UNIT-I

Assignment :-1

Page No.
Date

Q.1. What is Algorithm ?

→ Algorithm is a Step-by-Step process to perform some action a problem.

Q.2. Define Properties of algorithms.

→ The Properties of algorithm are as Follows -

(1) input :- In this algorithm uses values from a Specific set in EE.

(2) Output :- For each input the algorithm produces a values from Specific task, Every input input has an output.

(3) Precision :- In this steps are precisely defined.

(4) Correctness :- Input is defined for that output is correct and desired.

(5) Finiteness :- In this ~~Otp~~ output after finite number of steps for each input.

(6) Determination :- The Result should be guaranteed.

(7) Generality :- Procedure applies to all problems not a Special Subset.

Q.3. Define Time Complexity?

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

Q.4. Define Space Complexity?

→ It defines that 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 number using quick sort

→ 50, 31, 71, 38, 77, 81, 12, 33.

Quick Sort Algorithm -

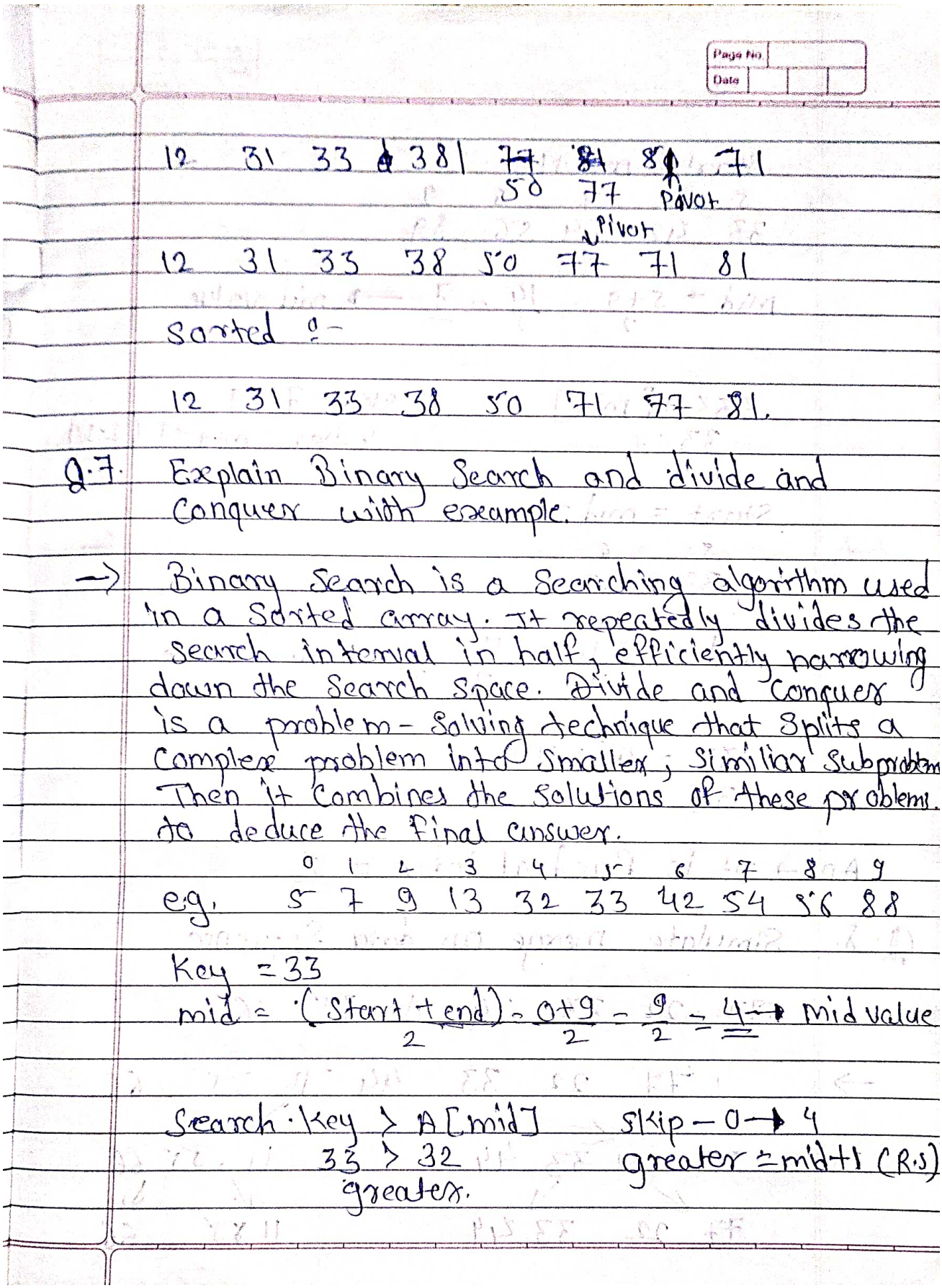
pivot

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 77 81 88 71
50 77 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, similar subproblems. Then it combines the solutions of these problems to deduce the final answer.

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

Key = 33

mid = (start + end) / 2 = 0+9/2 = 9/2 = 4 → mid value

Search: key > A[mid] 33 > 32 greater.

Skip - 0 → 4 greater = mid+1 (R.S)

Start = mid + 1

5	6	7	8	9
33	42	54	56	89

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

33 < A[mid] Skip = 7 → 9

33 < 7 lesser = mid − 1 (2-3).

Start = mid − 1

5	6
33	42

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

33 = A[mid]

33 = 33

Ans → It is found at index = 5

Q. 8. Simulate merge on data Sequence.

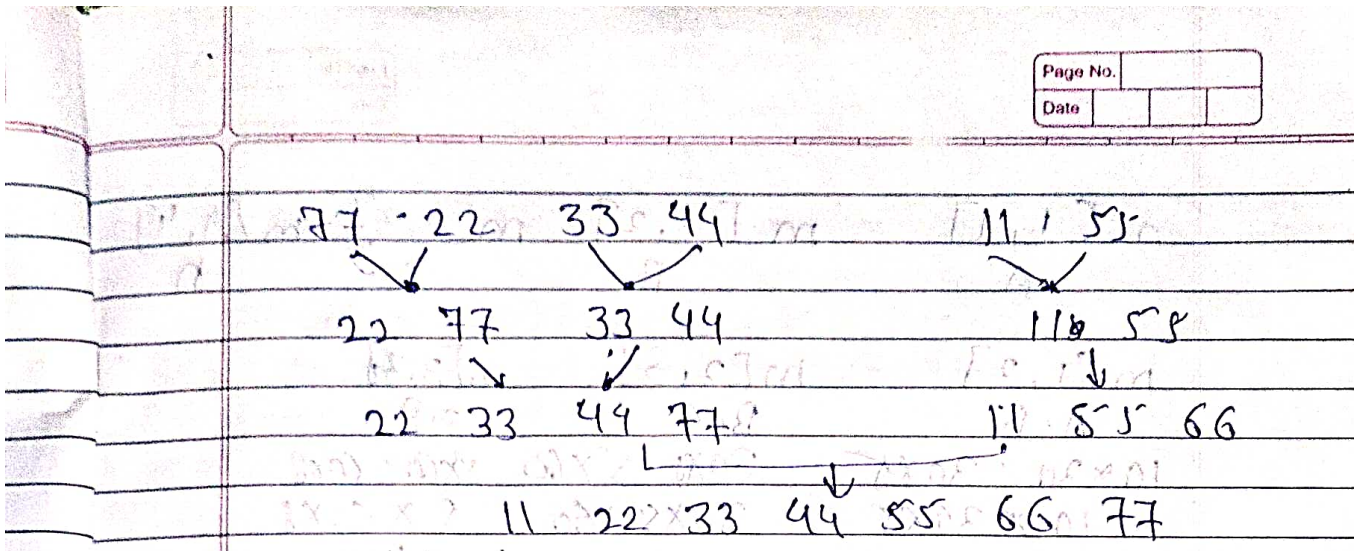
77 22 33 44 11 55 66

→ 77 22 33 44 11 55 66

77 22 33 44 | 11 55 66

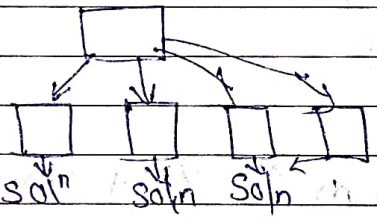
77 22 | 33 44 | 11 55 | 66

77 | 22 | 33 | 44 | 11 | 55 | 66



s]. 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

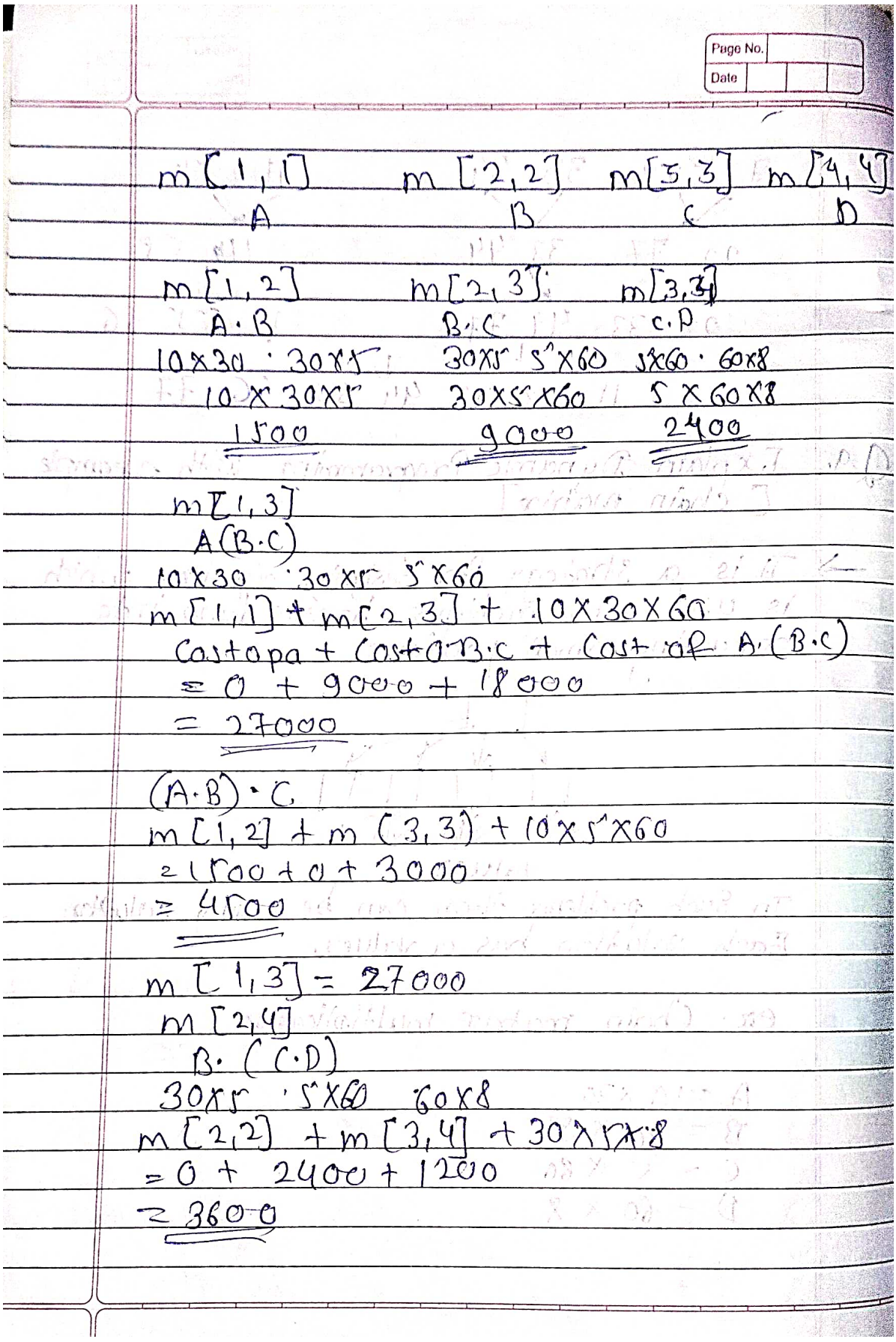


values.

In such problems there can be mainy 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 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 x 5 x 60 , 60 x 8

m[2,3] + m[4,4] + 30 x 60 x 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 x 30 x 8 ,
m(1,2) + m(3,4) + 10 x 5 x 8
m(1,3) + m(4,4) + 10 x 6 x 8 }

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

= min { 6000 , 4300 , 31800 }

= 31800

UNIT-5

Assignment :- 2

Page No.
Date

Q.1. What is network flow?

→ In Combinatorial optimization, network flow problems are a class of computational problems in which the input is a flow network (a graph with numerical capacities on its edges), and the constraints and that have incoming flow equal to....

Q.2. Define Residual Networks?

→ The Residual Network consists of an edge that can admit more net flow. Suppose we have a flow network $G=(V, E)$ with source s and sink t. Let f be a flow in G, and examine a pair of vertices $u, v \in V$. The sum of additional net flow we can push from u to v before exceeding the capacity $c(u,v)$ is the residual capacity of (u, v) given by $c_f(u,v) = c(u,v) - f(u,v)$.

Q.3. Define Augmenting path?

→ Given a flow network $G=(V, E)$ and a flow f, an augmenting path p is a simple path from s to t in the residual network G. By the solution of the residual network, each edge (u,v) on an augmenting path admits some additional positive net flow into v without violating the capacity constraint on the edge.

Page No.
Date

Let $G = (V, E)$ be a flow network with flow f. The residual capacity of an augmenting path p is

$$c_f(p) = \min \{c_f(u,v) : (u,v) \text{ is on } p\}$$

The residual capacity is the maximal amount of flow that can be pushed through the augmenting path.

Q.4. Which algorithm is use to solve maximum flow networks?

→ Ford-Fulkerson algorithm is a greedy approach for calculating the maximum possible flow in a network or a graph. A term, flow network, is used to describe a network of vertice and edges with a source (S) and a sink (T).

Q.5. What is maximum flow and minimum flow?

→ In Computer Science and optimization theory, the max-flow min-cut theorem states that in a flow network, the maximum amount of flow passing from the source to sink is equal to the total weight of the edges in a minimum cut: i.e the smallest total weight of the edges which if removed would disconnect the source.

Page No.
Date

Q.6. Write the steps of maximum flow?

→ (1) Initialize Flow :- Set the flow in all edges to 0.

(2) Choose an Augmenting Path :- Use a suitable algorithm (e.g. DFS, BFS) to find an augmenting path from the source to the sink in the residual graph.

- An augmenting path is a path in which all edges have residual capacity greater than 0.

(3) Determine Maximum Flow :- Determine the maximum flow that can be sent along the augmenting path. This is the minimum residual capacity of the edges in the path.

(4) Augment Flow :- Update the flow along the augmenting path by adding the determined maximum flow to each edge in the forward direction and subtracting it in the reverse direction.

(5) Update Residual Graph :-

- Update the residual capacities of the edges in the residual graph based on the new flows.
- For each forward edge with flow f, subtract f from its residual capacity, and for each reverse edge with flow f, add f to its residual capacity

Page No.
Date

(6) Repeat Steps 2-5 :- Repeat Steps 2-5 until no more augmenting paths can be found in the residual graph.

(7) Calculate Maximum Flow :-
- Once there are no more augmenting paths can be found in the network is now at its maximum.
- Sum up all the flows leaving the source (or entering the sink) to get the maximum flow.

(8) Termination :- Ensure that the algorithm, ~~Consider using advanced data~~ terminates, In some cases, if capacities, are not integers, you may need to use algorithms like Edmonds-Karp to guarantee termination.

(9) Optimization :- To optimize the algorithm, Consider using advanced data structures or path selection strategies.

Q.7. Write the steps for Fordfulkerson algorithm?

→ (1) Initialize Flow :- Set the flow in all edges to 0

(2) Find Augmenting Paths :-
- Start with an empty flow.
- Find an augmenting path from the source to the sink in the residual graph.

• You can use depth-first Search (DFS) or breadth-first Search (BFS) to find augmenting paths.

(3) Augment Flow :- • Once you find an augmenting path, determine the maximum flow that can be sent along this path.

(4) Update Residual Graph :- Update the residual capacities of the edges in the residual graph based on the new flow.

(5) Repeat Step 2-4 :- • Repeat Steps 2-4 until no more augmenting path can be found in the residual graph.

(6) Calculate Maximum Flow :- Once there are no more augmenting path, the flow in the network is now at its maximum.

Q. 8. Difference between Ford Fulkerson and maximum flow?

→ "Maximum Flow" refers to the overall concept and problem of finding the maximum amount of flow the can be sent from a source node to a sink node in a flow network. "Ford - Fulkerson" is one of algorithms designed to solve the maximum flow problem.

Page No.
Date

(1) Maximum Flow :-

- Definition :- Maximum flow is a concept in graph theory and network flow problems that represents the maximum amount of flow that can be sent from a designated source node to a designated sink node in a flow network.

- Objective :- The objective is to determine the optimal distribution of flow along the edges of the network to maximize the flow from the source to the sink, while respecting the capacities of the edge.

(2) Ford-Fulkerson :-

- Definition :- ~~Maximum flow is a concept~~ Ford-Fulkerson is an algorithmic approach to find the maximum flow in a flow network.

- Algorithmic Approach :- Ford-Fulkerson is a generic method for solving the maximum flow problem. It doesn't prescribe a specific rule for choosing augmenting paths.

- Termination Issue :- The basic Ford-Fulkerson algorithm doesn't guarantee termination when capacities have fractional values, so variations like Edmonds-Karp address this by always choosing the shortest augmenting path.

Page No.
Date

Q.9 Write Steps for maximum cut?

→ (1) Initialize Partitions :- Start with an arbitrary partition of the vertices into two sets, A and B.

(2) While Improvement Can Be Made :- Repeat the following steps as long as improvement can be made to the cut.

(3) Compute Cut Size :- Calculate the size of the cut by counting the number of edges crossing the partition (i.e. having one endpoint in Set A and the other in Set B).

(4) Identify Improving Edges :- Identify the edges that, if moved from one set to the other, would increase the cut size.

(5) Move Edge :- Choose one or more improving edges and move their vertice from one set to the other. This increases the cut size.

(6) Repeat Steps 3-5 :- Repeatedly Compute the cut size and move improving edges until no further improvements can be made.

(7) Output Result :- The final partition of the vertices into sets A and B represent the maximum cut in the graph.

Unit - 2

Assignment - 3

Page No.
Date

Q.1. Define disjoint Set ?

→ These are sets of data structure which supports 3 operations make Set, Union and find Set. It can be define as the subsets where there is no common element between the two sets.

Exe :- 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 between 2 sets.

Q.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 A∪B

Let A = {1, 2, 3} B = {3, 4, 5, 6}

A∪B = {1, 2, 3, 4, 5, 6}

Q.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.

Q.4. What are applications of disjoint set.

→ (1) Hashing functions, union find algorithms.
(2) stack operations.
(3) Heap operations, cycle detection in graph
(4) pushing & popping values.
(5) Job sequencing problem solving.
(6) kruskals algorithm, computer networks.

Q.5. Define lower bound theory.

→ Lower bound theory says that no algorithm can do the job in fewer than L(n) time units for arbitary 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.

Q.6. Explain disjoint set with one example.

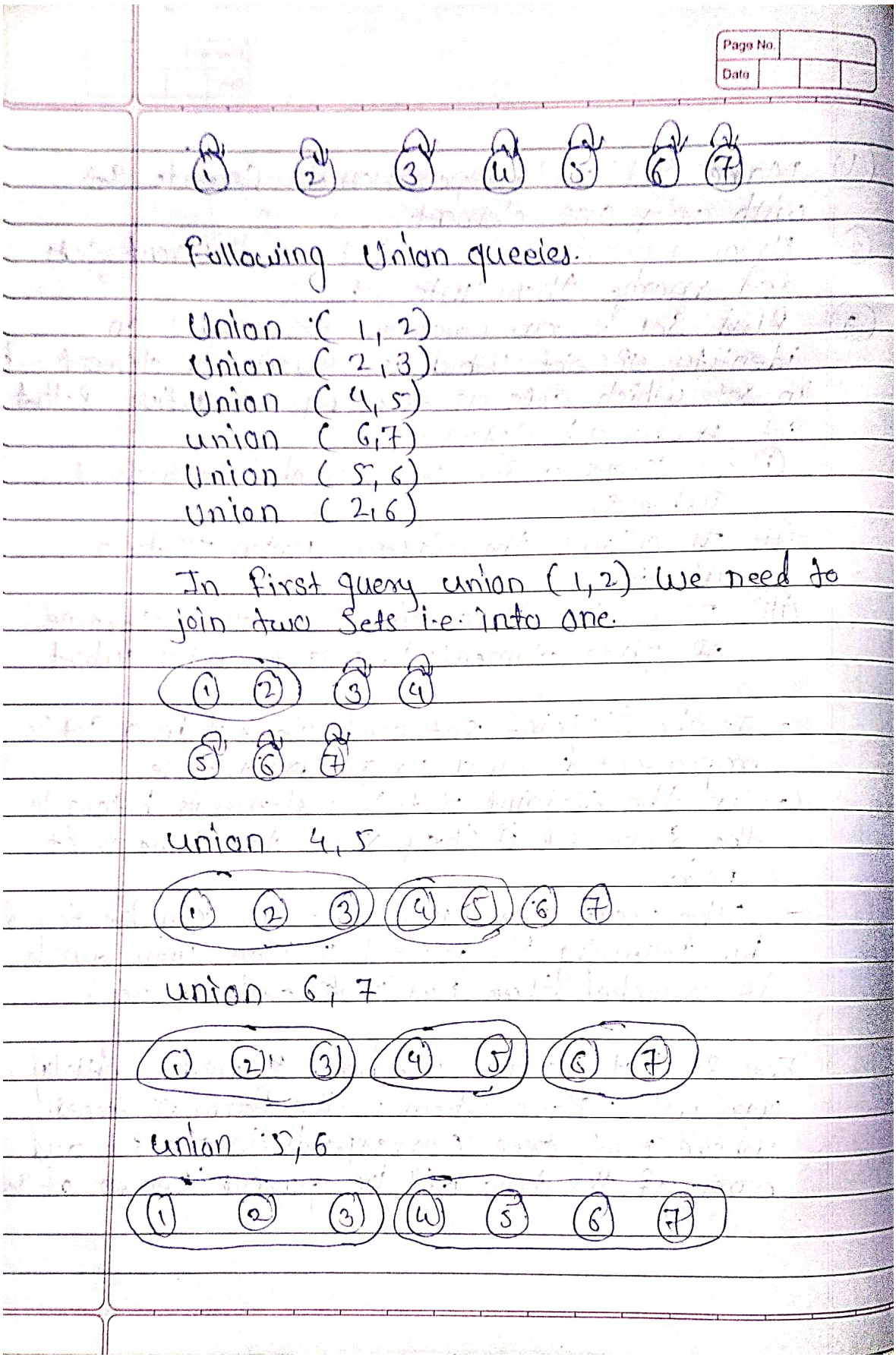
→ These are the sets of data structure which supports 3 operations make set, union and find set.

Page No.
Date

(1) Make Set is the Operation to Create Set with only one element.
(2) Union Operation is to take 2 different sets and merge them into 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 —
   (i) It keeps a Set partitioned into disjoint subsets.
   (ii) It allows the efficient union of two subsets.
   (iii) It makes it possible to quickly determine of 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 9 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


Page No.
Date
1 2 3 4 5 6 7
Following Union queeies.
Union (1, 2)
Union (2, 3)
Union (4, 5)
union (6, 7)
Union (5, 6)
union (2, 6)
In first query union (1, 2) we need to join two sets i.e. into one.
1 2 3 4
5 6 7
union 4, 5
1 2 3 4 5 6 7
union 6, 7
1 2 3 4 5 6 7
union 5, 6
1 2 3 4 5 6 7

union 2, 6

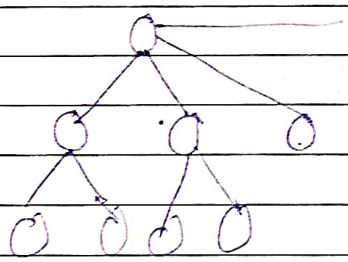
(1) (2) (3) (4) (5) (6) (7)

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.

Ex :-

Given 2 trees : Tree-1

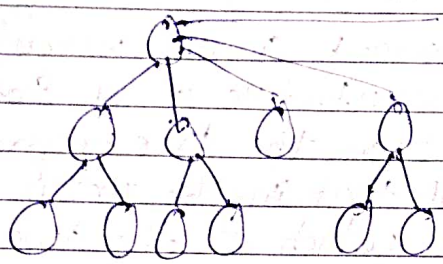


Tree - 2

Leader

Rank = 1

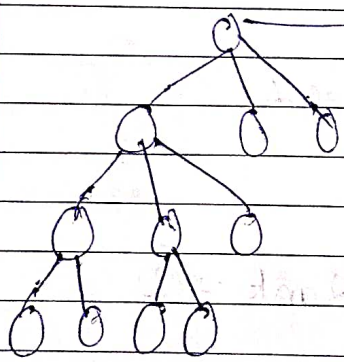
If we combine their rank then rank [tree 1] = 2 and rank [tree 2] = 1 combined..



Leader

Rank = 2

We can see that since tree 2 is attached to trees so rank of the combined tree is same as tree 1. If we combine tree 1 & 2 checking their rank then it would be



Leader

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 bound L(n) of an algorithm.
Once lower bound is calculated then we can compare it with actual complexity of algorithm & if their order are same then we declare it has optimal (best)
The Technique used for lower bound theory

Q.9 Difference Between Structure and Union

Structure	Union
(1) The struct keyword is used to define a structure.	(1) The union keyword is used to define union.
(2) Each variable member occupied a unique memory space.	(2) Variables members share the memory space of the largest size variable.
(3) Changing the value of a member will not affect other variable members	(3) Changing the value of one member will also affect other variables members.

(4) Each variable member will be assessed at a time.	(4) Only one variable member will be assessed at a time.
(5) It is used to store different data type values.	(5) It is used for storing one at a time from different data type values.
(6) We can initialize multiple variables of a structure at a time	(6) In union, only the first data member can be initialized.

Q.10. Explain techniques used for lower bound theory

→ (1) Comparisan 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
$a_i > a_j$ greater than
$a_i \geq a_j$ greater than or equal to
$a_i = a_j$ equal to

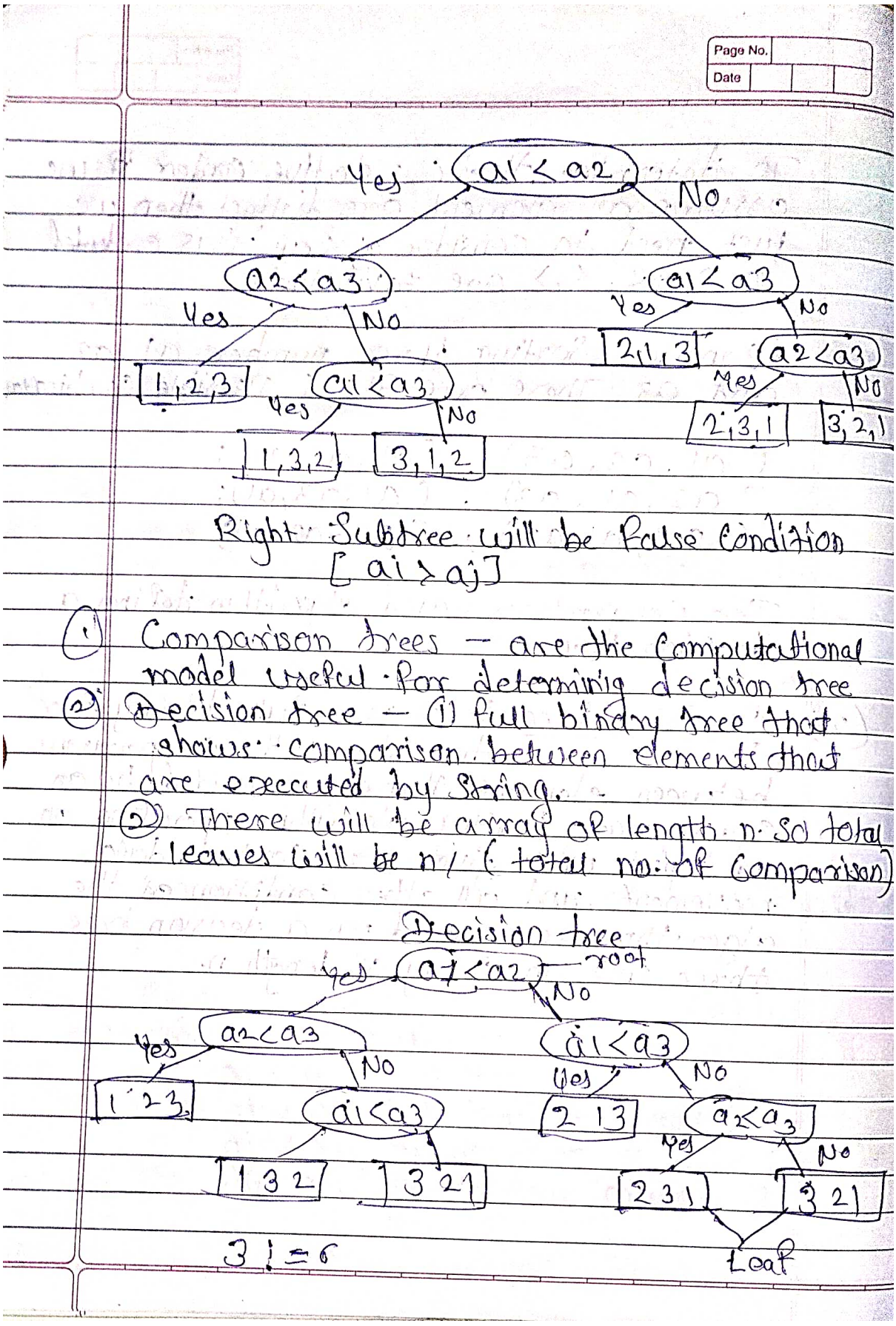
To determine their relative order if we assume all element are distinct then we just need to consider $a_i < a_j$ '=' is excluded or $\geq$; $\leq$, $<$, $>$ are equivalent.

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 condition of the algorithm are ignored In a decision tree three will an array of length n.


Page No.
Date
a1 < a2
Yes
No
a2 < a3
a1 < a3
Yes
No
Yes
No
1,2,3
a1 < a3
2,1,3
a2 < a3
Yes
No
Yes
No
1,3,2
3,1,2
2,3,1
3,2,1
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
root
Yes
No
a2 < a3
a1 < a3
Yes
No
Yes
No
1 2 3
a1 < a3
2 1 3
a2 < a3
Yes
No
1 3 2
3 2 1
2 3 1
3 2 1
3! = 6
Leaf

Page No.
Date

Left Subtree will be true ai<aj
Right Subtree will be false (ai>aj)

Unit - 3

Assignment - 4

Page No.
Date

Q.1. List various String matching algorithm

→ (i) The Navie String matching Algorithm
(ii) The Rabin - Karp - Algorithm
(iii) Finite Automata.
(iv) The Knuth - morries - Pratt Algorithm
(v) The Boyer - moore Algorithm.

Q.2. Explain in short naive string machine algorithm

→ • This is simple and efficient brute force approach.
• It compares the first character of pattern with searchable Text
• If a match is found, pointers is both strings are advanced.
• If a match is not found, the pointer to text is incremented & pointer of the pattern is reset
• This process is repeated till the end of the text.

Q.3. Define Pattern matching.

→ • Pattern matching is widely used in computer science and many other fields
• Pattern matching algorithm are used to search for patterns within a large text or data set.

- Pattern machine algorithms are used to find patterns within a bigger it of data or reset.
- Pattern matching algorithm are important because they allow us to search for patterns in an large data it quickly.

Q.4. Define string matching with finite automata.

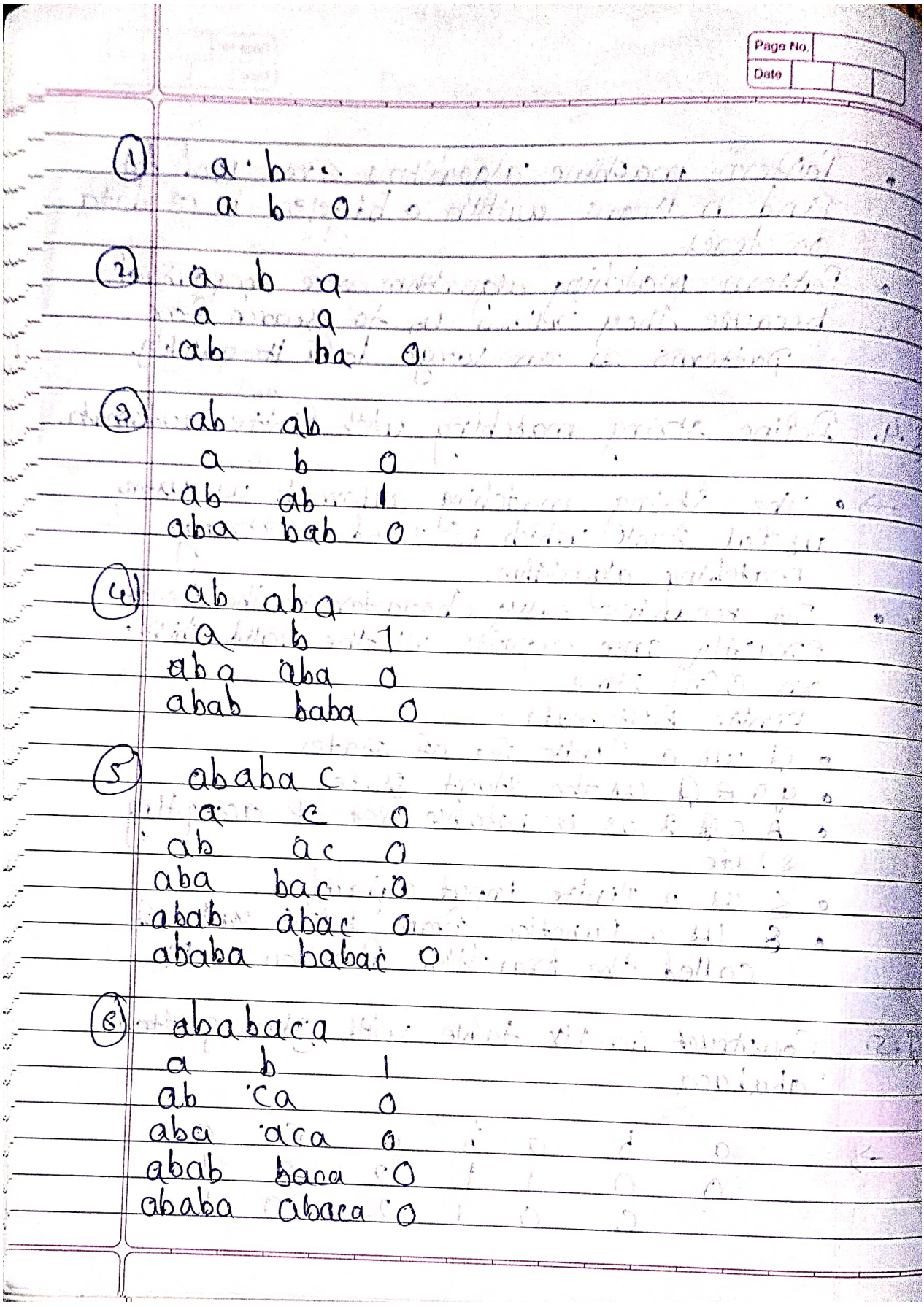
→ • The String matching automata as very useful tool which is used in string matching algorithm.
- It examins each character in the reset exactly once reports all the valid shifts in O(n) time

Finite Automata
- Q is a finite set of states
- $q_0 \in Q$ is the start state.
- $A \subseteq Q$ is a notable set of accepting state
- Σ is a finite input alphabet,
- δ is a function from $Q \times \Sigma$ into Q called the transition function of M

Q.5. Construct prefix table with given pattern ababaca

→

a	b	a	b	a	c	a
0	0	1	1	2	0	1
0	0	1	2	3	0	1



(1) a b
a b 0

(2) a b a
a a
ab ba 0

(3) ab ab
a b 0
ab ab 1
aba bab 0

(4) ab aba
a b 1
aba aba 0
abab baba 0

(5) ababa c
a c 0
ab ac 0
aba bac 0
abab abac 0
ababa babac 0

(6) ababaca
a b 1
ab ca 0
aba aca 0
abab baca 0
ababa abaca 0

Page No.
Date

Q6. Write and explain string matching with Finite automata with one example.

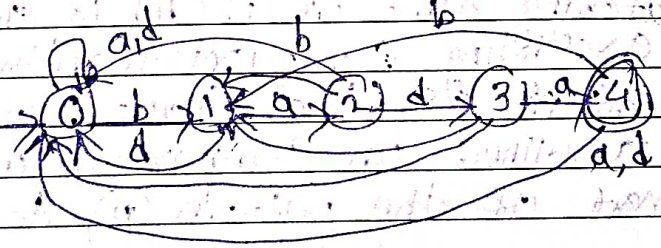
→ String Matching :-

- The string matching automation as a very useful tool which is used in string matching algorithm
- The goal of string matching as to find the occurrence of specific pattern within the larger body of text.

Finite Automata :

- A Finite Automata :-

(i) A Finite automata in as a 5-tuple (Q, q0, A, Σ, δ), where
(ii) Q is a finite set of states.
(iii) q0 ∈ Q is the start state.
(iv) A ⊆ Q is a notable set of accepting states.
(v) Σ is a finite input alphabet
(vi) δ is a function from Q x Σ into Q. Called the transition function of M.
(vii) Example :- Construction string matching with finite automata of transition Table (boda)



	a	b	d
0	0	1	0
1	2	1	0
2	1	0	3
3	4	1	0
4	0	1	0

(1) b → a bb = 1
b → d bd = 0

(2) ba → a baa = 0
ba → b bab = 1

(3) bad → b badb = 1
bad → d badd = 0

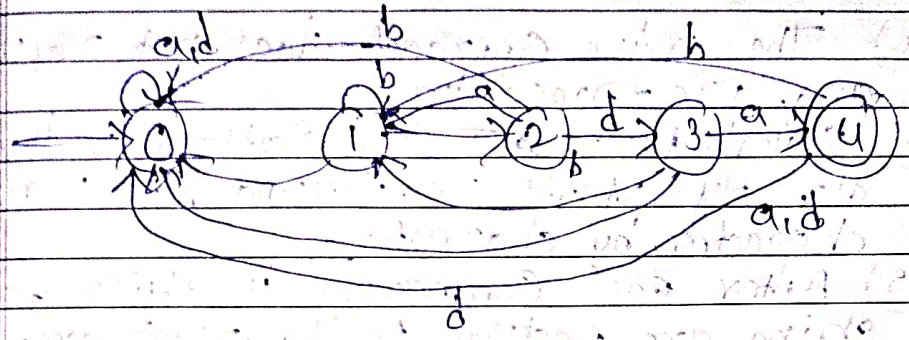
(4) bada → a badaa = 0
bada → b badab = 1
bada → d badad = 0

Q.7 Explain and write knuth-morris-pratt algorithm with one example.

→ • Knuth - Morris - Pratt introduce a linear time algorithm for the string linear matching problem.
• A matching time of O(n) is achieved by avoiding comparison with an element of (S) that have previously been involved in comparison with some element of the pattern (P) to be matched

Page No.
Date

(8) Construct prefix table with given pattern (bada) and draw transition diagram.



	a	b	d
0	0	1	0
1	2	1	0
2	1	0	3
3	4	1	0
4	0	1	0

Pefic & construction

(1) b → a bb = 1
b → d bd = 0

(2) ba → a baa = a
ba → b bab = 1

(3) bada → a badaa = a
bada → b badab = 1
bada → d badad = 0

Page No.
Date

Q.9. Explain in brief Naive - string matching algorithm with example.

→ (1) The naive approach does not require any pre-processing.
(2) Given text T and pattern P. It directly starts comparing both strings character by character.
(3) After each comparsion it shifts pattern string one position to the right example.
Q.7.

Q.10. Describe various String matching Algorithm.

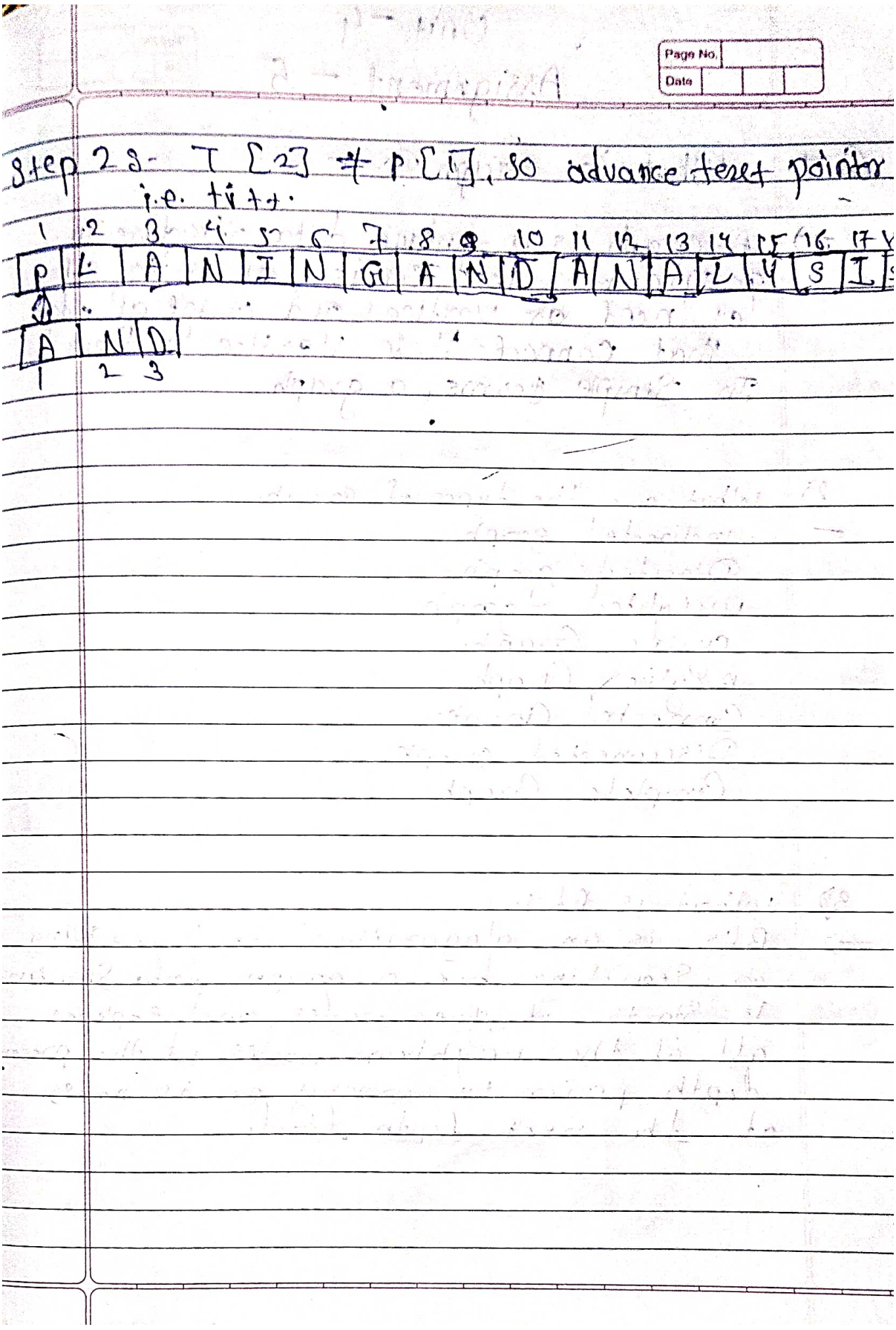
→ (1) The Naive String Matching Algorithm
- This is simple and efficient brute-force approach
- It compares the first character of pattern with searchtable text
  i.e. backtracking on the string (S) never occurs.

example :-

Step I :- T[1] ≠ P[1] ,& advance text pointer.
i.e. if

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
P	L	A	N	I	N	G	A	N	D	A	N	A	L	Y	S	I/S

1	2	3
A	N	D


Page No.
Date
Step 2 :- T [2] ≠ P [1], so advance text pointer i.e. i++.
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
P L A N I N G A N D A N A L Y S I
A N D
1 2 3

Unit - 4

Assignment - 5

Page No.
Date

Q.1: What is Graph?

→ A graph is a unique data structure in programming that consists of finite set of need or vertices and a set of edge that connect these vertice to them. In simple terms, a graph

2) What are the types of graph.

— Undirected gaph
Directed gaph
weighted gaph
cyclic Graph.
Acylic Graph
Connected Graph.
Disconnected Graph.
Complete Graph

3) What is BFS.

→ BFS is an algorithm for traversing or Searching tree or graph data Structure. It starts at given nodes and explores all of the neighbour nodes at the present depth prior to moving on to nodes at the next depth level.

Page No.
Date

4. Define tree edge, forward edge, Cross edge back edge.

→ Tree: An edge that is part of the tree structure formed by DFS traversal. It connects a node to a newly discovered node.

Forward edge: An edge that connects a nodes to a descendant in the DFS tree but is not a tree edge.

Cross edge :- An edge that connects nodes across different branches of the DFS tree

Back Edge :- An edge that connects a node to one of its ancestors in the DFS tree forming a cycle.

Q.5. What is DFS

→ DFS (Depth First Search) is an algorithm for traversing or searching tree or graph data structures. It starts at a given nodes (root) and explores as far as possible along each branch before backtracking.

Page No.
Date

Q.6 Write algorithm for BFS with one example.

→ The steps involved in the BFS algorithm to explore a graph are given as follows

Step 1 :- SET STATUS = 1 (ready state) for each node in G

Step 2 :- Enqueue the starting node A and set its STATUS = 2 (waiting state)

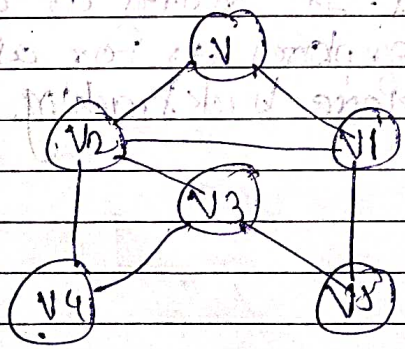
Step 3 :- Repeat step 4 and 5 until Que of is empty

Step 4 :- Dequeue a node N. Process it.

Step 5 Enqueue all the neighbours of N that are in the ready.

State (where

Step Step 6 :- Exit



Page No.
Date

Step 1 :- V

Step 2 :- V2 V1

Step 3 :- V1 V4 V3

Step 4 :- V4 V3 V5

Step 5 :- V3 V5 Delete V4

Step 6 :- V5 Delete V3

Step 7 :- Delete V5

Q.7. Write algorithm for DFS with one example.

Step 1 :- Set status := 1 (ready state) for each node in G.

Step 2 :- Push the starting node A on the stack & set its status := 2 (waiting state)

Step 3 :- Repeat Step 4 and 5 until stack is empty.

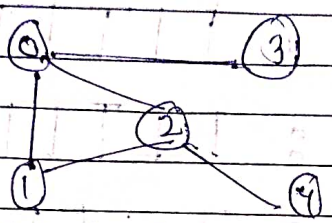
Step 4 :- Pop the top node N & process it and set its status = 3 processed state.

Page No.
Date

Step 5 :- Push on the Stack all the neighbours of N that are in the ready state. (Whose status = 1) and set these status = 2 (waiting state)
[End of loop]

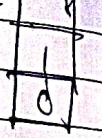
Step 6 :- Exit

Ex :-



Step 1 :- Select Starting Point & Push

Step 2 :- Push adjecent element of 0 into Stack push 1

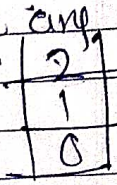


Step 3 :- Push 2

Step 4 :- Push 4 because 0, 1, 2, are

Page No.
Date

Step 5 :- Since 4 does not have any adjecent element left : Pop 4 from Stack



Pop=4

Step 6 :- Same as above pop 2

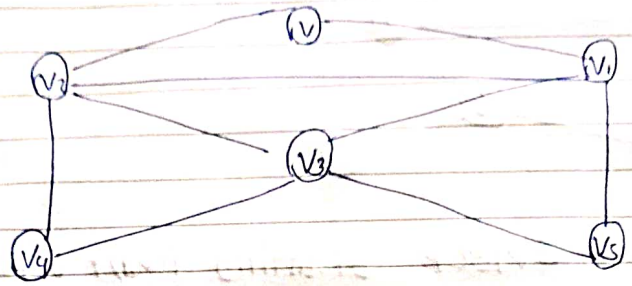
Pop=2

step 7 :- Pop 1

8 Difference between BFS and DFS.

BFS	DFS
• BFS stands for Breadth First search	DFS stands for DEPth First search.
• It a vertex-based technique to find the shortest Path in a graph.	It is an edge-based technique because the vertices along the edge are explored first from the starting to the en
• BFS is a traversal technique in which all the nodes of the same level are explored first and then we move to the next level.	DFS is also a traversal technique in which tra traversal is started fr the root node and exp the nodes as far as possible until.
• Queue data structure is used for the BFS traversal	Stack data structure is used for the BFS traversal.
• BFS does not use the backtracking concept	DFS uses backtracking traverse all the unvisi nodes
• BFS is slower than DFS	DFS is faster than BF

9 Determine BFS.



Step 1 : V

Step 2 : V2 V1

Step 3 : V1 V4 V3

Step 4 : V4 V3 V5

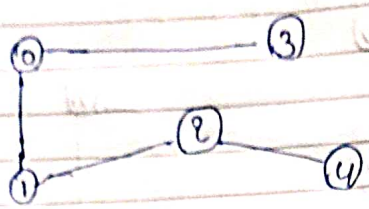
Step 5 : V3 V5

There is no element present adjacent to V4
Delete V4

Step 6 : V5 Delete V3

Step 7 : Delete V5

Q. Determine DFS

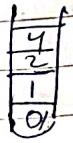


Step 1 : Select starting point 0. push 0

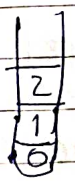
Step 2 : Push adjacent element of 0 into stack push 1

Step 3 : Push 2

Step 4 : Push 4 because 0, 1, 2, are



Step 5 : Since 4 does not have any adjacent element so we will pop 4 from stack



Pop 4

Step 6 : Same as above Pop 2

Pop = 2

Step 7 :- Pop 1