Assignment - 1.

Q.1) What is Algorithm?

→ Algorithm is a step-by-step process to perform some action or solve a problem.

Q.2) Define Properties of algorithms.

→ The Properties of algorithms are as follows -

i) input - In this algorithm uses values from a specific set in ℰ.

ii) output - For each input the algorithm produces a values from specific task. Every input has an output.

iii) Precision - In this steps are precisely defined.

iv) Correctness - Input is defined for that output is correct and desired.

v) Finiteness - In this Output after finite number of steps for each input.

vi) Determination - The Result should be guaranteed.

vii) Generality - Procedure applies to all problems not a special subset.

Q.3) Define Time Complexity.

→ It defines that how much time it consumes to execute the program is known as Time Complexity.

Q.4) Define Space Complexity.

→ It defines that the amount of memory space required for an algorithm or program during the execution is known as space complexity.

Q.5) What is Binary Search.

→ Binary Search is an efficient algorithm for searching in a sorted array.

Q.6) Sort the following numbers using quick sort.

→ 50 31 71 38 77 81 12 33.

Quick Sort Algorithm -

pivot

50 31 71 | 38 | 77 81 12 33

pivot = < pivot > = pivot.

12 31 33 | 38 | 77 81 50 71

pivot pivot

12 31 33 38 | 50 77 81 71

pivot pivot

12 31 33 38 50 77 71 81

Sorted -

12 31 33 38 50 71 77 81.

Q.7) Explain Binary Search and divide and conquer with example.

→ Binary search is a searching algorithm used in a sorted array. It repeatedly divides the search interval in half, efficiently narrowing down the search space. Divide and conquer is a problem-solving technique that splits a complex problem into smaller, similiar subproblem. Then it combines the solutions of these problems to deduce the final answer.

eg. —

0	1	2	3	4	5	6	7	8	9
5	7	9	13	32	33	42	54	56	88

key = 33

$mid = \frac{(start + end)}{2} = \frac{0+9}{2} = \frac{9}{2} = 4$ → mid value

search key > A[mid] skip - 0 → 4

33 > 32 greater

greater = mid + 1 (R.S)

start = mid + 1

5	6	7	8	9
33	42	54	56	88

$mid = \frac{5+9}{2} = \frac{14}{2} = 7$ → mid value.

33 < A[mid] skip - 7 → 9

33 < 7 lesser = mid - 1 (L.S).

less

start = mid = 1

5	6
33	42

mid = $\frac{5+6}{2} = \frac{11}{2} = 5$ → mid value

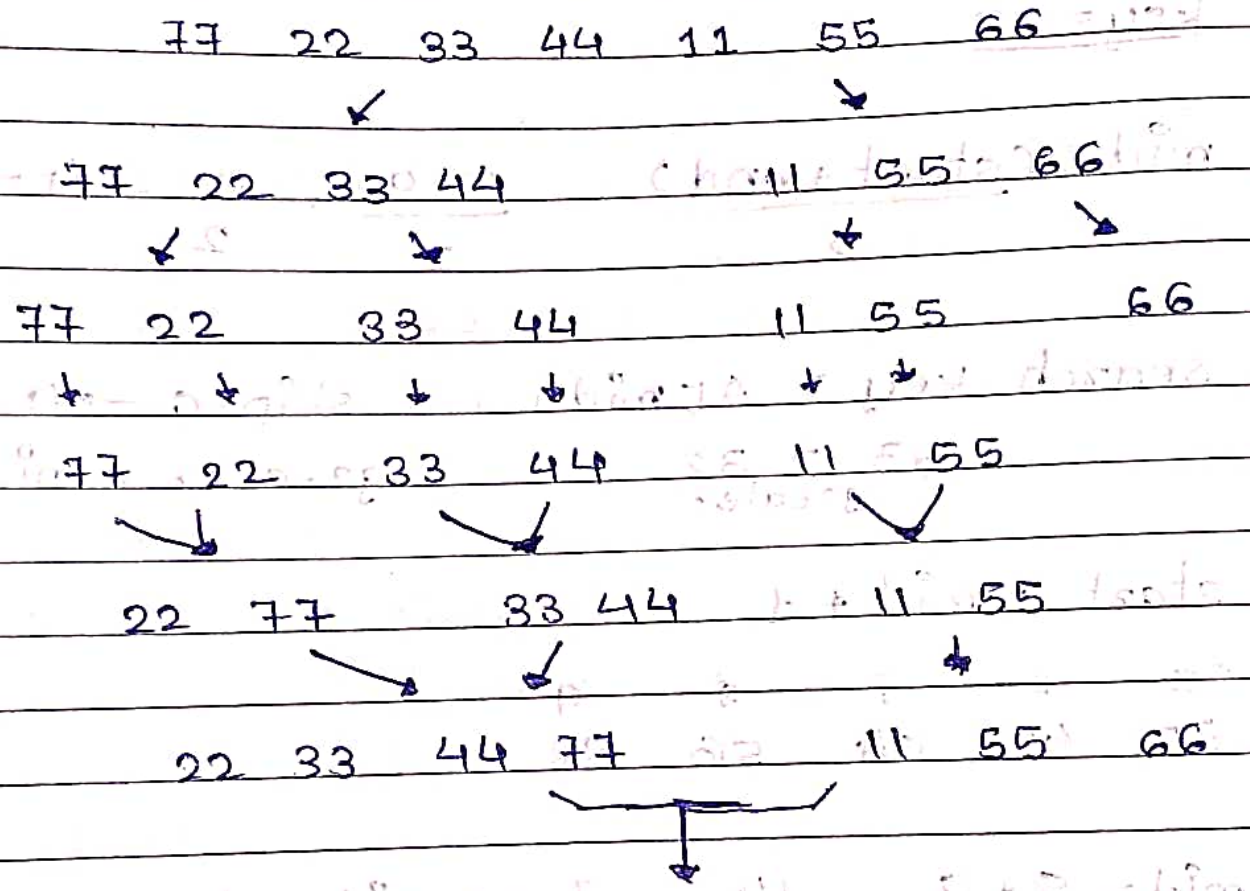
33 = A[mid]

33 = 33

It is found at index 5.

8) Simulate merge on data sequence.
77 22 33 44 11 55 66

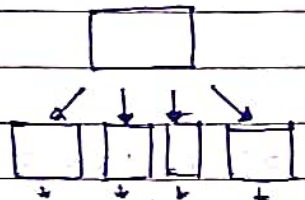
→



11 22 33 44 55 66 77

Q.9) Explain Dynamic Programming with example. (chain matrix).

→ It is a strategy for designing algorithm which is used when problems, breaks down into recursing small problems.



values.

In such problems there can be many solution. Each solution has a values.

ex. chain matrix multiplication.

A = 10 × 30
B = 30 × 5
C = 5 × 60
D = 60 × 8

m[1,1] m[2,2] m[3,3] m[4,4]
A B C D

m[1,2] m[2,3] m[3,4]
A · B B · C C · D
10×30 30×5 30×5 5×60 5×60 60×8
10 × 30 × 5 30×5×60 5×60×8
1500 9000 2400

m[1,3]
A·(B·C)
10×30 30×5 5×60
m[1,1] + m[2,3] + 10×30×60
cost of A + cost of B·C + cost of A·(B·C).
= 0 + 9000 + 18000
= 27000

(A·B)·C
m[1,2] + m[3,3] + 10×5×60
= 1500 + 0 + 3000
= 4500

m[1,3] = 27000

m[2,4]
B·(C·D)
30×5 5×60 60×8
m[2,2] + m[3,4] + 30×5×8
= 0 + 2400 + 1200
= 3600

(B·C)·D
30×5 5×60 60×8
m[2,3] + m[4,4] + 30×60×8
= 9000 + 0 + 1440
= 10,440

m[1,4]
= (A·B·C·D)
= (A·B)·(C·D)
= (A·B·C)·D

$= \min \{ m(1,1) + m(2,4) + 10 \times 30 \times 8,$
$m(1,2) + m(3,4) + 10 \times 5 \times 8,$
$m(1,3) + m(4,4) + 10 \times 6 \times 8 \}$

$= \min \{ 0 + 3600 + 2400,$
$1500 + 2400 + 400,$
$27000 + 0 + 4800 \}$

$= \min \{ 6000, 4300, 31800 \}$

$= 31800$

PLACE function determines the position of the queen in $O(n)$ time. This function is called n times.

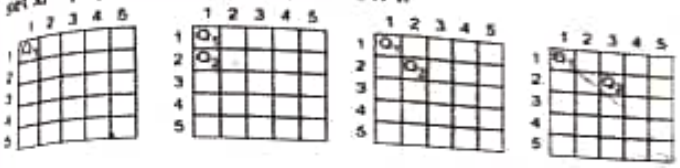
Thus, the recurrence of n-Queen problem is defined as, $T(n) = n*T(n-1) + n^2$. Solution to recurrence would be $O(n!)$.

Ex. 6.1.1 : Find all possible solutions for five queen problem using backtracking approach.

Soln. :

Solution of N queen problem is represented using n-tuple $X = [x_1, x_2, x_3, \ldots, x_n]$. Each $x_i = 1, 2, \ldots, n$.

If queen Q_i can be placed successfully in column j, then set $x_i = j$. Q_i is always placed in row i.

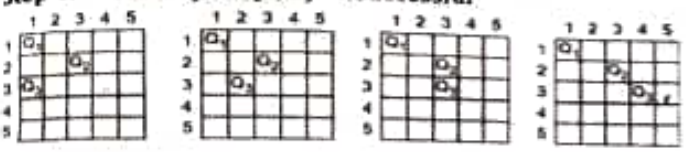


Step 1: Place Q_1 on (1, 1) → Successful

Step 2: Place Q_2 on (2, 1) → Fail → Backtrack

Step 3: Place Q_2 on (2, 2) → Fail → Backtrack

Step 4: Place Q_2 on (2, 3) → Successful

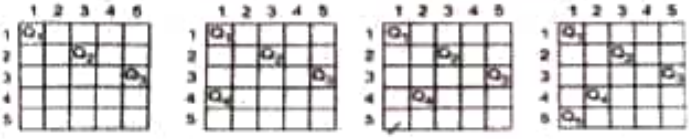


Step 5: Place Q_3 on (3, 1) → Fail → Backtrack

Step 6: Place Q_3 on (3, 2) → Fail → Backtrack

Step 7: Place Q_3 on (3, 3) → Fail → Backtrack

Step 8: Place Q_3 on (3, 4) → Fail → Backtrack

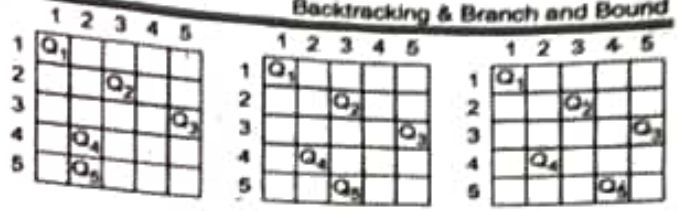


Step 9: Place Q_3 on (3, 5) → Successful

Step 10: Place Q_4 on (4, 1) → Fail → Backtrack

Step 11: Place Q_4 on (4, 2) → Successful

Step 12: Place Q_5 on (5, 1) → Fail → Backtrack



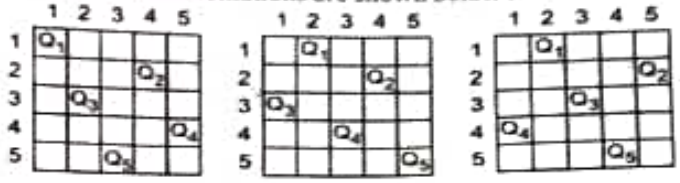
Step 13: Place Q_5 on (5, 2) → Fail → Backtrack

Step 14: Place Q_5 on (5, 3) → Fail → Backtrack

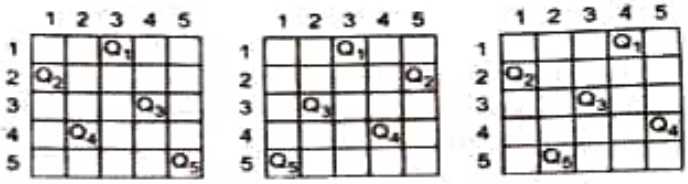
Step 15: Place Q_5 on (5, 4) → Successful

Thus, the solution of this instance is (1, 3, 5, 2, 4).

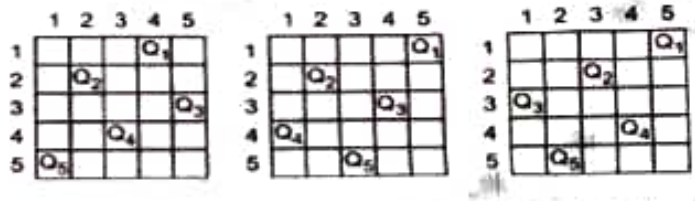
Few more combinations are shown below :



Solution: (1, 4, 2, 5, 3) **Solution:** (2, 4, 1, 3, 5) **Solution:** (2, 5, 3, 1, 4)



Solution: (3, 1, 4, 2, 5) Solution: (3, 5, 2, 4, 1) Solution: (4, 1, 3, 5, 2)



Solution :(4, 2, 5, 3, 1) Solution: (5, 2, 4, 1, 3) Solution: (5, 3, 1, 4, 2)

Ex. 6.1.2 : Current configuration is (6, 4, 7, 1) for 8-queens problem. Find answer tuple.

Soln. :

– Tuple (6, 4, 7, 1) indicates the first queen is in the 6th column, the second queen is in the 4th column, the third queen is in the 7th column and forth queen is in the 1st column. Given arrangement of queen is,

Assignment - 3
Unit - II

Date
Page

1) Define disjoint set.

→ These are sets of data structure which supports 3 operations make set, union and find set.
It can be defined as the subsets where there is no common element between the two sets.
Ex. We have 2 subsets S1 and S2.
S1 contains the element 1, 2, 3, 4
S2 contains 5, 6, 7, 8
There is no common element betwee 2 sets.

2) Define union.

→ Union operation is to take 2 different sets and merge them into 1 set.
Union of set A and B is defined to be the set of all those elements which belong to A or B or both and is denoted by AUB.
Let $A = \{1, 2, 3\}$ $B = \{3, 4, 5, 6\}$
$AUB = \{1, 2, 3, 4, 5, 6\}$

3) Define union by Rank.

→ To ensure that when we combine two trees, we try to keep the overall depth of the resulting tree small.
This technique used to optimize the union operation by ensuring that the smaller tree is always attached to the root of the larger tree.
This approach prevents the trees from becoming imbalanced, which would lead to inefficient find operations.

4) What are applications of disjoint set
→ 1) Hashing functions, union find algorithms
2) stack operations
3) Heap operations, cycle detection in graphs
4) pushing & popping values
5) Job sequencing problem solving
6) Kruskal's algorithm, computer networks

5) Define lower bound theory.
→ Lower bound theory says that no algorithm can do the job in fewer than L(n) times units for arbitrary inputs
Calculation of minimum time that is required to execute an algorithm is known as a lower bound theory.
It uses a number of methods to find out the lower bound
① Comparison trees
② decision tree

6) Explain disjoint set with one example
→ These are the sets of data structure which supports 3 operations make set, union and find set.
1) Make set is the operation to create a set with only one ~~union~~ element.
2) Union operation is to take 2 different sets and merge them into 1 set.
3) Find set is an operation to return an identity of set which is usually an element

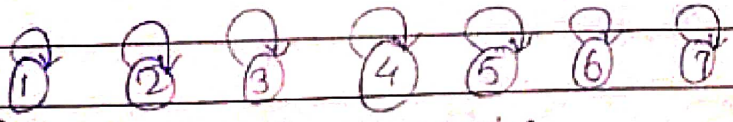
in set which acts as representative of that set

characteristics —

① It keeps a set partitioned into disjoint subsets.
② It allows the efficient union of two subsets
③ It makes it possible to quickly determine a given element belongs to which subset.

- In the disjoint set each element in a set is represented by a unique root node.
- In the disjoint set, two elements belong to the same set if they share the same root node
- The root node of an element can be found by following the parent pointers until a node is reached that has itself as its parent

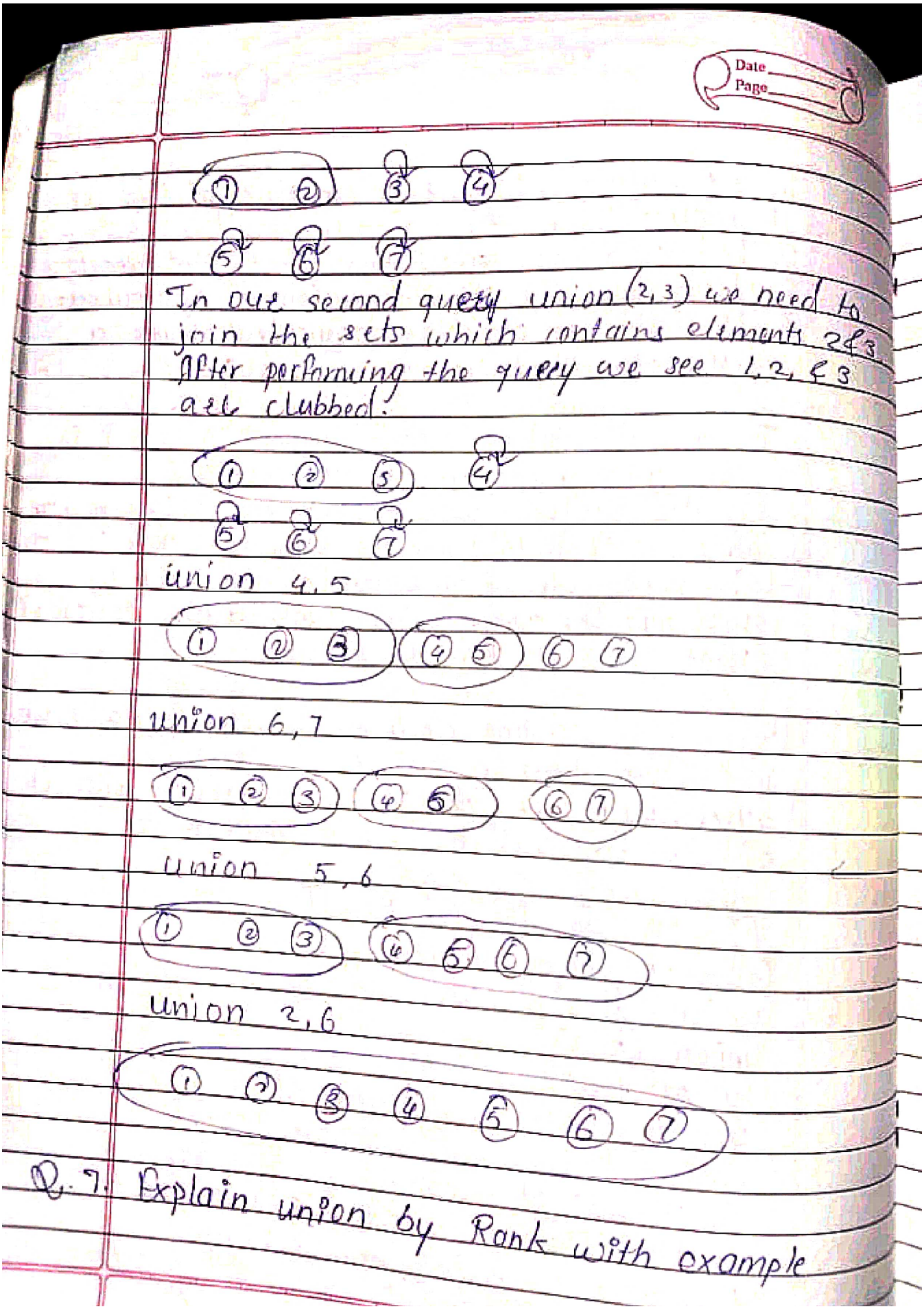
Ex. Let assume we have 7 nodes initially we will store them in the form of trees. where each tree corresponds to one set and root of the tree will be parent/leader of set.



Following union queeies.

union (1, 2)
union (2, 3)
union (4, 5)
union (6, 7)
union (5, 6)
union (2, 6)

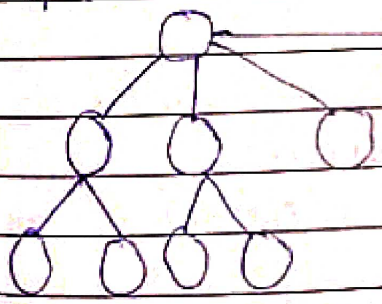
In first queey union (1,2) we need to join two sets i.e into one.


Date
Page
In our second query union (2,3) we need to join the sets which contains elements 2 & 3 After performing the query we see 1, 2, & 3 are clubbed!
union 4, 5
union 6, 7
union 5, 6
union 2, 6
Q. 7. Explain union by Rank with example

We need a new array of integers called rank[]. The size of this array is the same as the parent array Parent[].
If i is a representative of a set, rank[i] is the height of the tree representing the set.
Now recall that in the union operation it doesn't matter which of the two trees is moved under the other. Now what we want to do is minimize the height of the resulting tree.
If we are uniting two trees (or sets) lets call them left and right then it all depends on the rank of left & rank of right.
If the ranks are equal it doesn't matter which tree goes under the other but the rank of the result will always be one greater than the rank of the trees.

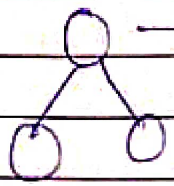
Example :-
Given 2 trees

Tree 1 :-



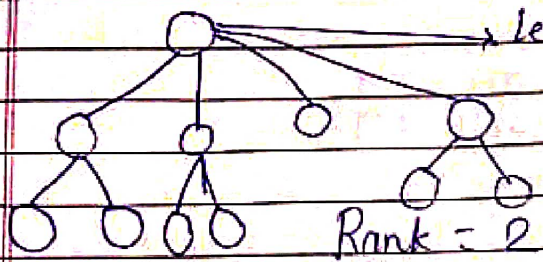
Rank = 2

Tree 2



Rank = 1

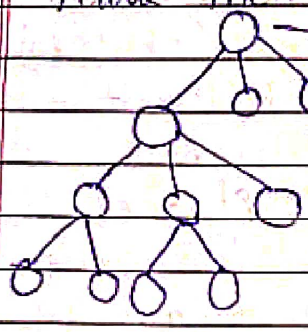
If we combine their rank then rank [tree 1] = 2 and rank [tree 2] = 1 combined.



Rank = 2

We can see that since tree 2 is attached to tree 1 so rank of the combined tree is same as tree 1

If we combine Tree 1 & 2 w/o checking their rank then it would be



Rank = 3

Q.8 Describe briefly lower Bound theory

→ Lower bound theory concept is based upon the calculation of minimum time that is required to execute an algorithm.

It used to calculate minimum number of comparis required to execute an algorithm.

According to lower bound theory for lower boun $L(n)$ of an algorithm.

Once lower bound is calculated then we can compar it with actual complexity of algorithm & if their order are same then we declare it has optimal (best)

Technique used for lower bound theory

Q. 9. Difference between structure and union.

→

Structure	Union
1. To declare a structure the keyword 'struct' is used	To declare a Union the keyword 'union' is used.
2. The compiler allocates the memory for each member when a variable is associated with a structure. The size of a structure is greater or equal to the sum of the sizes of its members	- The compiler allocates the memory ~~for each member when a variable is associated with a structure. The size of a structure is greater or equal to the sum of the sizes of its members~~ by considering the size of the largest member when a variable is associated with union. size of union = size of largest member
3. Each member within structure is assigned a unique storage area of location.	- All the members of the union share the same memory allocated
4. We can initialize multiple variables at a time.	- Only the first data member can be initialized
5. Structure allows accessing and retrieving any data member at a time	- union allows retrieving only one data member at a time.
6. Manipulation of one member of structure wont affect the values of any other member	- Manipulation of one member will affect the other member value in case of union
7. Separate memory for each member	- Shared memory for all members

8. equal to or greater than sum of members sizes.	-Size equal to the largest members size.
9. Multiple members can be accessed simultaneously.	-Only one member can be accessed at a time.
10. changing the value of one member does not affect others.	-change the value of one member affects others.
11. Each member has its own dedicated memory space	-Members share the same memory space.
12. Multiple members can be initialized simultaneously	-Only the first member can be initialized.

Q.10. Explain techniques used for lower bound theory.

→ 1. Comparison trees:

In a comparison sort we use only comparisons between elements to gain order information about an input sequence $(a_1, a_2, \ldots, a_n)$.

Given a_i, a_j from $(a_1, a_2, \ldots, a_n)$ we perform one of the comparison

$a_i < a_j$ less than

$a_i \leq a_j$ less than or equal to

$a_j > a_j$ greater than

$a_i \geq a_j$ greater than or equal to

$a_i = a_j$ equal to

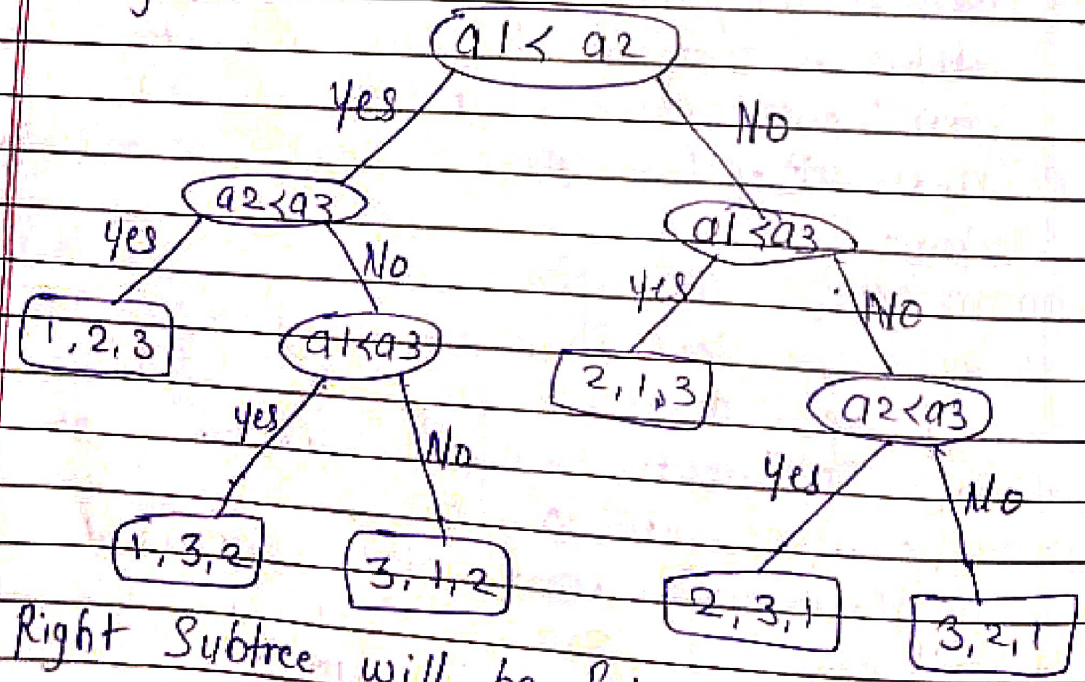
To determine their relative order if we assum all elements are distinct then we just need t consider $a_j \leq a_j$ '=' is excluded $<, \geq, \leq, >$ are equivalent.

Consider sorting three numbers a1, a2 and a3. There are 3! = 6 possible combination.

(a1, a2, a3) , (a1, a3, a2);
(a2, a1, a3) , (a2, a3, a1)
(a3, a1, a2) , (a3, a2, a1)

The comparison based algorithm defines a decision tree.

2] Decision tree – It is a fully binary tree that shows the comparisons between elements that are executed by an appropriate sorting algorithm operating on an input of a given size. Control, data movement, and all other conditions of the algorithm are ignored.
In a decision tree there will be an array of length n.



Right Subtree will be False condition [ai > aj]

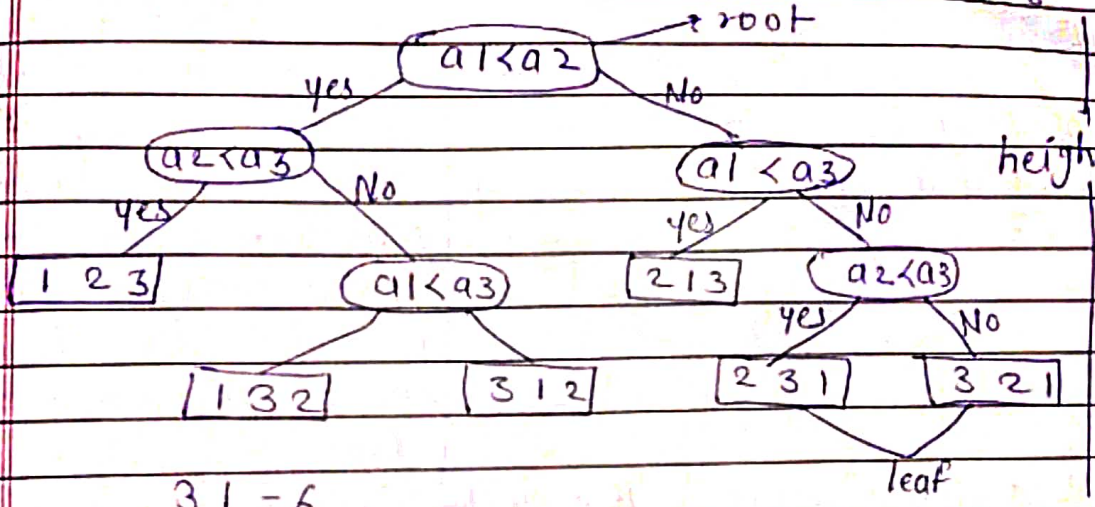
① Comparison trees – are the computational model useful for determining decision tree

② Decision tree – ① Full binary tree that shows comparison between elements that are executed by string

② There will be array of length n so total leaves will be n! (total no. of comparison)

Decision tree

a1 a2 a3
6 7 8



3! = 6

Left subtree will be true $a_i \leq a_j$
right subtree will be false $(a_i > a_j)$.

According to lower bound theory for a lower bound L(n) of an algorithm it is not possible to have any other algorithm (for a common problem) whose time complexity is less than L(n) for random input. Also every algorithm must take at least L(n) time in the worst case.
Note that L(n) here is the minimum of all the possible algorithms of maximum complexity.

Left subtree will be true condition i.e [ai<aj] by comparing a1 a2 a3