Subject Algorithms (6) الخوارزميات * 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 (dyle) 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 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

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 = ~~0~~ 1
j = 4

i	j	sorted
~~0~~ 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~~	~~X~~
	~~3~~	~~X~~
	2	F
	1	F

(1, 2, 5, 4, 6)

---

* Phase 3 :- i = 2

j = 4

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

0 1 2 3 4

(1, 2, 5, 4, 6)

4 > 6 = F

(1, 2, 5, 4, 6)

5 > 4 = ~~F~~ T

(1, 2, ~~5~~ 4, 5, 6)

---

* Phase 4 :- i = 3

i	j	sorted
3	~~4~~	~~F~~
	3	T

(1, 2, 4, 5, 6)

5 > 6 = F

---

* Phase 5 :- i = 4

i	j	sorted
4	4	F

~~4~~ — i < last

4 < 4 ⇒ F

breaks;

NOTEBOOK

Subject Date No.

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

Bubble sort
- Best when The ~~Arr~~ 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 ↑ حالة سيئة

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

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

insertion sort

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

↳ why?

* The selection is Alwayse O(N²) ⇒ So it's Excluded (مستبعد)

* ~~The insertion~~

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

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

NOTEBOOK

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, Sach 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

Subject Date No.

* Divide And conquer

- Divide تقسيم
  - How to divide The Problem?
  - Keep continued on The same dividing Technique in The First divide
- conquer إيجاد حل → what is The solution?
- combine جمع الحلول

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

* All ~~Hard~~ works is done in The Hard section

- Merge sort
  - Easy Division
  - Hard combination
  - why? ⇒ سؤال في الفاينال
- 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

↳ Als, 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.

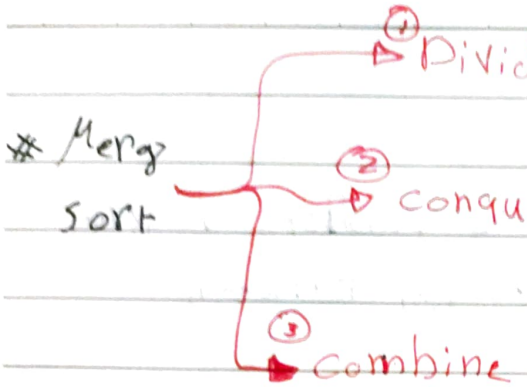
③ using ONLY The Merg sort
↳ which is based on divide And conquer :) Loopback

↳ keep in Mind Two Things :-

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

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

Subject Date No.



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

(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

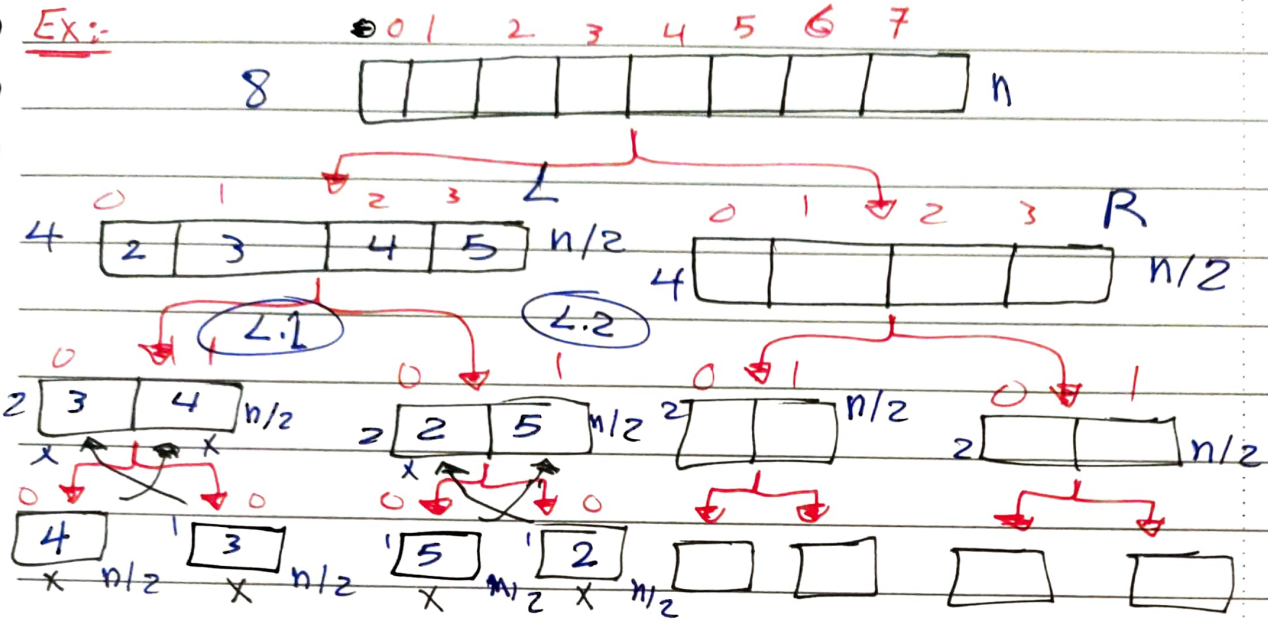
~~→ We combine Recursir~~

→ 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.

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

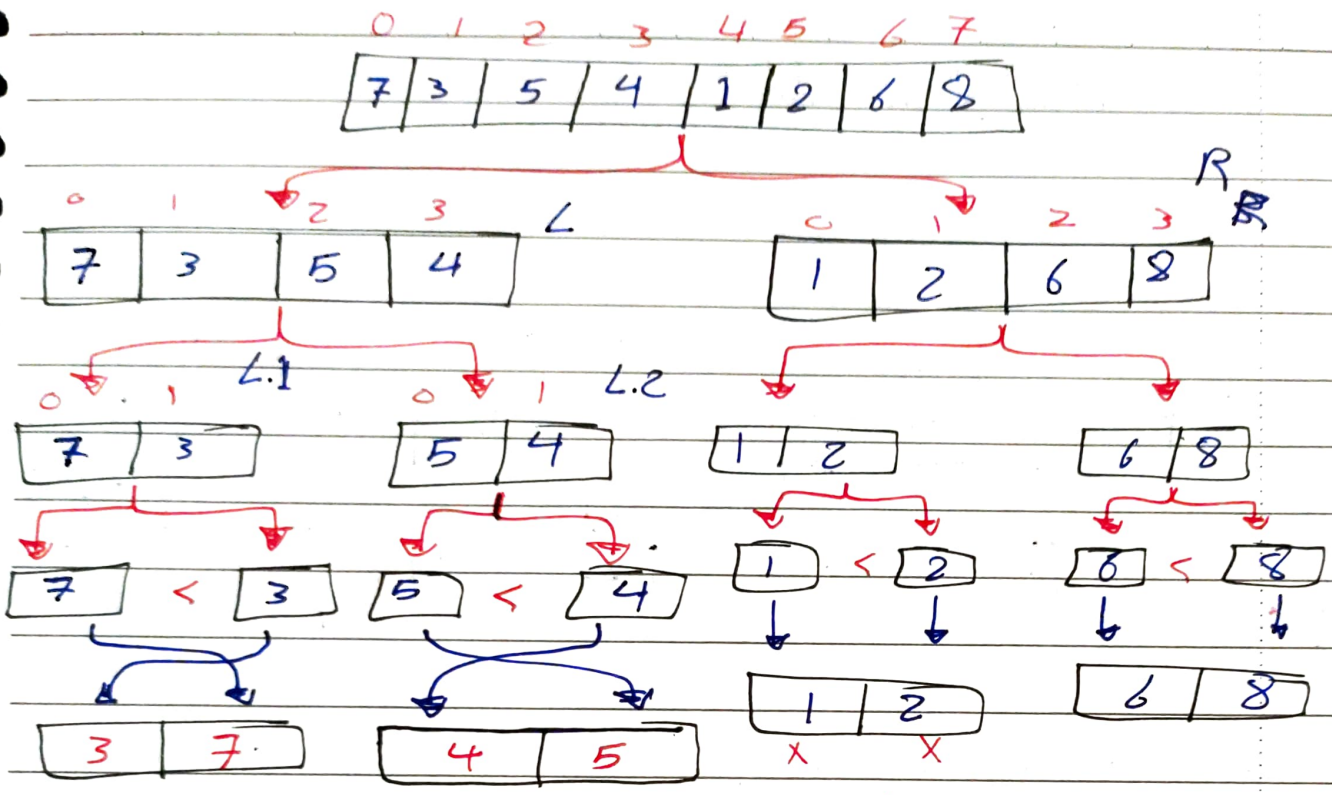
③ $3 < 2 = F$

④ $3 < 5 = T$

⑤ $4 < 5 = T$

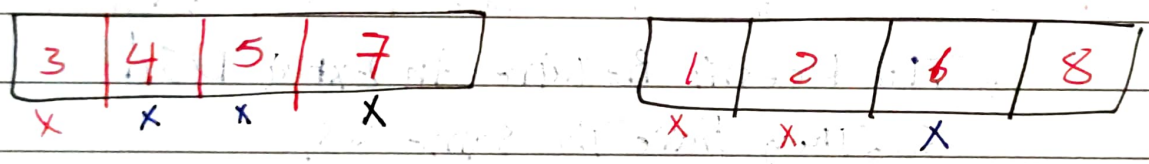
⑥ nothing To compare with 5 so it's Max one, And if there is More Elements will be The Same

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



3<4, 7>4, 7<5,

1<6, 1<8, 2<6, 2<8



x → 3<1, 3<2, 3<6

x → 4<6, 5<6, 7<6

x → 7<8, 8

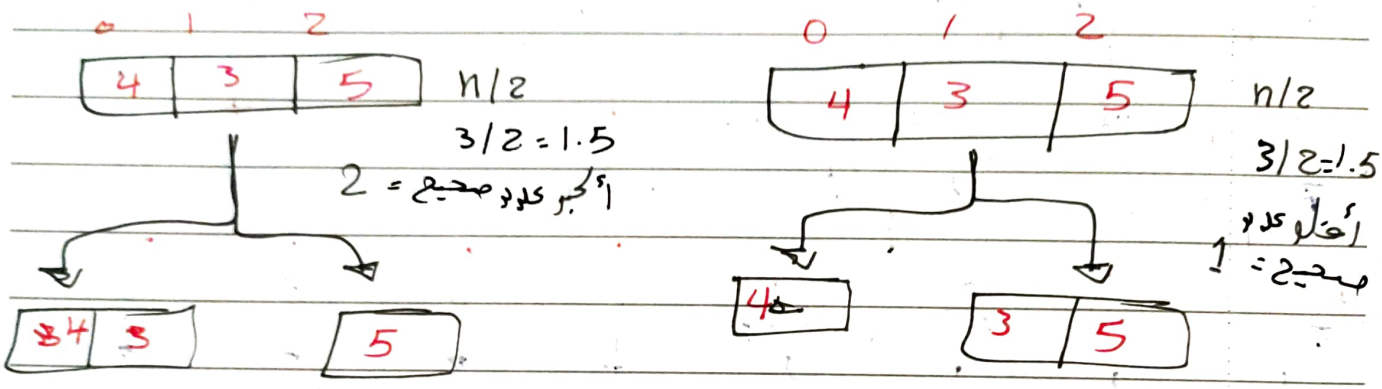
1	2	3	4	5	6	7	8

هاد أول خطوة combine و يعني اللي حكينا DSA زيادة وإنهاء و التوصيف

Subject Date No.

* what if The "n" is odd?

Ex:-



↳ Are They The same?

Yes, How or Why?

↳ Bec They Have The same cost? ⇒ How?

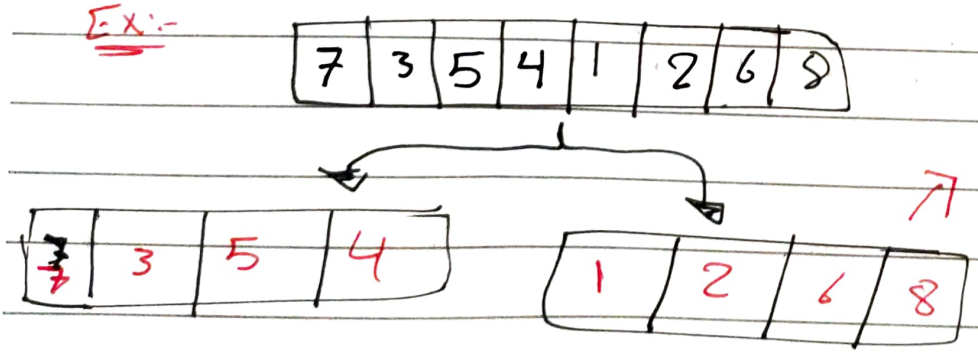
↳ The Poesrit Require An Extra DSA

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:-



Subject Date No.

↳ No, you Have to divide Them ⇒ Why?
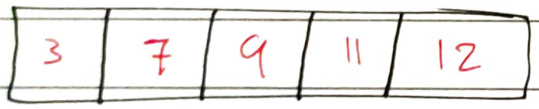
*For Two Main Reasons:

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.

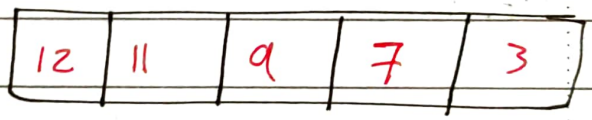
* Case Study:

(1)



↳ 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)



↳ what is The Big-O?
① Ω(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 هو الجواب في الاختبار
أو في الاختبار ← الجواب هو الجواب في الاختبار

← الجواب: ندور على سؤال يكون للاختبار

*Notes:

① ← أنت غير مضطر تكتب Pseudocode وإذا كتبت وكان في أخطاء برمجية يوقف تحاسب عليها
← من أكتب Pseudocode إذا كانت الفكرة كاملة
إذا كانت الفكرة كاملة كتابة الـ code أسهل
مثال: أكتب من الشرح
خوارزمية لتحزين
أسهل

② ← يمكن أخلي عمل الـ swap في الـ shift بدل ما أعمل swap في الـ Selection ؟ وحيث خوارزمية أخرى + عمل swap في الـ Merge sort

NOTEBOOK

Subject Algorithms (8) المحاضرة # Date No.

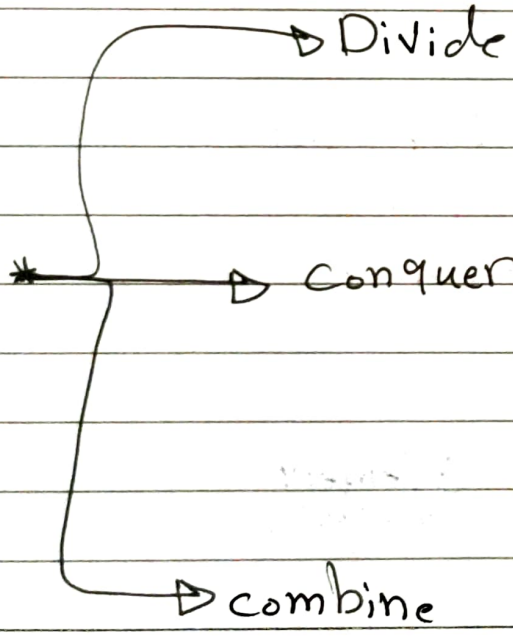
*Quick Sort:

↳ 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 The list By using n/2 => and That was constant for all cases

↳ The divide is The Hard work 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	<

→ 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 or Greater Than Pivot?

↳ we Have to iterate on The Array to know That

↳ How much it takes?

↳ O(N).

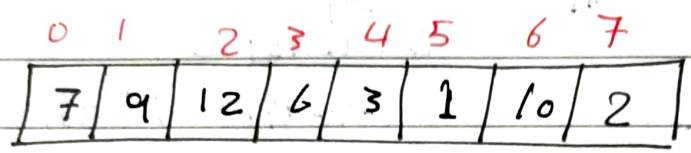
↳ Is that cost will be added on The Algorithm or it will be Embedded in The Algorithm?

↳ 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

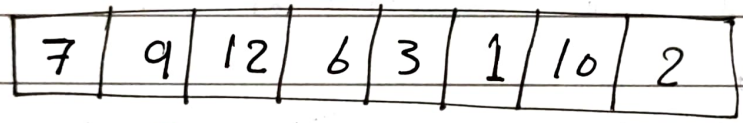
Tracing: suppose The following Numbers To sort Them
(7, 9, 12, 6, 3, 1, 10, 2)



pivot = arr[0]
= 7

first ≤ 7 = T first++
7 ≤ 7

last > 7



first ≤ 7

last > 7

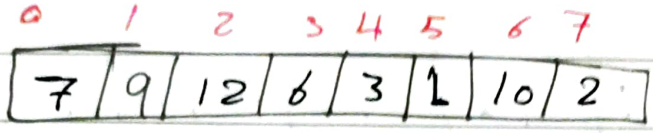
* 9 ≤ 7 = F

↳ 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

Subject Date No.



9 5 7 = F

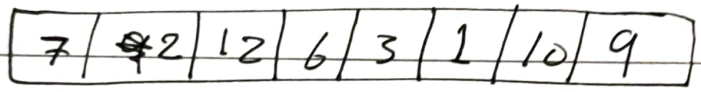
last > 7

~~last~~ 2 > 7 = F

swap(9, 2)

swap(arr[1], arr[7])

first ++, last --;



first < 7

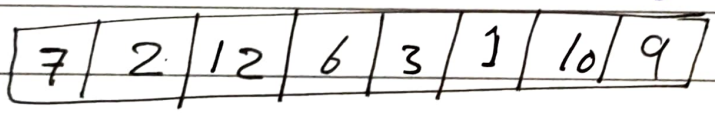
12 < 7 = F

first ++

last > 7

10 > 7 = T

last --;



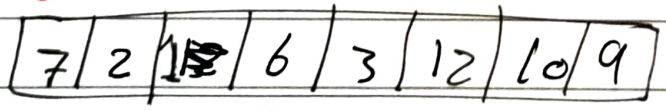
12 < 7 = F

first ++

1 > 7 = F

last --;

swap(12, 1)



6 < 7 = T

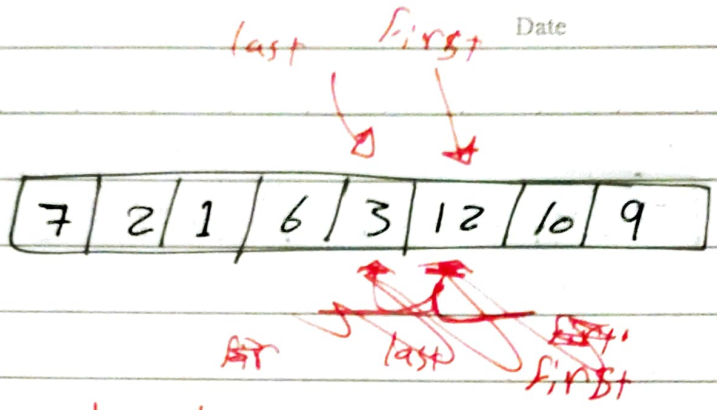
first ++

3 > 7 = F

3 < 7 = T

first ++

Subject Date Noon: 29 Min

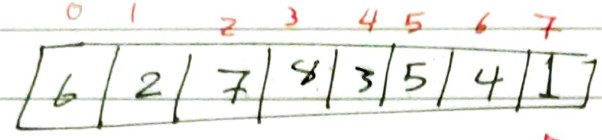


breaks:

→ if one of the Two Pointers Exceeds (i > j)
The Another one we break

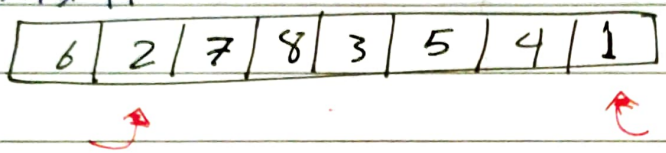
Subject Date No.

Tracing 2:



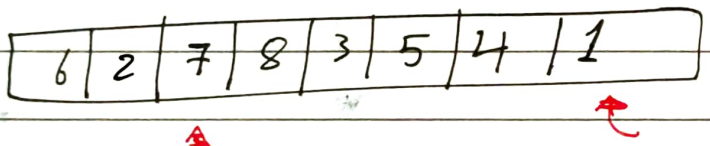
$6 \leq 6 = T$

first ++



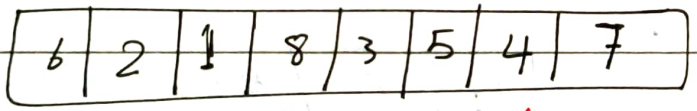
$2 \leq 6 = T$

first ++



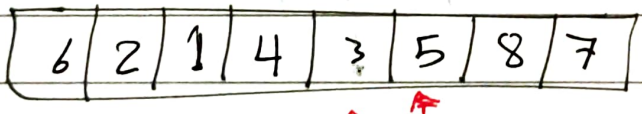
$7 \leq 6 = F$ swap (7, 1) $1 > 6 = F$ last --

first ++



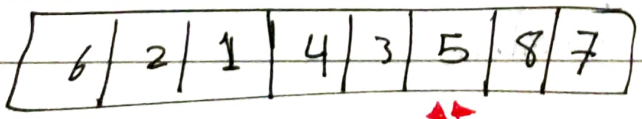
$8 \leq 6 = F$ $4 > 6 = F$

first ++ swap (8, 4) last --

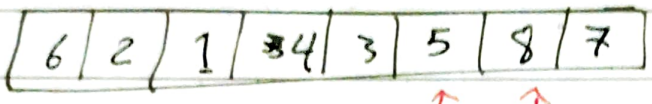


$3 \leq 6 = T$

first ++

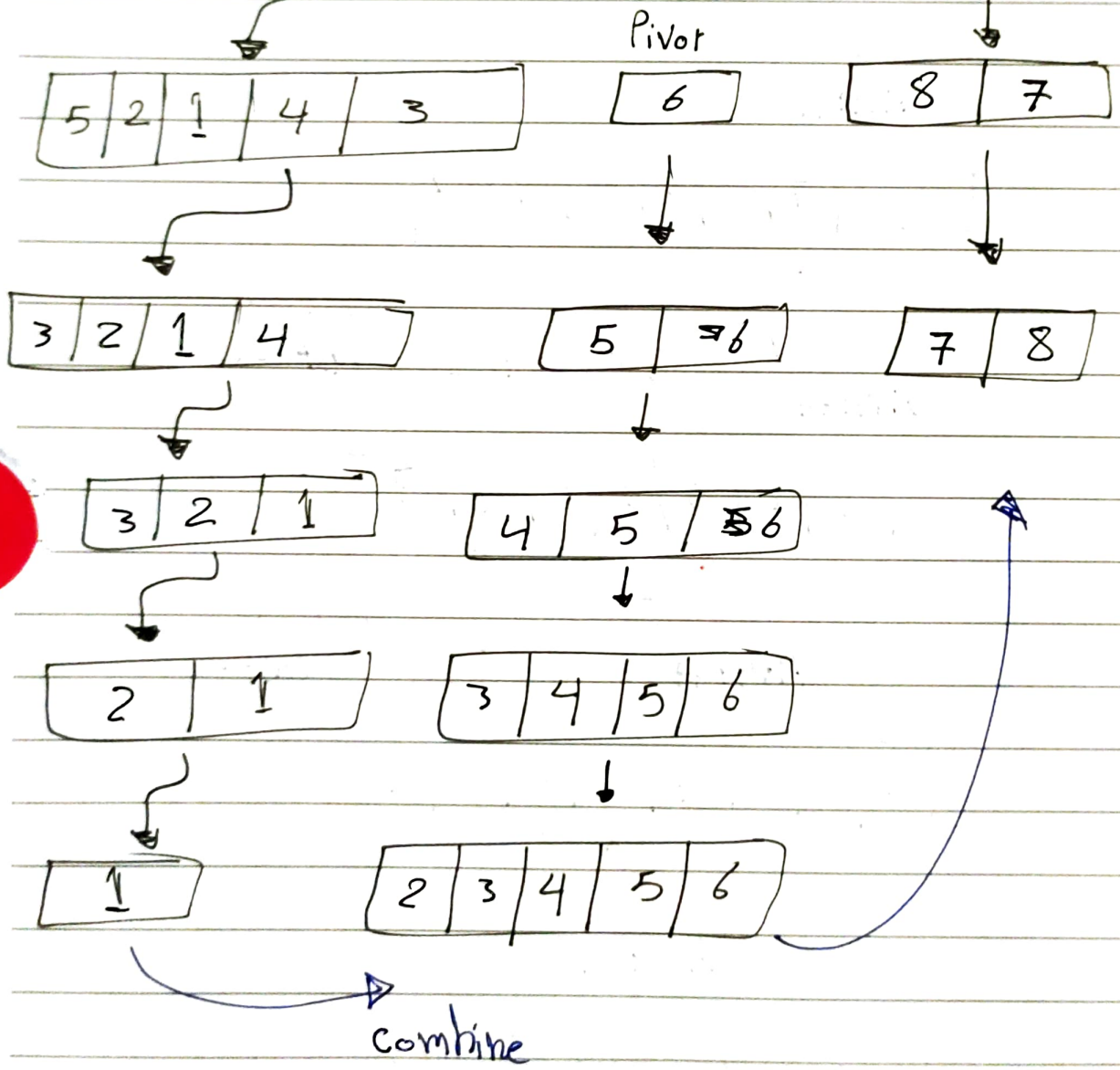


$5 \leq 6 = T$



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

breaks



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 Quicks 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

Subject Date No.

↳ if we Don't Have information about The Data And - we choose The Quick sort we May Have The 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 frequently Marks 35 for Example :).

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

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

Subject Algorithms (9) Date No.

* Binary Search Tree :-

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

→ Binary →

* True or false
* Two choices only

→ Tree :- The Tree is slecial type of The Graph, whats 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

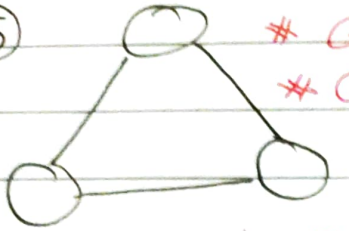
(2)  Graph Node
Isolated ~~Graph~~; The Most Essential
Part of The ~~Node~~ Graph
~~It's Exit~~ ~~Box~~ Exisits

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

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

an Edge

(5)



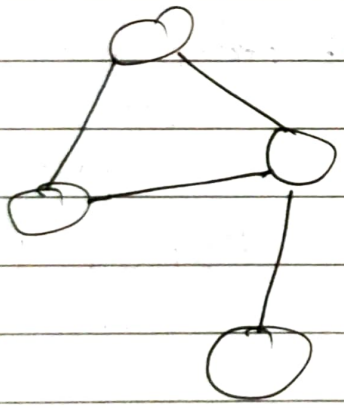
* Graph
* cycle Graph
  ↳ why? If we start in Any Node will back to it (The Start and The End Node Are The same)
* undirected Graph

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

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

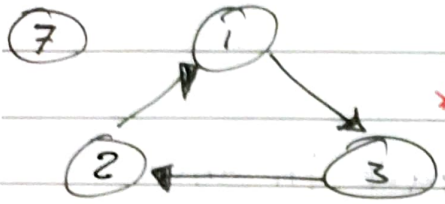
---

(6)



* Graph
* Undirected Graph
* Not cycle Graph

But, it is a Graph Has a cycle



* Graph

* Directed Graph

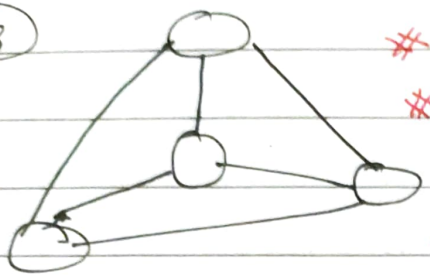
↳ If we Have Directed Graph we Have Two chocises

1. Strongly Connected
2. Weakly Connected

* This Graph is Weakly connected

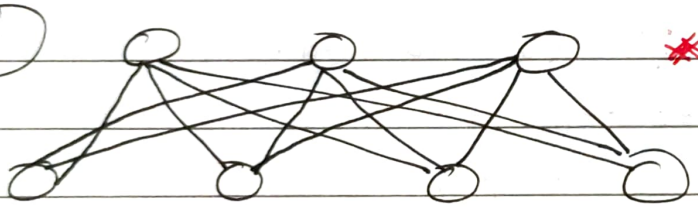
↳ Graph
- Not connected
- connected
  - Undirected → Always strongly connected
  - Directed
    - Strongly Connected
    - Weakly Connected

(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)



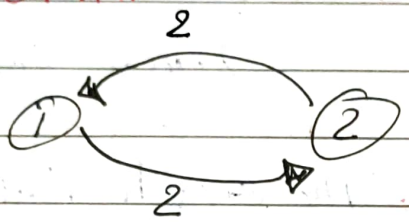
* ~~Bipartite Graph~~
  ~~we can't~~
* complete Bipartite Graph
  ↳ The Graph is divided into Two Parts
  ~~↳ By using The coloring system~~
  ↳ The Graph is divided By using The coloring System

Subject Date No.



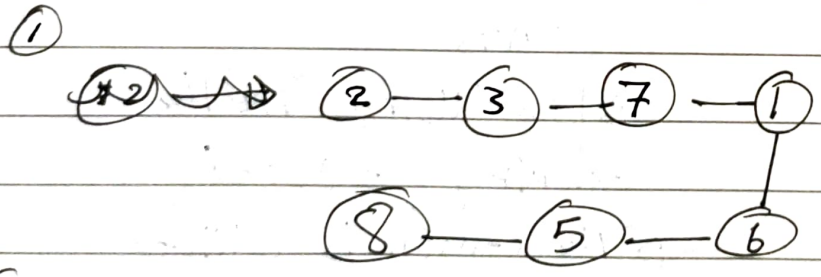
* Bidirectional
* Bidirectional Graph

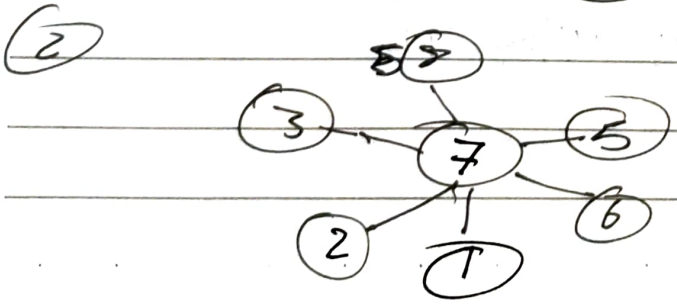
* Note The Bidirectional Graph is diffrent from This Graph



⇒ Why? The Example of The Two computers in The Network
If A, B are computers in The Network and The connection prolled 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 :)

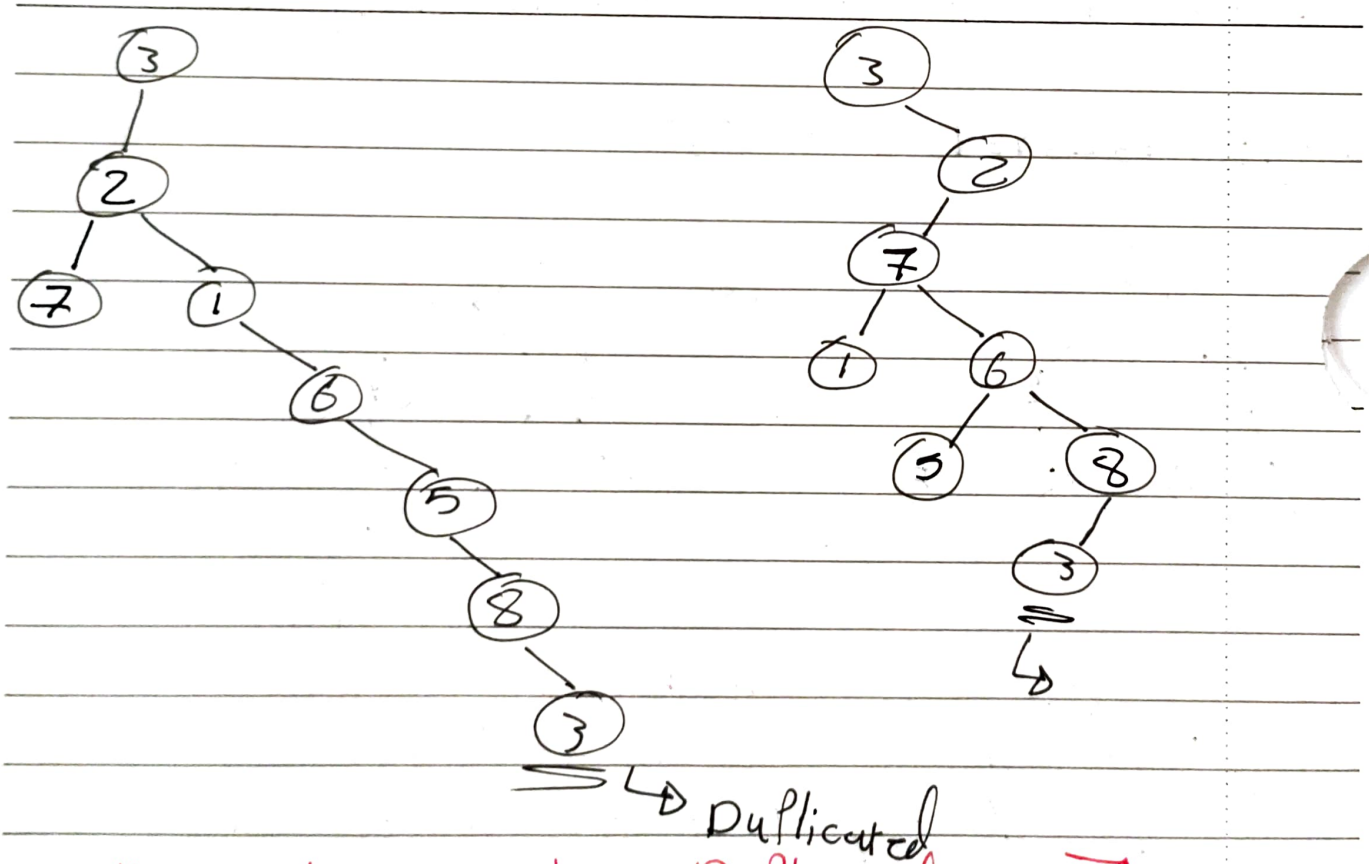
Subject Date No.

→ 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

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

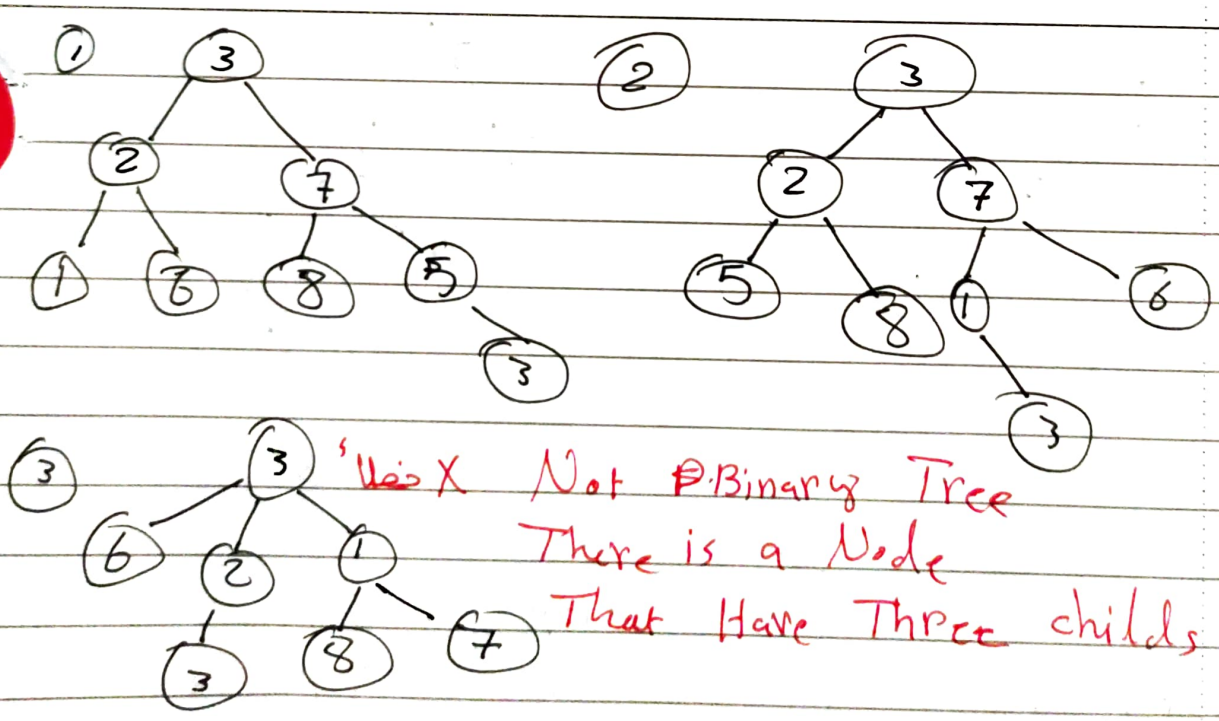
*Note:- We Make unlimited Number of Trees from a dataset

* 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)



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

Subject Date No.

Binary Tree ↖ ↗ Binary search Tree

* How to convert The BT to BST?

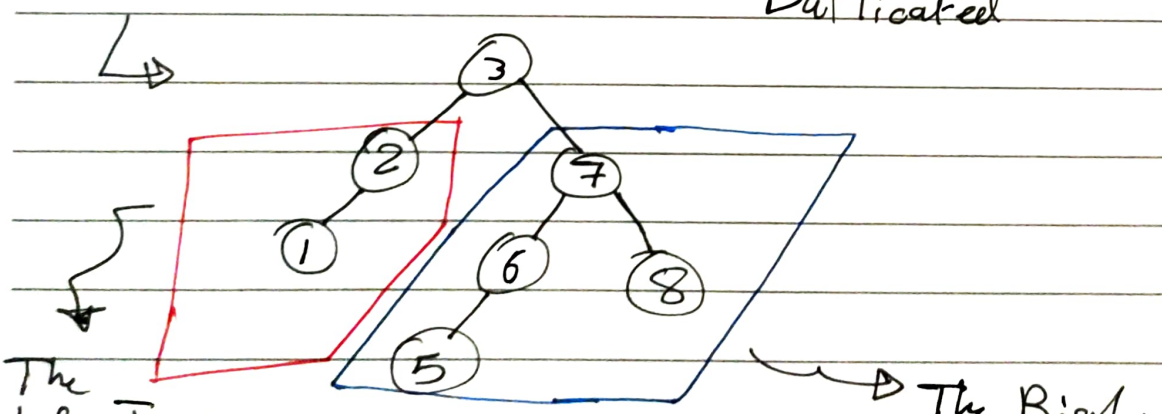
① No Duplicated Values in BST.
Remove

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) 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

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
- ~~Str~~ 2-3 pages
- The first page should contain
  - The semster
  - The University Name
  - Your Name
  - Your ID
  - The Project Name
- The seconde Page
  - ↳ contains The ~~Tabel~~ Table of Contents
- The Third Page
  - ↳ contains References
    - Min 3 references
    - The following ARE Not ~~r~~ 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~~ No. best Algorithm for sorting" ← No Reference is Mentioned

-0.25 -1 -0.25 -1 -0.25 -1 -0.25 -1

~~to~~ 4

NOTEBOOK

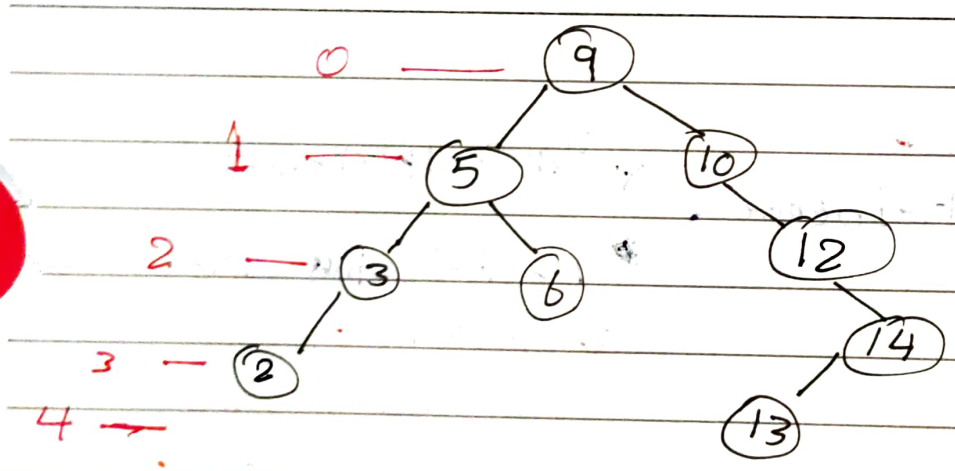
~~The Editor~~

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

* The BST :-

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

→ consider The Following ~~Free~~ 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 numbe

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

↳ 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.

Subject Date No.

~~Q4 B~~

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

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

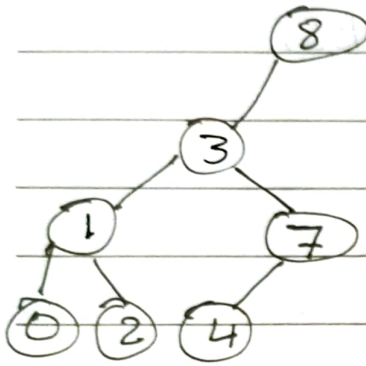
Level = Height = Depth.

Subject Algorithms (11) Date No.

* Operations on BST:-

⇒ Build a BST from the following data

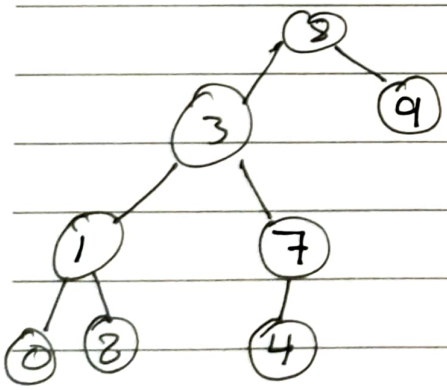
[8, 3, 7, 4, 1, 2, 0]



* 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 BST 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

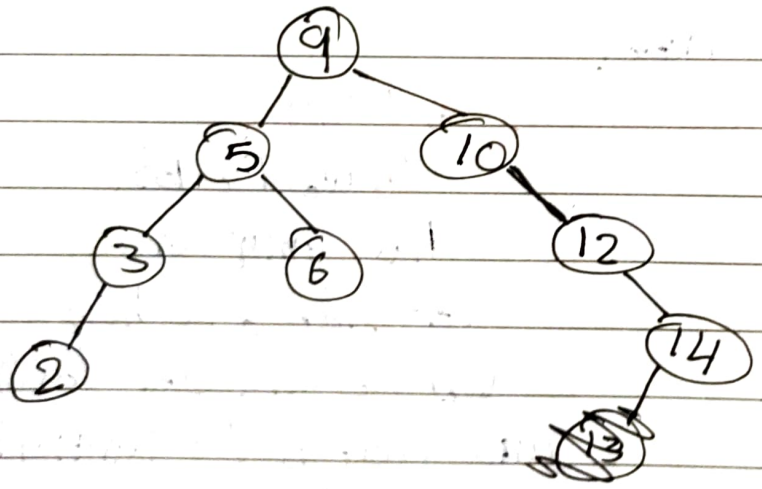
Subject Date No.

* 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.



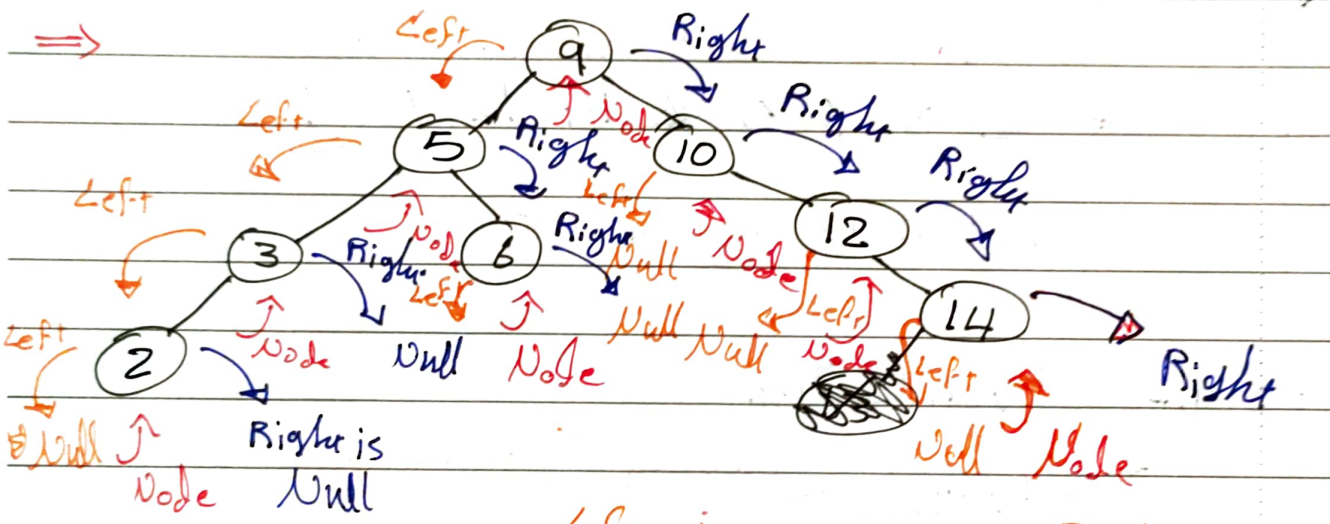
NOTEBOOK

Subject Date No.

→ in-order Traverse :-

↳ LNR (Left - Node - Right)

* Visit the Left Node
* Then The ~~Root~~ or the Parent of the Node.
* ~~At~~ Then The Right Node of the Parent
* Apply this ~~Each~~ order on The Rest 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 ~~B~~ 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 (~~ascending~~)
* 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)

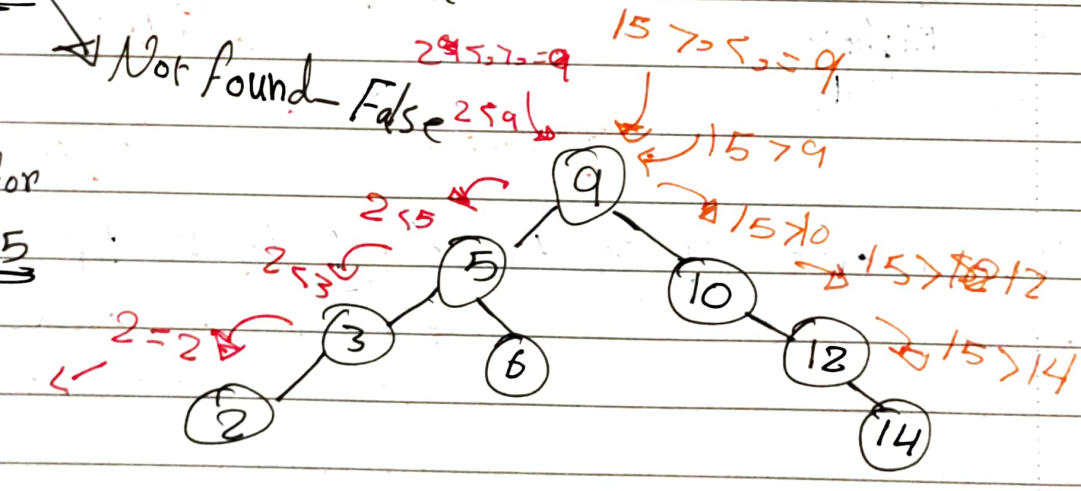
Post-order :- 2 3 6 5 14 12 10 9

↳ The Root is The Last in order

* Search → found_True

→ Not found_False

search 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 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

NOTEBOOK

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

Ok, Let's 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~~
~~Let 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

Subject Date No.

③ Two childs:

To

⇒ ~~The~~ Get The:

* Max ⇒ Go left while you get Null
* Min ⇒ Go Right while you get Null

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

① starting from a Node

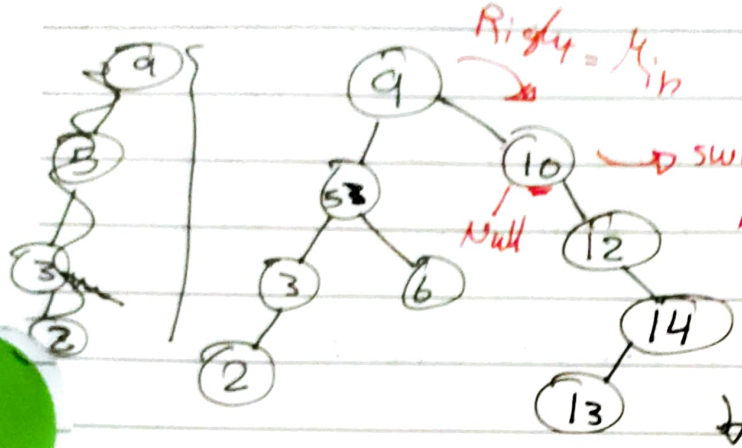
② will Go Either:
- left → Max Element in The left of The Node.
- Right → choose The Min Element in Right of The Node

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

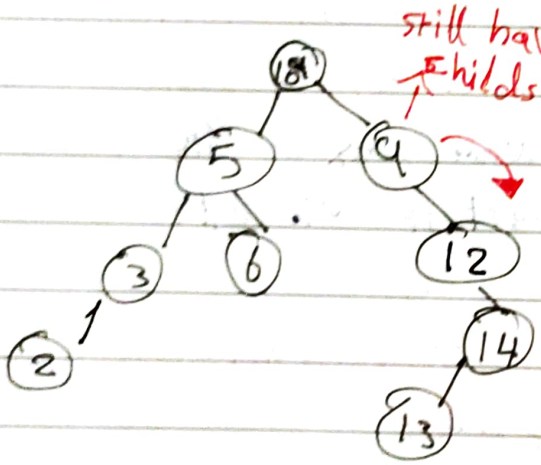
↳ Why? To convert The Node from Having Two child to ~~Do childs~~ One child

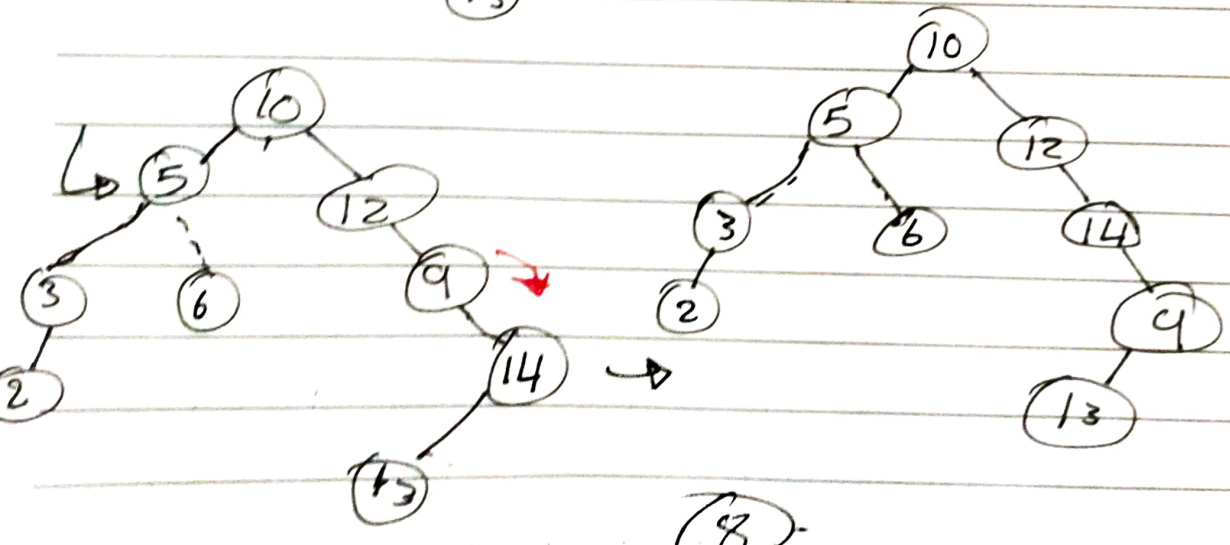
Subject Date No

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

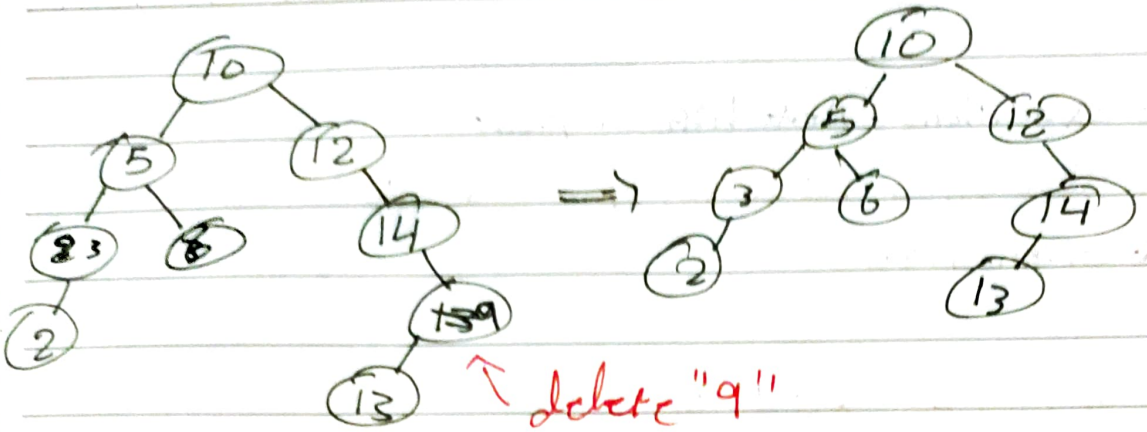


If we ~~choose~~ choose The Right-subtree Then we have to choose The Min Element ⇒ How?
↳ By Going to left of The subtree and so on.





NOTEBOOK



✱ 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.

⇒ Profosal 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 Travesal is $O(N)$

Subject Date No.

(2) search :-

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

* ~~No~~ * 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~~ to convert it To BST what is The Root?

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

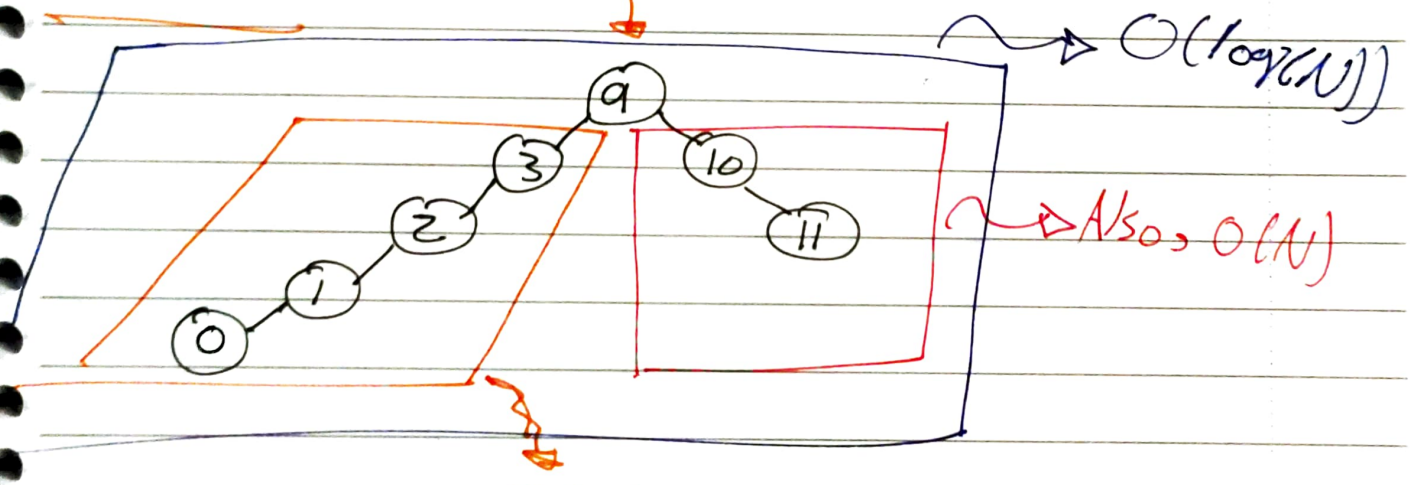
Subject Date No.

⟹ 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

NOTEBOOK