (2)



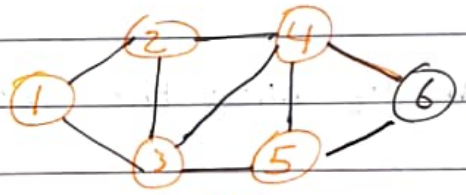
Step (3)



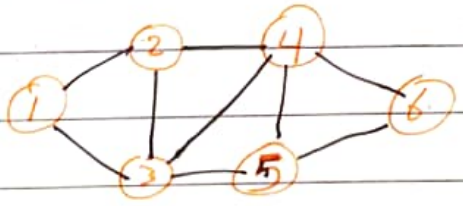
Step (4)



Step (5)



Step (6)



→ Path ①: 1 - 2 - 3 - 4 - 5 - 6

* Note: if we delete the Edge between "2" and "3"
The path will be

→ Path ②: 1 - 2 - 4 - 3 - 5 - 6

NOTEBOOK

Subject Date No.

* Note: if the Number of Nodes that is visited in the Path equal # to the Number of Nodes in the Graph
Then, the Graph is Connected
↳ There is a Path for each Node in the Graph (Undirected)

→ From Path(1) can we know if There is a Path Between "2" and "5" ?
↳ Yes
↳ Why? Because the Graph is connected; Then There Must be a Path between Them.

→ can We know the exact Path between "2" and "5"?
↳ No

* Note: The Traverse Traversing operation for directed Graph it's as same as the undirected Graph, But follow the Directions of the Edges.

→ Qustion: if we changed the starting Point; Does the Path change?
↳ ~~yes~~ yes

NOTEBOOK

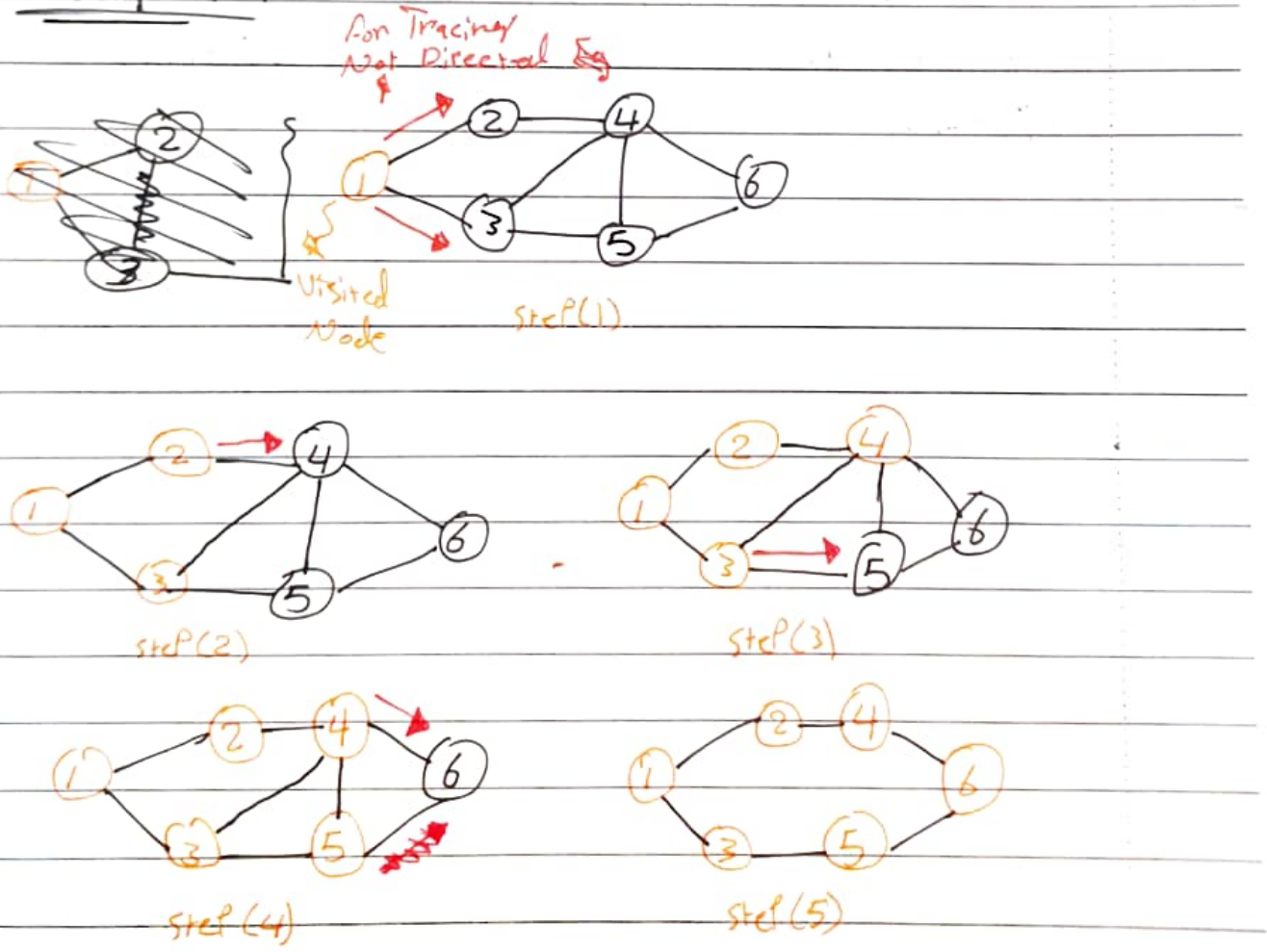
Subject Date No.

(2) Breadth_first search(BFS):-

- Traversing Nodes Level by Level
- Visit all Nodes connected to the starting Node, Then visiting All Nodes connected with the Next Alphabetical ordered Node and so on.

- The condition as same as DFS
  - Reachable
  - Alphabetical ordered
  - Unvisited

* Tracing BFS:-



Subject Date No.

→ Path: 1 - 2 - 3 - 4 - 5 - 6

Notes:

- The Path sequence Produced from BFS it's Not Alwayse equal to the Path Produced by DFS.
- Number of Nodes in the Path = Number of Nodes in The Graph, Hene the Graph is connected.
- There is No Any Benefit from knowing The order of The Nodes in The Path that is Produced by ~~BFS~~ BFS or ~~DFS~~ DFS.

Subject Date No.

* Some Graph Algorithms:

① shortest Path Algorithms
- unweighted Graph
- weighted Graph (Dijkstra's)
  - Positive weight
  - Negative weight ✗

This what we will study ↑

② ~~To~~ Tolological sorting
- DFS
- Khan's Algorithm

③ Network Flow Problem *

④ Minimum Spanning Tree (MST)

① shortest Path Algorithms:

→ To find The shortest Path in a Graph we have to identify a source Node and a Distination Node
↳ Then The shortest Path between source and Distination Nodes will be found.

→ Who identifies the source Node? source + Distination
↳ ~~Based~~ Based on The Problem

→ shortest Path: Min Number of Edges between Two Nodes

Subject Date No.

→ we have Two choices for selecting source Node and Distination ▷ single source — single distination

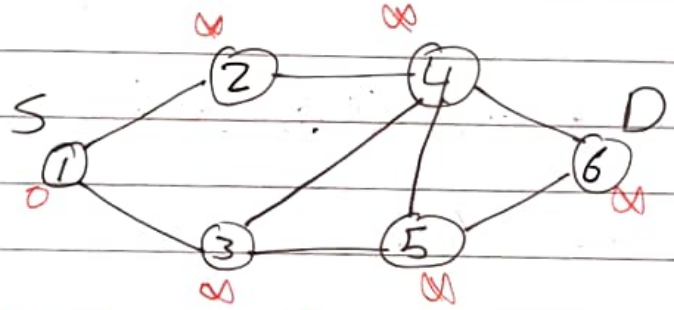
▷ single source — Multi distination.

*Note:- Multi source — Multi distination is Routing Algorithm work.

ذي هالوت

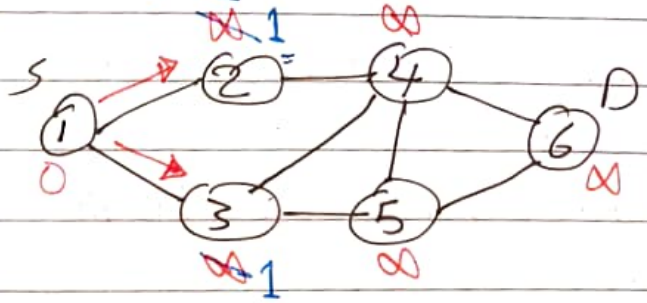
* Handling the shortest Path for Unweighted Graph

↳ Ex



Prev cost + 1

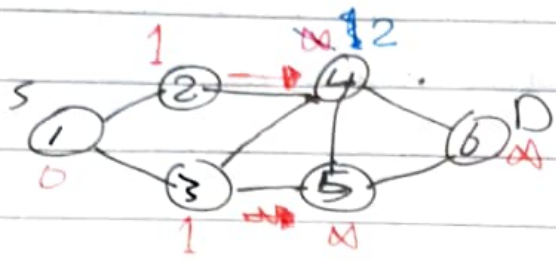
0 + 1 = 1



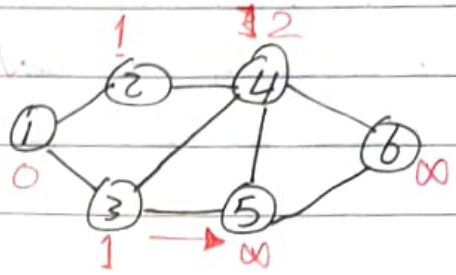
* The cost in The source Node = 0
↳ why? No Any So cost is Paid yet

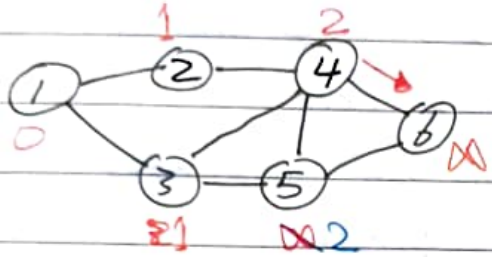
* The cost for the Nodes is: ∞
↳ why? The cost is Not known yet
↳ How To calculate The cost?
↳ Previous cost + 1
↳ why "+1"?
↳ we Move Each Time one Edge ⇒ 1

Subject Date No.

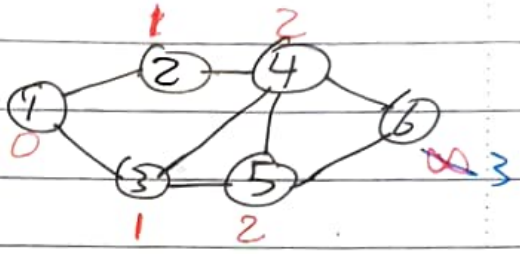


⇒

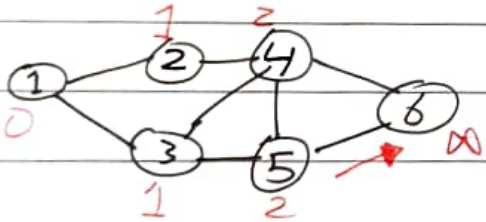




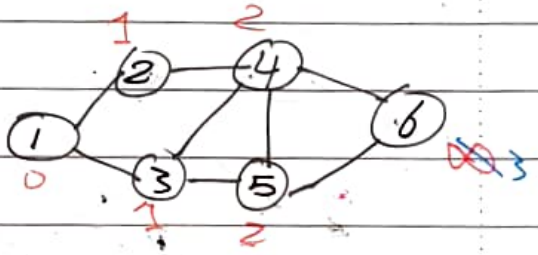
⇒



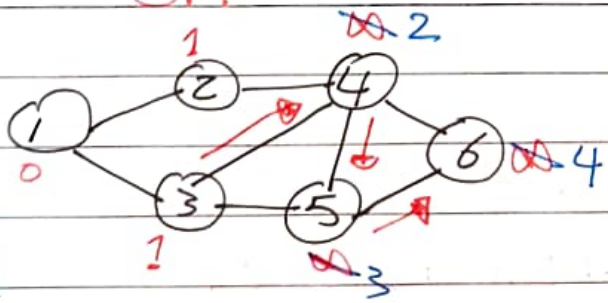
OR



⇒



OR



⇒ Not shortest Path

↳ Hence; Wrong Answer

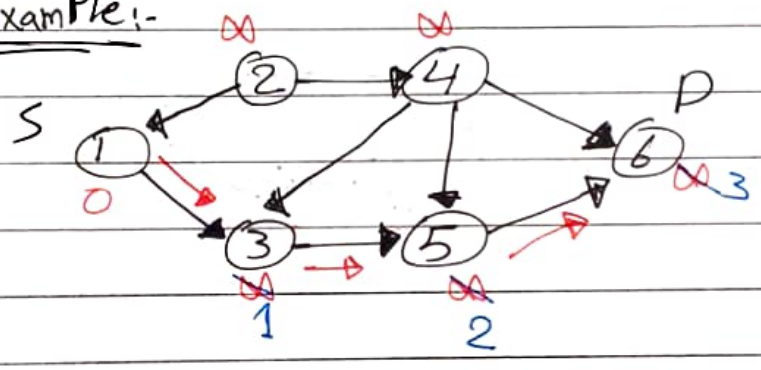
X

Subject Date No.

→ we are Handling The ~~two~~ unweighted Graph; but what About the Directed or Undirected Graphs?

Shortest Path + Network Flow + Topological sort
↳ Handles Both Directed or Undirected Graphs
→ but in Real Life Application There is No Meaning of the undirected Graph in ~~Point~~ Point of View from These Algorithms.

* Example:-



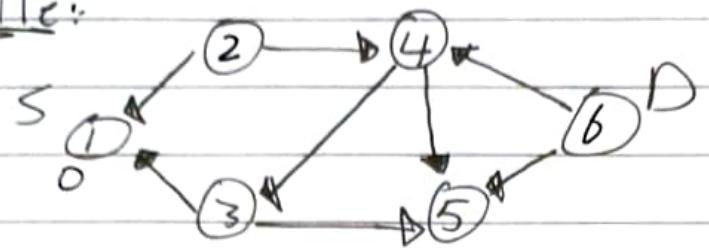
→ Path: ~~1-2~~
↳ 1 - 3 - 5 - 6

→ Is There Any other Path?
No; Because it's ~~Directed~~ Directed Graph.

* Note: in The shortest Path → we have to check All ~~Possible~~ Possible Path From Source to distination
↳ BFS is better Than DFS in This che

Subject Date No.

Example:



→ Is There Any Path from "S" To "D"?

↳ No, why?

→ The source Node has out-degree = 0

→ The distination Node has in-degree = 0

→ Note (important) :-

~~↳ A Node~~

→ A Node with out-degree = 0 ~~will~~

↳ it will ~~Here~~ Never be a soure Node

→ A Node with in-degree = 0

↳ it will Never be a distination Node

Subject : Algorithms (18) المحاضرة *

*outlines :-

- Dijkstra's Algorithm
- Minimum spanning Tree
  - Prim's Algorithm
  - kruskal's Algorithm.

*Dijkstra's Algorithm:

- One of the shortest Path Algorithms That finds The shortest Path From single source To Multi distination.
- The shortest Path is founded on weighted, undirected or directed Graph.
- The shortest Path Algorithm ~~were founded by~~ for unweighted Graph were The Min Number of Edges
- In contrast, in weighted Graph; The shortest Path is The Min cost (weight) between source and distination.
- Dijkstra's Algorithm Try All ~~Possible~~ Possible Path That's Leading To some Node in The Graph; Then Selects The Least cost Path between Them.

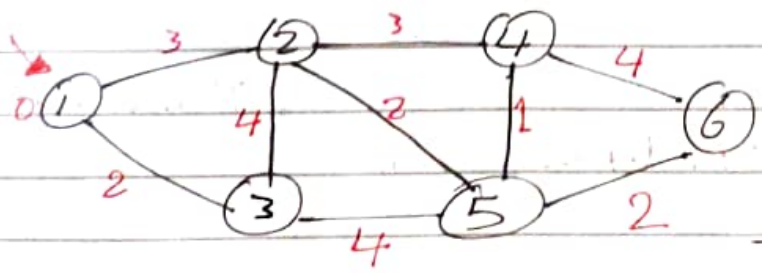
Subject :

✱ How Dijkstra's Algorithm works :.

- starting from the source Node, Reach or Visit All Nodes that is reachable from source Node.

- ~~calc~~
- calculate the cost that is paid for each reachable Node
- Insert the reachable Nodes in the Table
  - ✱ The Table will be explained in the Example
- Then, Do the same for the other reachable Nodes
- But:
  - if a Path from one Node to Another is founded AND it has less cost than the one exisits Replace the old Path with Min one
  - Else if the Path were Greater than the one exisits, Drop it

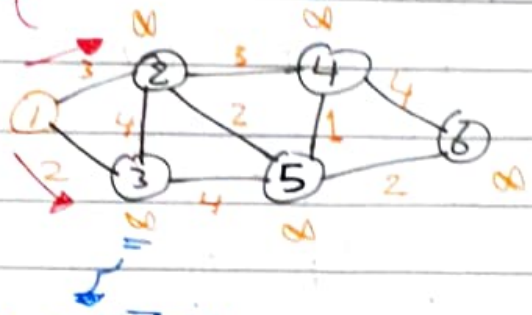
✱ Example :. Trace Dijkstra's Algorithm on the following Graph



Node	d_v	P_v	T
1	0	–	T
2			
3			
4			
5			
6			

— orange color for selected Node And path
— brdacks color for unselected or Dropped Node and Path

Subject already Direction for Tracing NOT Directed Edge



The cost That's Paid

Distance Date From Source Node to Node

Previous No. Node for Node V

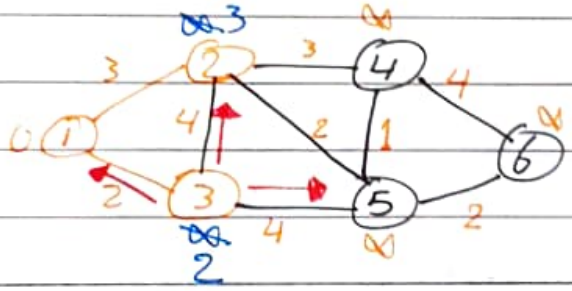
V	dv	pv	T
1	0	-	T
2	3	1	T
3	2	1	T
4			
5			
6			

True or false

True if The Node inserted into the Path False if Not

→ we Have Two Nodes that's we can Reach Another Node From Them; and They are "2" and "3".

↳ Let's start with "3"



V	dv	pv	T
1	10	-	T
2	3	1	T
3	2	1	T
4			
5	6	3	T
6			

→ From Node "3" we can go to Three Nodes:

① 3-1 → Drop it; Already selected

② 3-2 → Drop it; why?

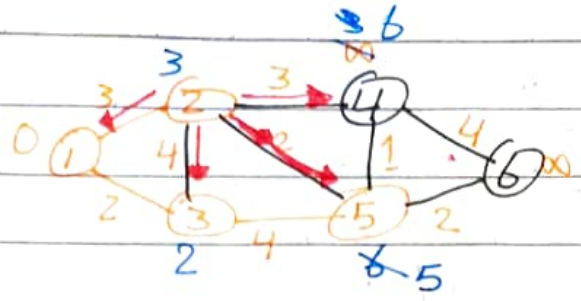
↳ The Path from 1-3-2 has cost of = 6 which is bigger than the exisiting Path (1-2) So, it will be Droped since we looking for The shortest Path

* Review The Prev Page

③ 3-5

↳ Not selected Before + shortest Path until Now

Subject Date No.



V	dv	Pv	T
1	0	-	T
2	3	1	T
3	2	1	T
4	6	~~3~~ 2	T
5	~~6~~ 5	3 2	T
6			

① 2 - 1 → Drop it; Already ~~s~~selected

② 2 - 3 → Drop it; ~~w~~Higher cost is Paid

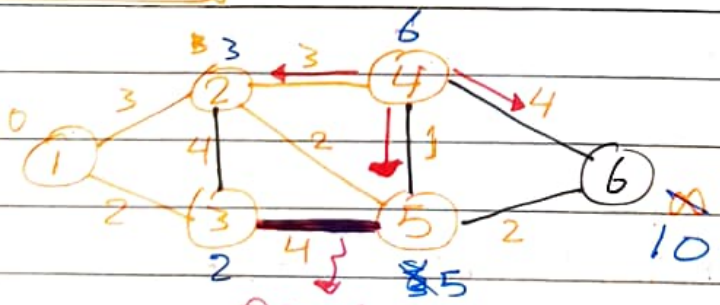
③ 2 - 5 → select it !! ⇒ why?

↓

The cost That is Paid from (1-2-5) is Less ~~Than~~ Than (1-3-5)

④ 2 - 4 → select it ⇒ why?

↳ Not selected Before



Droped Path

V	dv	Pv	T
1	0	-	T
2	3	1	T
3	2	1	T
4	6	~~3~~ 2	T
5	5	2	T
6	10	4	T

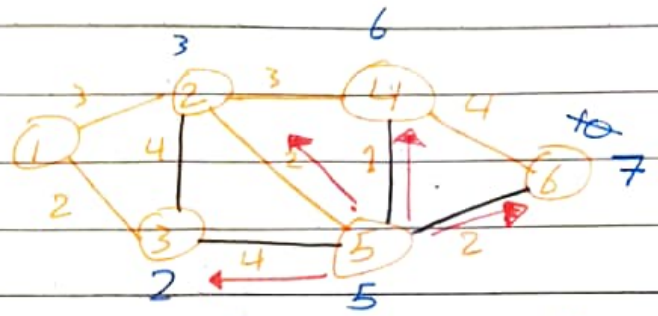
① H - 2 → Drop it; Already Selected

② 4 - 5 → Drop it; Higher cost

③ 4 - 6 → select it;

NOTEBOOK

Subject Date No.



① 5 - 3 → Drop ; Because it's Already Droped

② 5 - 2 → Drop it ; Higher cost

③ 5 - 4 → Drop it ; Higher cost

④ 5 - 6 → select it → why ?

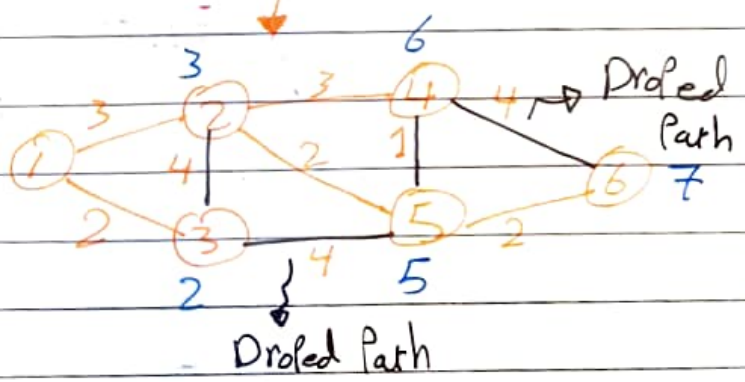
↓

The cost will be less

v	dv	Pv	T
1	0	-	T
2	3	1	T
3	2	1	T
4	6	3 2	T
5	5	2	T
6	∞ 7	∞ 5	T

The Final Graph is :

↓



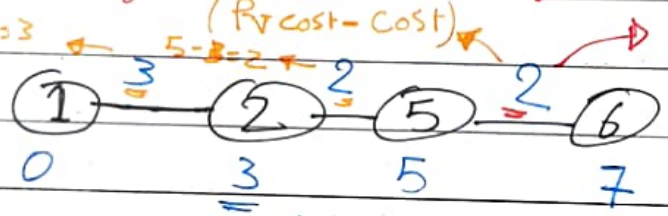
Subject Date No.

→ Question:- From The following Table, Can we find a Path from "1" to "6"?

V	d_v	P_v	T
1	0	-	T
2	3	1	T
3	2	1	T
4	6	3	T
5	5	2	T
6	7	5	T

↳ starting from The last Node in The Required Path which is "6"

↳ Then what is P_v for it And then what is P_v for the Previous Node until you reach Node "1"



↳ "d_v" from The Table

→ How?

↳ it was on a Node with cost of (5) Move to a Node with with cost of (7), How Much we Paid?

→ 2

→ Previous Node's cost - The current Node's cost

NOTEBOOK

Subject Date No.

✳ Minimum Spanning Tree (MST) :-

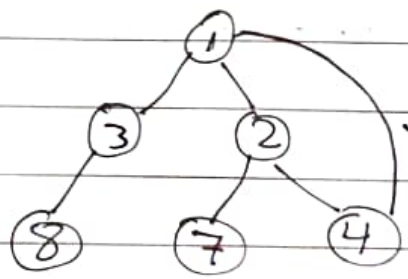
→ Minimum :- The Minimum cost to reach a Node in The Graph.

→ Spanning :- Reaching All Node in The Graph or (يعني) There is a Path to Any Node in The Graph.

→ Tree :- Converting The Graph into Tree

- what is a Tree?
  - Special Type of Graph characterized by The Following :-
    1. Root :- The Root in MST is Any Node in The Graph
    2. No cycles in The Tree :- when converting a Graph into MST we have to remove The cycles from The Graph
    3. No back Edges :- ~~to~~ grandchild
       - where the ~~child~~ has an Edge connected with the Root.

Ex :-



→ This is a back edge NOT Allowed in The Tree

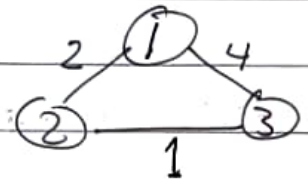
Subject Date No.

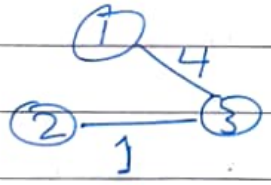
→ Summary:- MST it's converting a Graph into a Tree where All Nodes are reachable with Minimum cost for Any Path.

* Note:- MST works on Undirected Graph.

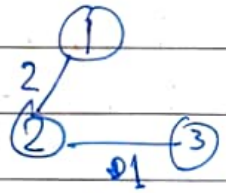
* Example on Removing cycles from The Graph:-

↳ How Many MST in The following Graph?

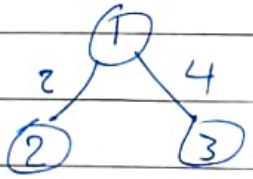




(1)



(2)



(3)

* ↳ Which one of Them is The correct MST?

↳ The one with Minimum cost

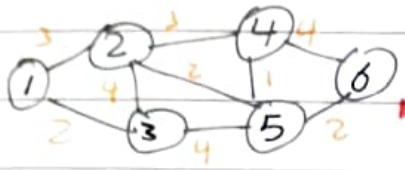
↳ How to Know The Minimum cost?

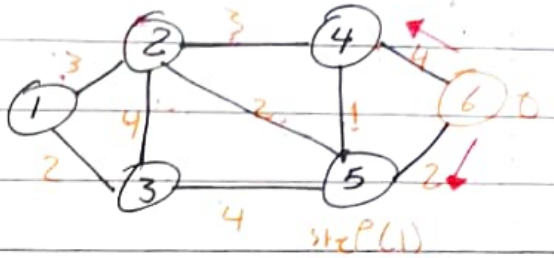
↳ consider the Next Example for Tracing The General MST Algorithm.

NOTEBOOK

Subject: Date: No.

* Example :-





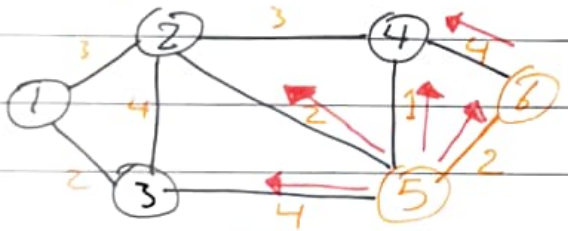
step (1)

- Try All Paths that's Reachable from "6".
- choose The Min Path
- Min Path is (6-5)
- The Tree will be :

(5) 2 (6)

- Then in each step Try All Posible Paths in The Tree

And choose The Path Non cycle



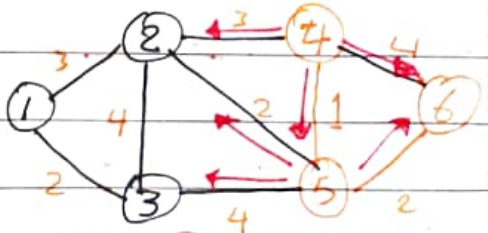
step (2)

- Try All Paths from "5" and "6"
- The Min Path is (5-4)
- The Tree will be

(4) 1 (5) 2 (6)

↳ Try All Paths from "4", "5", and "6" Then select The Min one and it's NOT making cycly path

step (3)



(2) 3 (4) 1 (5) 2 (6)

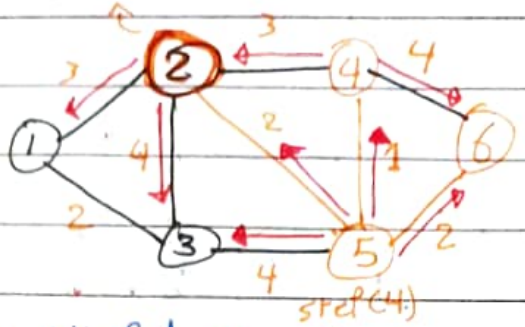
4-6 ⇒ Drop it; why?

↳ Makes The Tree cycle + Higher cost

#5-2 ⇒ #select it; Min cost in All Paths

Subject Date No.

Change Node :)



→ Try All Path from Each Node in The Tree

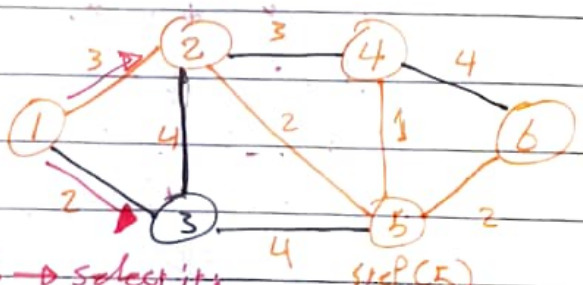
2 —2— 4 —1— 5 —2— 6

* 4-2 → Drop it; cycle Path
* 4-6 → Drop it; cycle Path + Already Dropped
* 4-5 → Drop it; Already selected
* 5-6 → Drop it; Already selected
* 5-4 → Drop it; Already selected
* 5-2 → Drop it; Already selected
* 5-3 → cost = 4
* 2-3 → cost = 4
* 2-1 → cost = 3

→ Select The Min one which is (2-1)

→ The Tree will be:

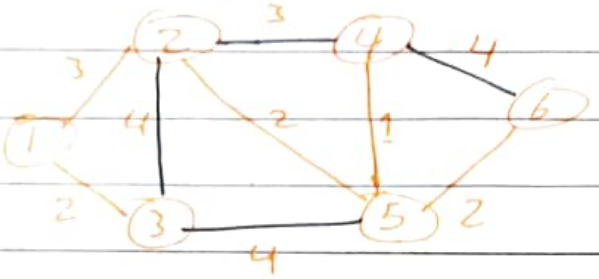
1 —3— 2 —2— 4 —1— 5 —2— 6



* 1-3 → select it; (why? Min & cost + All other Paths from other Node Either selected or cycle Paths)

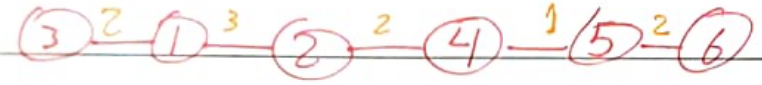
Subject Date No.

The final Graph is:



→ All Nodes Are visited !!

The final Tree will consisit of The following Nodes:



↳ In This Tree There is Path From Any Node To Another One

→ The Max cost for Any Path will be = 10

↳ The Max and That's Means There are others costs less than 10.

* MST Algorithms:
  - Prim's Algorithm ⇒ start from selecting Node
  - kruskal's Algorithm ⇒ start from selecting Edges

*Note: changing The start Node will Not change The final cost of MST; but The Topology of MST will be changed

↳ or Path sequence

NOTEBOOK

Subject Date No.

* Prim's Algorithm:-

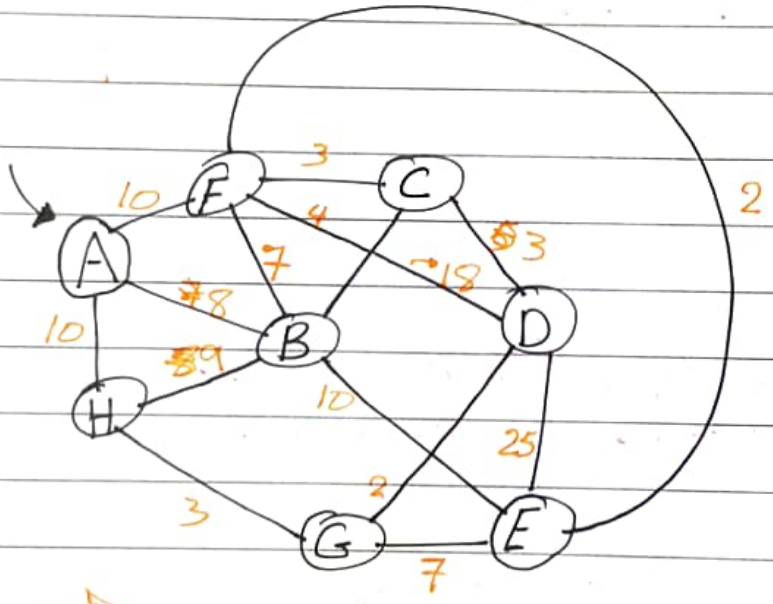
↳ Starting from selecting a start Node

→ $ choose The Min, Non cycle, unselected Path between All other Paths in ~~The Tree~~.

→ ~~from The~~ from The Nodes inserted in The Tree.

→ $ as same as the Prevouis Example.

Example:- Tracing Prims on The following Graph



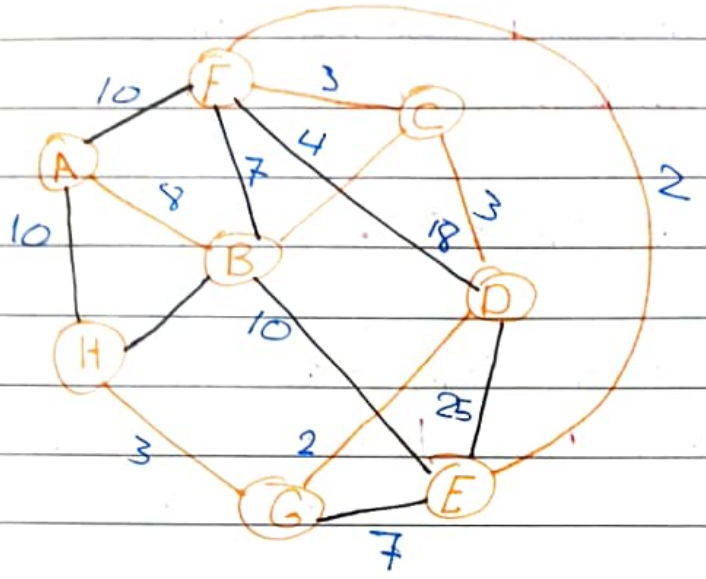
Proposal in The Exam

V	Tr	dv	Pv
A	T	0	-
B	T	8	A
C	T	4	B
D	T	3	C
E	T	2	F
F	T	3	C
G	T	2	D
H	T	3	G

Total cost = 25

Subject Date No.

→ After Applying Prim's Algorithm The Graph will be:



- orange color for selected Path or Nodes
- Black color for Dropped Paths

* Kruskals Algorithm:

→ Kruskal convert The Graph into Tree by selecting Edges

→ Since we looks up for The Minimum Spanning Tree So, we will sort The Edges in The Graph in an Ascending order (Min to Max)

* How kruskal's works:

→ Starting from listing All Edges in The Graph in an Ascending order

Subject Date No.

→ Then; select the Min Edge in The Table and Apply it in The Graph and Remember That the Edge should Not be a cycle.

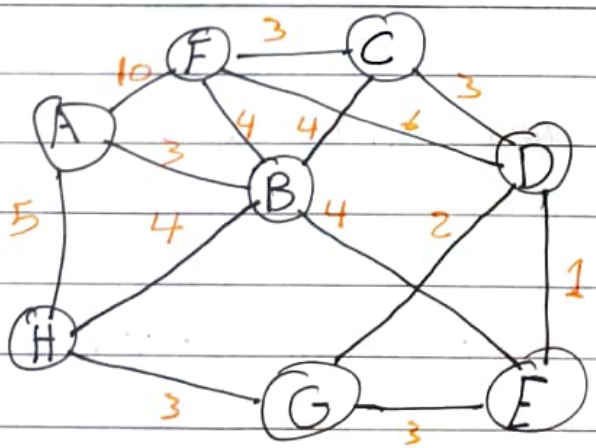
→ if we have More Than one edge you can select Any one first

↳ Hence; kruckal's Algorithm Gives you The Ability to choose The ToPology of The MST

⇒ Note:

kruskal's Algorithm could have More than one MST but All of Them have The same cost; BUT the ToPology for Each one of Them will be different

Example: APPly krustal's on The following Graph



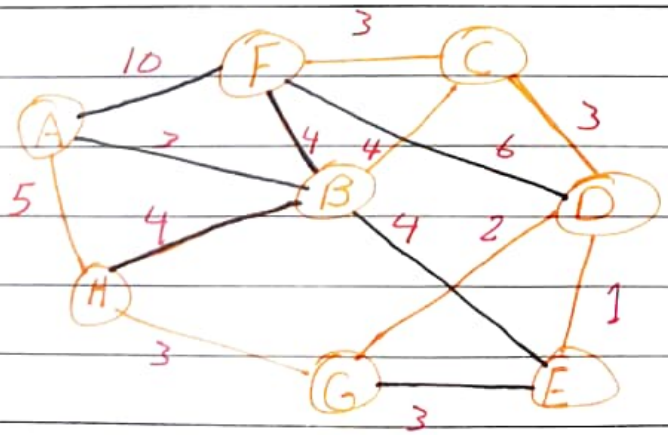
Subject Date No.

True or False

Edge	dv	T
DE	1	T
DG	2	T
CF	3	T
CD	3	T
HG	3	T
GE	3	F X
BC	4	T
BF	4	F
BH	4	F
BE	4	F
AH	5	T
DF	6	X
AB	8	X
AF	10	X

Cycle Path

* The Graph After Applying Kruskal's Algorithm:



Subject Algorithms (19) محاضرة #1 Date No.

Outlines :-

- Topological Sort
  - DFS
  - Khan's Algorithm
- Linear Time Sorting
  - Radix sort
  - Bucket Sort
  - Counting Sort

Topological Sort :-

↳ The Idea behind Topological sort That ~~is~~ we have some operations ~~of~~ ~~Proc~~. That's depending on each other

OR

There is a ~~Priority~~ Priority of completing some Tasks.

↳ And That's Mean you Can't do a Task without completing it's Prerequisite

→ Examples on Topological sorting :-

Ex1 : what is The Prerequisites for The "Algorithms" course?

C++ → OOP → DSA → Algorithms

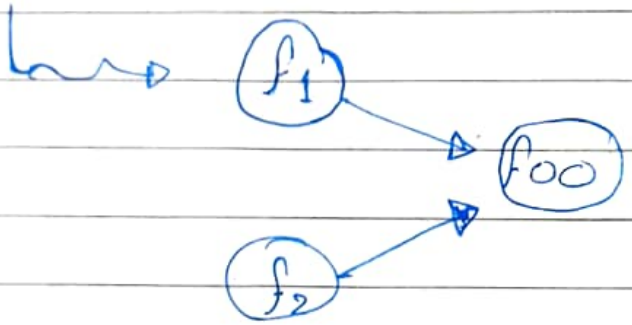
You can't Take "Algorithms" course without completing All other courses

NOTEBOOK

Subject Date No.

* Ex.2 :- The function "foo" Takes Two Parameters "a, b" where a is a returned value from "f1" function And b is a returned value from "f2" function

⟹ when The "foo" function could be excuted?
↳ when "a" and "b" is returned



* How The Topological sort will be Represented ?
(in other words How we will Represents That something depends(~~is~~) on Another Thing)

↳ By using Graph
↳ which Type of Graphs ?
↳ ~~Directed Connected Graph~~.
↳ Acyclic Directed Graph.

Acyclic → Graph That has No Any cycles in it.

Not cyclic Graph and Also Has No cycles

Subject: Date:

Conditions

* ~~Features~~ of Topological sorts Graph:

① At least there is one Node That has No Any Prerequisites;

↳ How to know That?

↳ The in-degree for such a Node equal zero "0"

Ex: Out Prerequisties → Not starting Point

(a)

→ Can we excute "a"?

↳ No; Why?

↳ The Arrow incidates for Prerequisites

~~② At least There~~

* Note: More Than one Node That we can start from it could be exisit - Two Nodes with in-degree = 0

↳ Ex: you can Take the "Discrete Math" course ~~simu~~ Simoultansly with "C++" course

② There is At least one Node with Out-degree equal to zero

↳ you Have to end in some Point

③ No cyclies in The Graph.

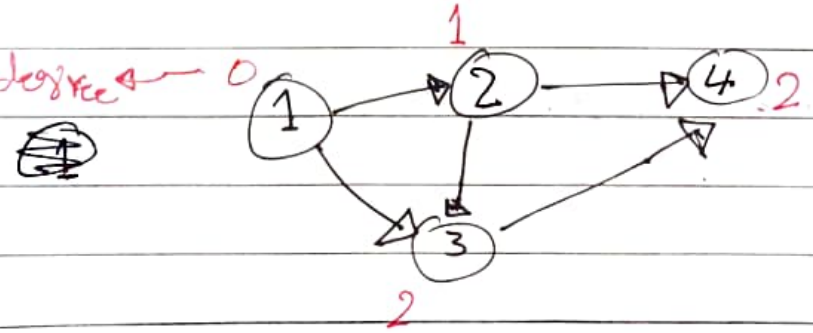
Subject Date No.

* There is Two Algorithms for Tolological sorting
  - Khan's Algorithm
  - DFS → Gift for You :)
    - Trace it by yourself.
    - Does The DFS have The same Path order of the Khan's Algorithm? Look for The Answer

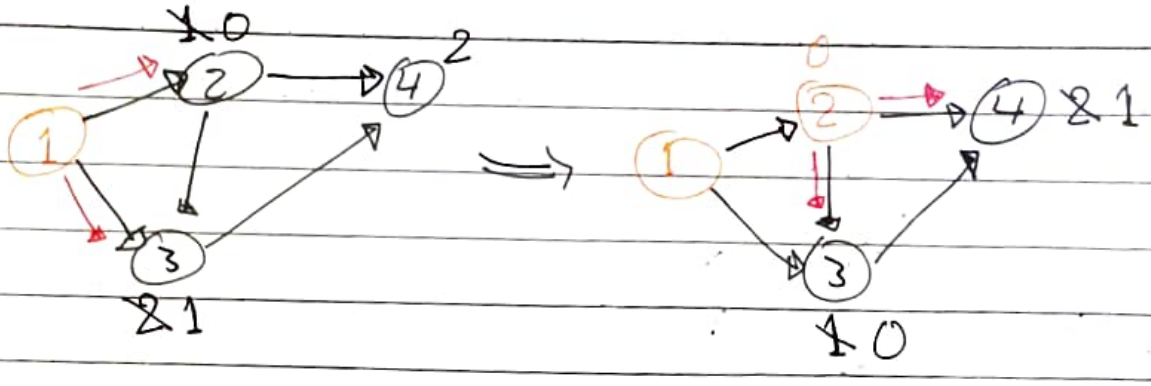
* Khan's Algorithm:

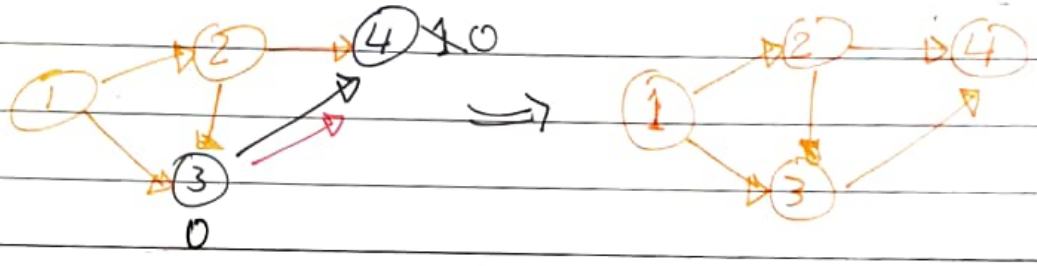
↳ How it works? a Node
- ⊛ starting from in-degree equal to zero ~~Node~~
- ⊛ substract "1" from each Node connected with the Node that we have started with
  - ↳ why substracting?
    - ↳ for Example ~~to~~ There is a course which Have Two Prerequisties, if one has been taken Then it will have one Prerequisties and so forth.
- ⊛ Repeat The Previous Two Steps for All Nodes until you

Example: consider The following Graph:



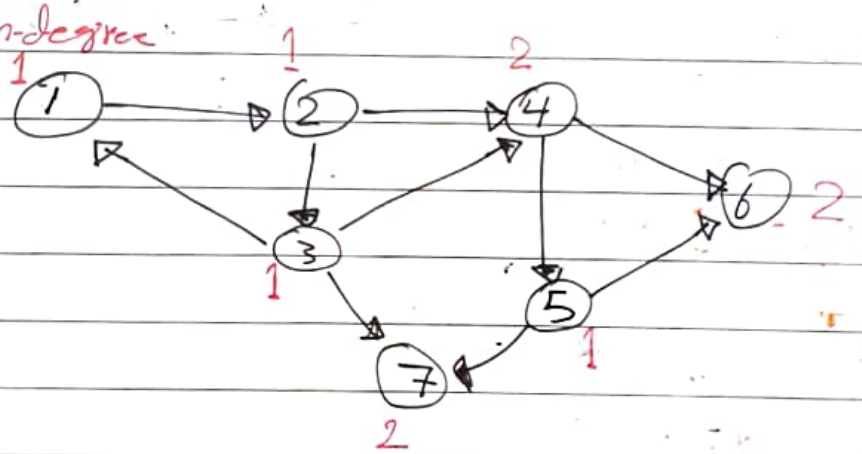
Subject Date





→ Path : 1, 2, 3, 4

Ex :- in-degree



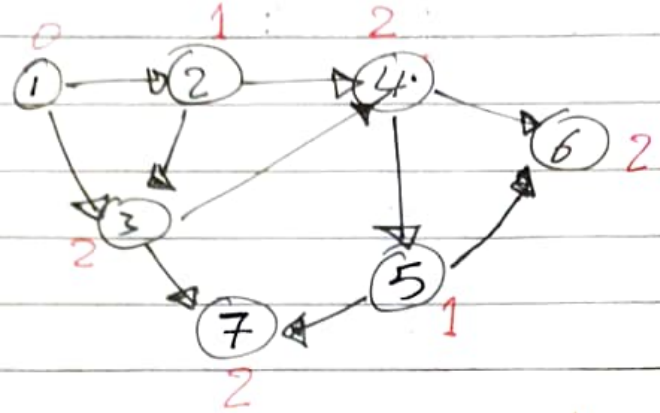
⇒ Can Not Apply Topological sort in The Previous ~~Graph~~ Graph; ~~why?~~

↳ why?

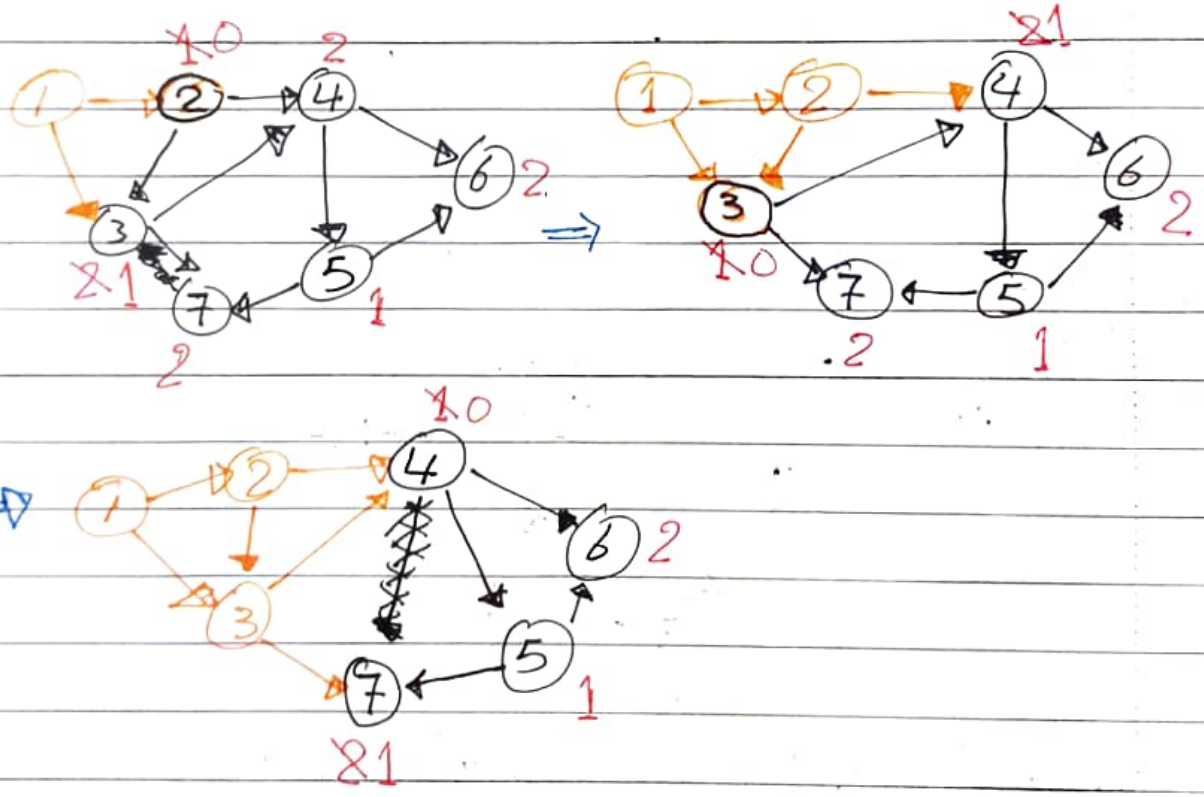
↳ There is No Any in-degree = 0

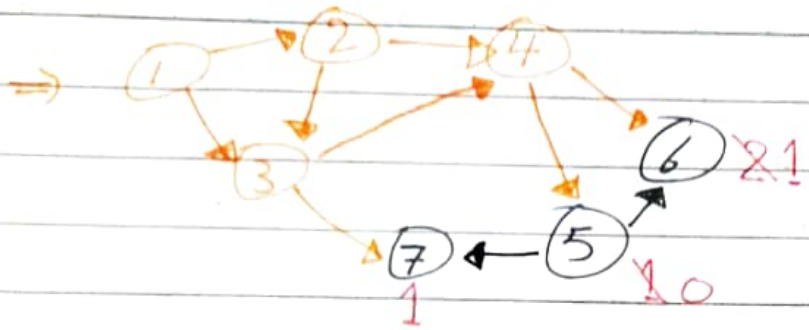
and There is a cycle between "1", "2" and "3"

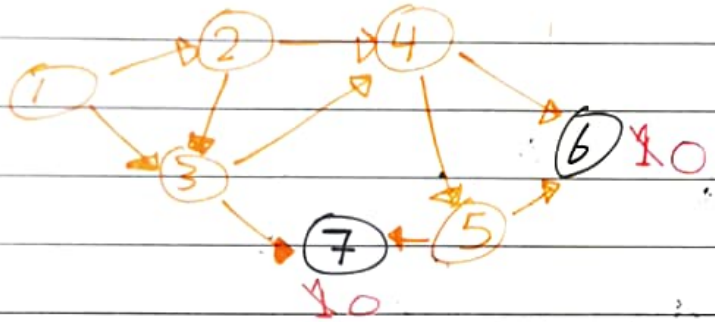
→ ok, if we Modify The Prev to be as The following



→ starting from Node "1"







↳ We can Notice That There is Two End Nodes in The Graph → How?

↳ The out-degree for each one of them is equal to one.

→ So, if we have More Than one End we can Go Next to Any one of Them.

↳ Then we will have More Than one Path

→ Paths: ① 1-2-3-4-5-6-7

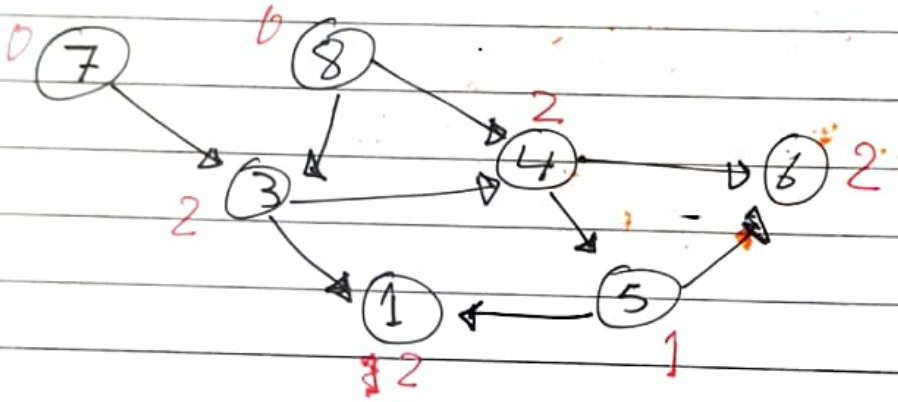
② 1-2-3-4-5-7-6

Subject Date No.

*Note: Also We could have More Than one Node with in-degree = 0.

↳ Hence; There is More Than one Path Each of starting from different start Node.

↳ Ex.



*Paths:

1. 7-8-3-4-5-6-1
2. 7-8-3-4-5-1-6
3. 8-7-3-4-5-6-1
4. 8-7-3-4-5-1-6

Subject Date No

*Linear Time Sorting Sorting :

↳ "Linear Time" ⇒ The Big-O for These Algorithms is O(N)

"Sorting" ⇒ The sorting Algorithms ~~That's~~ The Big-O is O(N); And They are:

1. Radix sort
2. Bucket Sort
3. counting Sort

In Lect (4), we called Them "Distribution sort"; and The Distribution sort is Not Based on The comparisons between Numbers or characters or strings.

→ if There is an Algorithms That's work in O(N); why ~~the~~ we use Another Algorithms with $O(N^2)$ or $O(N \log(N))$?

↳ Because The Linear Time sorting Work Under ~~certain~~ certain Conditions

Subject Date No.

* Radix Sort :-

⇒ How The Radix sort works?

↳ By comparing item digit bey digit ; ~~for Example~~

called Most significant Digit (MSD) OR Most significant Bit (MSB) ← 1 2 7 → ~~Most~~ Least significant ~~Digit~~ Digit (LSD) OR Least sinificant Bit (LSB)

digit 1 = d_2 = d_3

⇒ Based on (MSD) and (LSD) ; The Radix sort could be implemented by starting from (MSD) or starting from (LSD)

⇒ Radix sort by using (MSD) , How it work's?

↳ suppose That There are a Group of Numbers , starting sorting These Number from (MSD)

→ Put All Number That Have The same digit in one Group

→ After The Numbers sorted based on The first digit or (MSD) we Move To The seconde digit

NOTEBOOK

Subject Date No.

→ Do the same with
→ Do the same until Each Number in it's own Group

*Note:- when we sort by using (MSD), No Any Number Moves from one Group to Another

Example:

comparing with first digit

921	127	110	110	
127	123	127	123	110
291	110	123	127	123
123	291	209	209	127
871	209	291	291	= 209
460	370	370	370	291
370	460	410	410	370
209	410	460	460	410
410	871	871	871	460
110	921	921	921	871
				921

↑ compare with the second digit in Groups That contains More Than one Number

*Note:- The Black Line Means a Seperate Group

Subject Date No.

Radix sort by using (LSD). How it works?

→ starting from the ~~LSD~~ (LSD) and sort the Numbers in ascending (صعودي) order based on LSD

* Note: Numbers May be Moved from one Group To Another in (LSD) Radix sort, So There is No ~~Use~~ Meaning of Grouping Number
Hence; in each step ~~will~~ The Number will be sorted But the digit is different

→ Then; Move ~~the~~ To The digit That's Before (LSD) And Re-Sort The Numbers based on it

→ Repeat These steps for All digits for All Numbers.

* Note: Keep The order of Numbers ~~to~~ when Moving from one step to Another
→ Do Not sort Arbitrarily

→ Application from Real Life on LSD Radix sort That is identifying The virtual IP Address in Virtual Networks.

Subject Date No.

Example on LSD Radix Sort :-

LSD	second digit	third digit	
92[1]	4~~1~~0 4~~3~~0	209	110
127	~~1~~10 370	410	123
291	~~9~~91 410	110	127
123	~~3~~ 110	921	209
871	921	123	291
460	291	127	370
370	871	460	410
209	123	370	460
410	127	871	871
110	209	291	921

↑ *Sorted

→ condition of Radix sort :-
- ↳ All Numbers should have The same Number of digits.

→ The Big-O ?
- ↳ O(N) ⟹ How?
  - ↳ We iterate on N numbers and on each Number There is a "d" digities which is constient, The O(N*d) = O(N)

NOTEFULL

Subject Date No.

*Bucket sort :

- The idea of The Bucket sort is similar to The idea of The Hash Table
- Each Bucket Represents an Category of Values or we could say as a key value.

- Condition for The Bucket sort:
  - All Numbers should be between 0 and 0.99 which is [0,1)
    - open interval

** Bucket sort; How it works:

- Given a set of Numbers between [0,1); to sort Them by using Bucket Sort ~~the~~ The following should be done:-
  - Multiply All Numbers by 10
  - Take The first digit (The one before The comma)
  - The first digit will be between 0 and 9
  - Make "10" Buckets from 0 to 9
  - Put The original Values based on The first digit in The Proper Bucket

- *Note: original Values ~~The~~ The values Before Multiplying Them by 10

NOTEBOOK

Subject Date No.

Example: sort The Following Numbers by using Bucket sort

(0.01, 0.92, 0.31, 0.42, 0.91, 0.505, ~~0.43~~ 0.44)

→ Multiply by 10

Bucket Number

~~for~~ (0.1, 0.2, 3.1, 4.2, 9.1, 5.05, ~~4.4~~ 4.4)

0 → ~~0.1~~, 0.01

1

2

3 → ~~3.1~~ 0.31

4 → ~~4.2~~, ~~4.4~~ 0.42, 0.44

5 → ~~5.05~~ 0.505

6

7

8

9 → 0.91, 0.92

* Note: Determining ~~what~~ ~~number~~ Are the Numbers belong to which Bucket depending on The first ~~dig~~ digit After Multiplying by 10.

BUT: The values ~~of~~ ~~Put~~ in Each bucket is The values Before Multiplying by 10

Subject Date No.

=> Application of Bucket sort:

↳ wireless communication; were Each Bucket in Networking is devided and sent to The reciver; And The reciver Reassemble The Bucket by using Bucket sort; which is stable sorting

* Note: The values in Each single Bucket is Not sorted; To sort Them use Any sorting Algorithm Discussed Before

counting Sort:

↳ Based on counting The frequencies of The items and Defending on The frequencies The items are sorted.

=> condition:

↳ In counting sort you Must have a fixed range in The Numbers ~~That will be sorted~~

The Range is: 1 → k

↳ where "k" is The Max Value in The Array

The Range Is Not Requency or counter for The Numbers; it's just an Range of values

Subject Date No

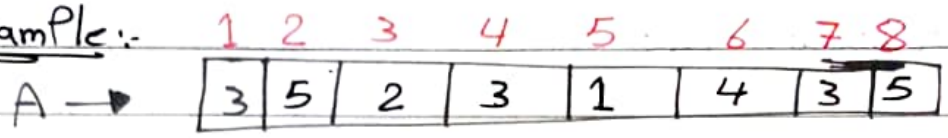
→ Counting Sort; How it works:

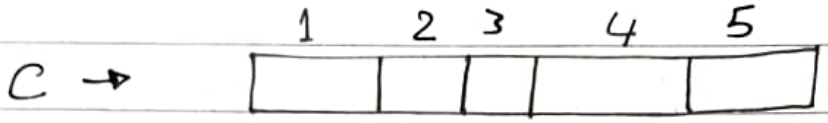
→ To start sorting by using counting sort, Three Arrayes Needed

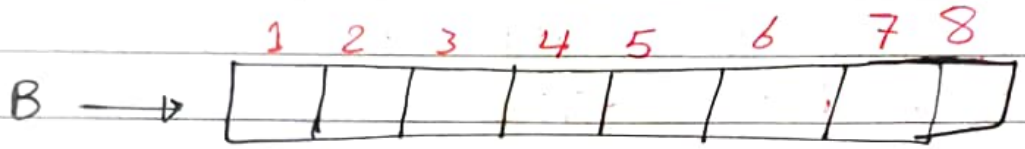
1. Array "A" ⇒ input Numbers; size = N
2. Array "B" ⇒ output sorted Numbers; size = N
3. Array "C" ⇒ ~~Five~~
   - ↳ The Array "C" consisites of The frequencies for Each Number; where the index value refer to the Number in "A" and the value is The frequency
   - → After calculating The frequencies for Each Number The proper indexies will be calculated by using frequencies of Each Number.
   - → The Array "C" could be considered As Mapping Between "A" and "B"
   - → The size of "C" is = k And The Indexes of "C" is from 1 to k

Subject Date No.

Example:-





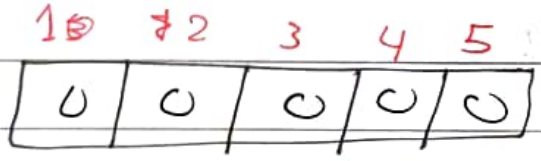


* Code And Tracing:-

① for i=1 to k
  C[i] = 0;

} To store "0" in All of "C" Elements → why?

The frequency for Each Elements is Not known yet



→ Complixity = K

② for j=1 to n
  C[a[j]] = C[a[j]] + 1

↳ To find The frequency of Each Elements in The Array "A"

Subject Date No.

→ Tracing The ② Loop:-

j=1

C[a[1]] = C[a[1]] + 1

C[3] = C[3] + 1

C[3] = 0 + 1

= 1

1	2	3	4	5
0	0	1	0	0

j=2

C[a[2]] = C[a[2]] + 1

C[5] = C[5] + 1

C[5] = 0 + 1

= 1

1	2	3	4	5
0	0	1	0	1

j=3

C[a[3]] = C[a[3]] + 1

C[2] = C[2] + 1

= 0 + 1

= 1

1	2	3	4	5
0	1	1	0	1

j=4

C[3] = C[3] + 1

= 1 + 1

= 2

1	2	3	4	5
0	1	2	0	1

Subject Date No.

j=5

C[1] = C[1]+1
= 0+1
= 1

1	2	3	4	5
1	1	2	0	1

j=6

C[4] = C[4]+1
= 0+1
= 1

1	2	3	4	5
1	1	2	1	1

j=7

C[3] = C[3]+1
= 2+1
= 3

1	2	3	4	5
1	1	3	1	1

j=8

C[5] = C[5]+1
= 1+1
= 2

1	2	3	4	5
1	1	3	1	2

NOTEBOOK

Subject Date No.

(3) for i=2 to k
C[i] = C[i] + C[i-1]

↳ We said That The counting sort sorting Numbers Based on The frequency of Each Element

⟹ How?

↳ if ~~you~~ we know That an Element from Array "A" is The Max Element in "A" Then where The Max Element should be in The Array "B"?

↳ After All The Elements

↳ So, in index = 5?

↳ NO; Because The Elements Before The Max Element could be Repeated

Hence; we have To calculate The Number of The Previous frequencies

↳ ~~How~~ How?

↳ By using The Previous Loop (3)

Subject Date No.

*Tracing loop (3) :-

i = 2

C[i] = C[i] + C[i-1]

C[2] = C[2] + C[1]

= 1 + 1

= 2

1	2	3	4	5
1	2	3	1	2

i = 3

C[3] = C[3] + C[2]

= 3 + 2

= 5

1	2	3	4	5
1	2	5	1	2

i = 4

C[4] = C[4] + C[3]

= 1 + 5

= 6

1	2	3	4	5
1	2	5	6	2

i = 5

C[5] = C[5] + C[4]

= 2 + 6

= 8

1	2	3	4	5
1	2	5	6	8

Note:-

index = 5

Value = 8

Subject Date No.

Note:- The Array "C" Now contains The Last index for Each Element in The Array "B"

↳ Remember That Array "C" Map the Array "A" and "B".

⇒ Ex:- index = 5

Value = 8

↳ The value "5" ends at index "8" in The Array "B"

↳ But How to know How Much "5" Element in The Array "B" will be?

↳ Based on The Previous Element's value in Array "C"; which is

⇒ Previous value + 1

6 + 1 = 7

↳ So, The value "5" will be stored in The Array "B" from "7" To "8"

Subject Date No.

(4) for j=n To 1

B[C[A[j]]] = A[j]

C[A[j]] = C[A[j]] -1

To Move to The Next index # for Each Element

↳ To Moving The Elements from "A" To "B" By using The values in "C", as were discussed in The Previous Page

* Tracing Loop (4) :-

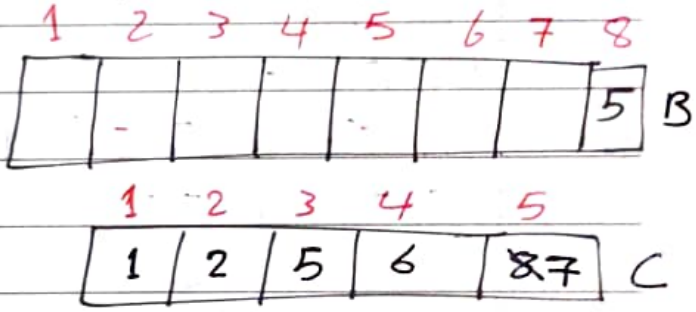
j=8

B[C[A[j]]] = A[j]

B[C[A[8]]] = A[8]

B[C[5]] = 5

B[8] = 5



C[A[j]] = C[A[j]] -1

C[5] = C[5] -1

= 8 - 1

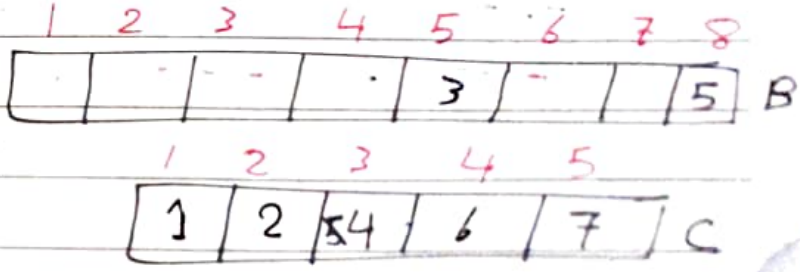
= 7

j=7

A[7]

B[C[3]] = A[5] A[5]

B[5] = 3



C[3] = C[3] - 1

= 5 - 1

= 4

Subject Date No.

j = 6

B[C[4]] = A[6]

B[6] = 4

C[A[6]] = C[A[6]] - 1

C[4] = C[4] - 1

= 6 - 1

= 5

A

1	2	3	4	5	6	7	8	
				3	4		5	B

1	2	3	4	5	
1	2	4	~~5~~	7	C

j = 5

B[C[1]] = A[5]

B[1] = 1

C[A[5]] = C[A[5]] - 1

C[1] = C[1] - 1

= 1 - 1

= 0

1	2	3	4	5	6	7	8	
1				3	4		5	B

1	2	3	4	5	
~~1~~0	2	4	5	7	C

j = 4

B[C[3]] = A[4]

B[4] = 3

C[A[4]] = C[A[4]] - 1

= 4 - 1

= 3

1	2	3	4	5	6	7	8	
1			3	3	4		5	B

1	2	3	4	5	
0	2	~~4~~3	5	7	C

Subject Date No.

j = 3

$B[C[2]] = A[3]$

$B[2] = 2$

$C[A[3]] = C[A[3]] - 1$

$C[2] = 2 - 1$

$= 1$

	1	2	3	4	5	6	7	8	
B	1	2		3	3	4		5	

	1	2	3	4	5	
C	0	~~1,2~~ 1	3	5	7	

j = 2

$B[C[5]] = A[2]$

$B[7] = 5$

$C[A[2]] = C[A[2]] - 1$

$C[5] = C[5] - 1$

$C[5] = 7 - 1$

$C[5] = 6$

	1	2	3	4	5	6	7	8	
B	1	2		3	3	4	5	5	

	1	2	3	4	5	
C	0	1	3	5	~~7~~ 6	

j = 1

$B[C[3]] = A[1]$

$B[3] = 3$

$C[A[1]] = C[A[1]] - 1$

$C[3] = 3 - 1$

$= 2$

	1	2	3	4	5	6	7	8	
B	1	2	3	3	3	4	5	5	

	1	2	3	4	5	
C	0	1	~~3~~ 2	5	6	

NOTEBOOK