School of Computing
Second CIA Exam – March 2024
Course Code: CSE318
Course Name: Algorithm Design Strategies & Analysis
Duration: 90 minutes Max Marks: 50

Key for Evaluation

PART A

Answer all the questions **10 x 2 = 20 Marks**

1. Compare the backtracking strategy with brute-force approach.

Ans:

The **brute-force** approach involves generating all possible combinations, which grows exponentially with the size of the input. Specifically, for each additional tree, there are n possible positions to place it, so for k additional trees, there are n^k combinations to consider.

The **backtracking** approach is more efficient than brute force in this case. It avoids generating all possible combinations and instead explores only the promising ones. It starts by placing additional trees one by one, considering all possible positions for each tree. At each step, it checks whether placing the tree at a certain position leads to a valid configuration

2. Predict the algorithm design strategy used in the following algorithms. (a) 0/1 Knapsack Problem (b) Sum of Subset Problem (b) String Editing Problem (d) Kruskal's Algorithm

Ans:

(a) Dynamic Programming
(b) Backtracking
(c) Dynamic Programming
(d) Greedy

3. Relate the hamiltonian cycles problem with the Travelling Salesman Problem.

Ans:

Hamiltonian Cycle Problem: Given a graph G, determine whether there exists a cycle that visits each vertex exactly once and returns to the starting vertex.

Traveling Salesman Problem (TSP): Given a complete graph G (where every pair of distinct vertices is connected by a unique edge) with edge weights, find the shortest possible tour that visits each vertex exactly once and returns to the starting vertex.

4. Mention the bounding conditions used for backtracking in sum of subset problem.

Ans:

1. BalanceSum = 0
2. BalanceSum < 0
3. Level Number = Size

5. Say True or False

(a) In the adjacency matrix representation of directed graph, the matrix is symmetric.
(b) In the adjacency matrix representation of undirected graph, the number of 1's is twice the number of vertices.

Ans:

(a) False
(b) False

6. Match the following.

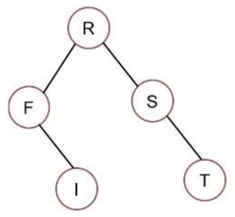
Prim's Algorithm	Topological Order
BFS	Priority Queue
DFS	SET concept
Kruskal's Algorithm	Queue

Ans:

Prims – Priority Queue
BFS – Queue
DFS – Topological Order
Kruskals – SET concept

7. Find the search cost for the following BST with the given probability of key elements.

n = 5	0	1	2	3	4	5
Keys[1..5]		F	I	R	S	T
P[1..5]		0.15	0.1	0.05	0.1	0.2
Q[0..5]	0.05	0.1	0.05	0.05	0.05	0.1



Ans:

C[0,5] = [0.05*1 + 0.15*2 + 0.1*2 + 0.1*3 + 0.2*3] + [0.05*2 + 0.1*3 + 0.05*3 + 0.05*2 + 0.05*3 + 0.1*3]
= .05 + .3 + .2 + .3 + .6 + .1 + .3 + .15 + .1 + .15 + .3
= 2.55

8. Write the recursive formula of optimal sub structure property for the 0/1 knapsack problem.

Ans:

$i, w \quad max(p_i + m[i-1, w_i - w_i], m[i-1, w])$

9. Describe the n-Queen problem.

Ans:

The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other. The chess queens can attack in any direction as horizontal, vertical, horizontal and diagonal way.

10. What is the use of State Space Tree?

Ans:

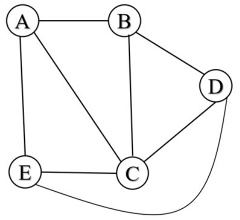
A state space tree is a tree representing all the possible states (solution or nonsolution) of the problem from the root as an initial state to the leaf as a terminal state.

PART B

Answer any three questions **3 x 10 = 30 Marks**

11. (a) Write the algorithm using backtracking strategy for the sum of subset problem. ***(5 Marks)***

(b) Find all the hamiltonian cycles present in the following graph by applying backtracking strategy. ***(5 Marks)***

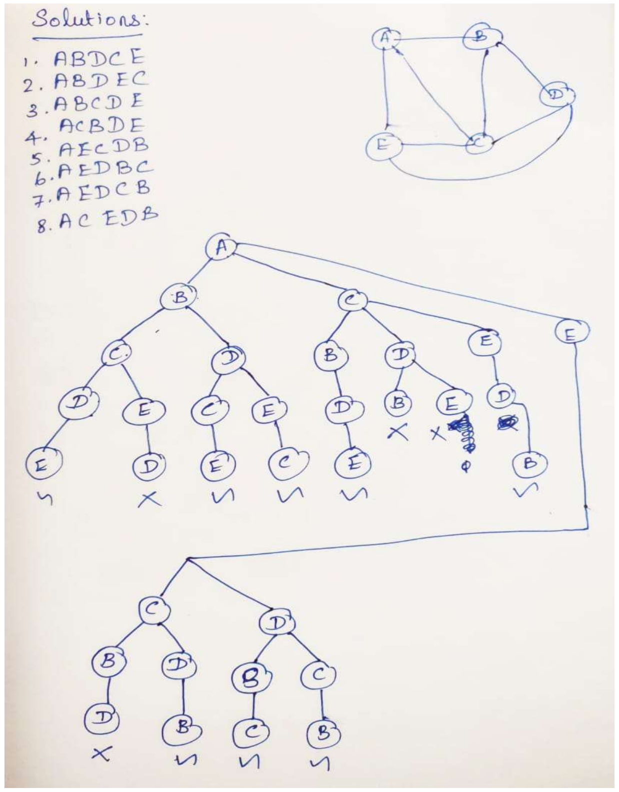


Ans:

(a) Sum of Subset – Algorithm

```
Algorithm SubSetSum(next, n, set[0..n-1], balanceSum, subset, solutionSet[], soutionSize)
        //No Solution found... Backtrack..
        If balanceSum<0 Then
                Return
        End If
        //Solution Found... Record the Solution... Backtrack...
        If balanceSum=0 Then
                solutionSet [solutionSize++] ← subset
                Return
        End If
        //If No more choices... Backtrack...
        If next = n Then
                Return
        End If
        //If next is included
        subset.X[next] ← 1
        SubSetSum(next +1, n, set, balanceSum - set[i], subset, solutionSet, solutionSize)
        //If next is not included
        subset.X[next] ← 0
        SubSetSum(next +1, n, set, balanceSum, subset, solutionSet, solutionSize)
End SubSetSum
```

(b) Steps (state space tree) – 3 Marks
Cycles (correct answer) – 2 Marks



Solutions:

1. ABDCE
2. ABDEC
3. ABCDE
4. ACBDE
5. AECDB
6. AEDBC
7. AEDCB
8. ACEDB

12. (a) Write dynamic programming algorithm for constructing optimal binary search tree. *(5 Marks)*

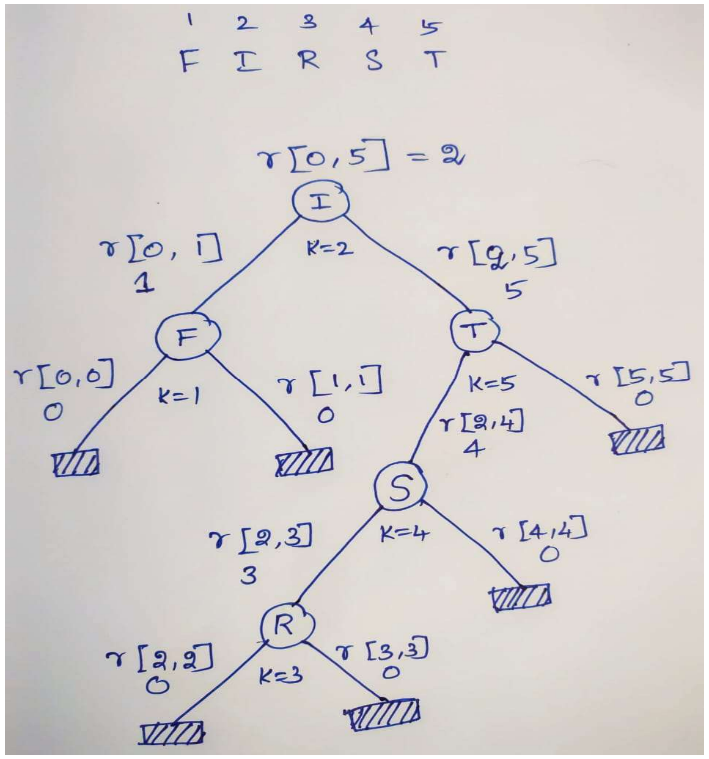
(b) Construct the optimal binary search tree for the following root table (r) which is obtained by applying dynamic programming approach for the key elements: Keys[1..5] = {F, I, R, S, T}. *(6 Marks)*

0	1	2	3	4	5	**r**
0	1	1	2	2	2	**0**
	0	2	2	2	4	**1**
		0	3	4	5	**2**
			0	4	5	**3**
				0	5	**4**
					0	**5**

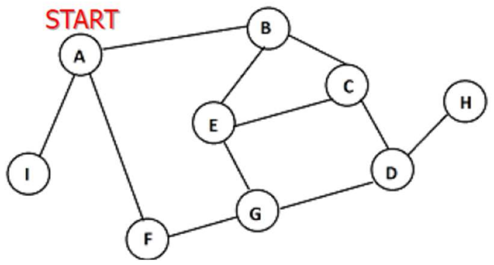
(a) Optimal BST – Algorithm

```
Algorithm OptimalBST(Keys[1..n], P[1..n], Q[0..n], n)
    Input:    Keys[1..n] – 'n' numbers of integer key elements
              P[1..n] – Probability of Successful Searches
              Q[0..n] – Probability of Unsuccessful Searches
              n – Number of Key Elements
    Output:   C[0..n, 0..n] – Cost Matrix
              R[0..n, 0..n] – Root Matrix

    Let C[0..n, 0..n] be an array – Cost Matrix
    Let W[0..n, 0..n] be an array – Weight Matrix
    Let R[0..n, 0..n] be an array – Root Matrix

    For Len ← 1 to n+1 do
        For i ← 0 to (n+1)-Len do
            j ← i + Len – 1
            If i=j then
                W[i, j] ← Q[i]
            Else
                W[i, j] ← W[i, j-1] + P[j] + Q[j]
            End If
        End For
    End For

For Len ← 1 to n+1 do
        For i ← 0 to (n+1)-Len do
            j ← i + Len – 1
            If i=j then
                C[i, j] ← R[i, j] ← 0
            Else
                Min ← ∞
                MinK ← -1
                For k ← i+1 to j do
                    Sum ← C[i, k-1] + C[k, j] + W[i, j]
                    If Sum<Min then
                        Min ← Sum
                        MinK ← k
                    End If
                End For
                C[i, j] ← Min
                R[i, j] ← MinK
            End If
        End For
    End For

Return C & R

End OptimalBST
```

(b)

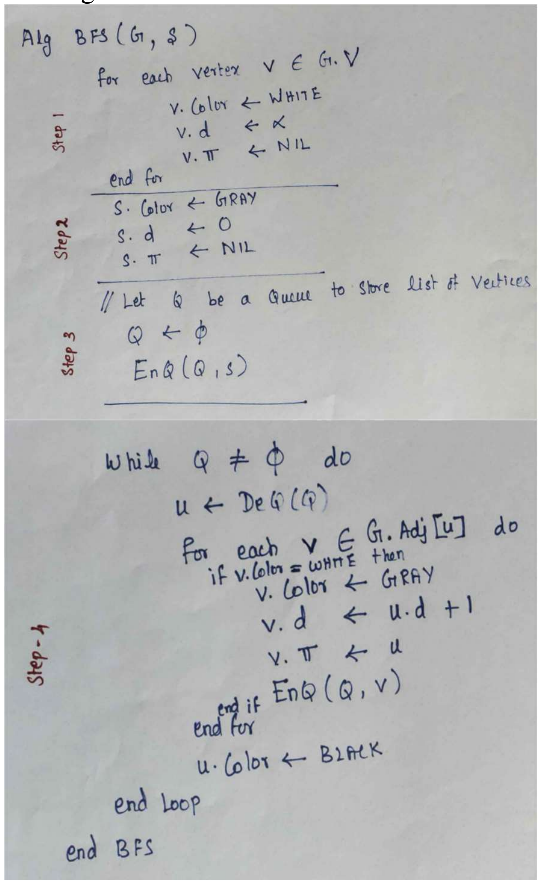


13. Which traversal algorithm used for finding shortest distance from the given starting vertex to all other vertices in a unweighted graph. Write the algorithm and find the shortest distance from 'A' to all other vertices by tracing algorithm.



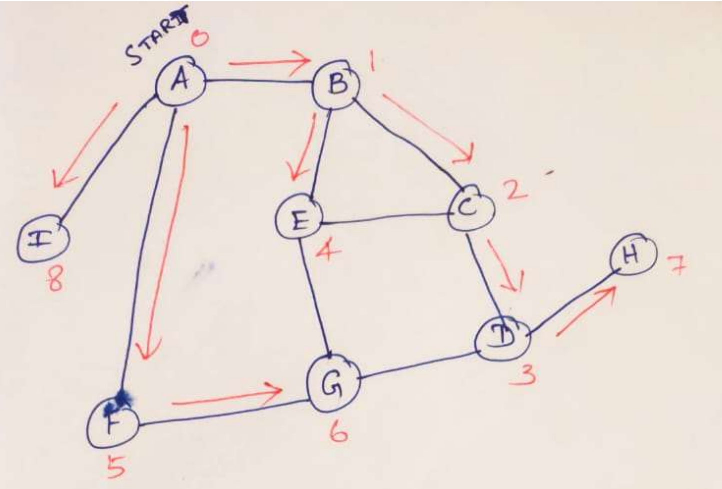
Ans:

BFS algorithm is used in shortest distance.



```
Alg BFS(G, s)
           for each vertex v ∈ G.V
                  v.Color ← WHITE
Step 1            v.d     ← ∝
                  v.π     ← NIL
           end for
           ________________________
           s.Color ← GRAY
Step 2     s.d     ← 0
           s.π     ← NIL
           ________________________
           // Let Q be a Queue to Store list of Vertices
Step 3        Q ← ϕ
              EnQ(Q, s)
           ________________________

           While Q ≠ ϕ do
                  u ← DeQ(Q)
                  for each v ∈ G.Adj[u] do
                     if v.Color = WHITE then
                        v.Color ← GRAY
Step - 4                v.d ← u.d + 1
                        v.π ← u
                        EnQ(Q, v)
                     end if
                  end for
                  u.Color ← BLACK
           end Loop
        end BFS
```

Tracing:

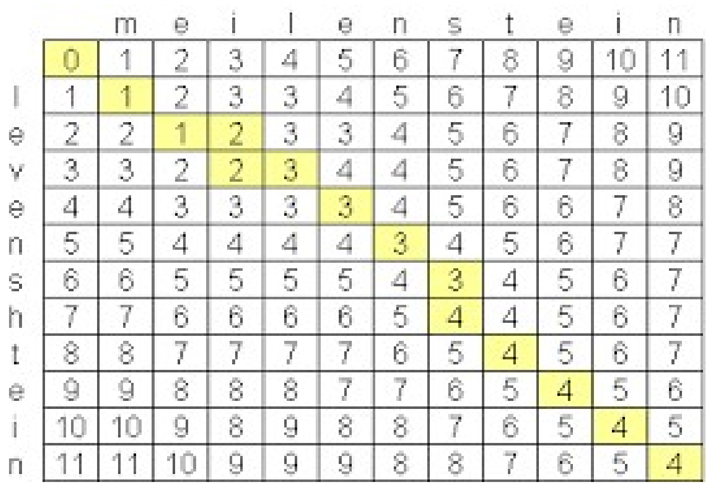


	value	Color	d	π	Adj	
~~u~~ ~~s~~	0	~~W~~ ~~G~~ B	~~∝~~ 0	~~N~~ N		→ ~~1~~ → ~~5~~ → ~~8~~
~~u~~	1	~~W~~ ~~G~~ B	~~∝~~ 1	~~N~~ 0		→ ~~0~~ → ~~2~~ → ~~4~~
~~u~~	2	~~W~~ ~~G~~ B	~~∝~~ 2	~~N~~ 1		→ ~~1~~ → ~~3~~ → ~~4~~
~~u~~	3	~~W~~ ~~G~~ B	~~∝~~ 3	~~N~~ 2		→ ~~2~~ → ~~6~~ → ~~7~~
~~u~~	4	~~W~~ ~~G~~ B	~~∝~~ 2	~~N~~ 1		→ ~~1~~ → ~~2~~ → ~~6~~
~~u~~	5	~~W~~ ~~G~~ B	~~∝~~ 1	~~N~~ 0		→ ~~0~~ → ~~6~~
~~u~~	6	~~W~~ ~~G~~ B	~~∝~~ 2	~~N~~ 5		→ ~~3~~ → ~~4~~ → ~~5~~
~~u~~	7	~~W~~ ~~G~~ B	∝ 4	~~N~~ 3		→ ~~3~~
~~u~~	8	~~W~~ ~~G~~ B	~~∝~~ 1	~~N~~ 0		→ ~~0~~

	1 ↓	3 ↓	4 ↓	5 ↓	7 ↓	8 ↓	10 ↓	13 ↓	17 ↓
Q	0	1	5	8	2	4	6	3	7
	↓ 2	↓ 6	↓ 9	↓ 11	↓ 12	↓ 14	↓ 15	↓ 16	↓ 18

14. Tranform a string “L E V E N S H T E I N” into another string “M E I L E N S T E I N” by using minimum numbers of editing operations by applying dynamic programming approach.

Ans:



		m	e	i	l	e	n	s	t	e	i	n
	0	1	2	3	4	5	6	7	8	9	10	11
l	1	1	2	3	3	4	5	6	7	8	9	10
e	2	2	1	2	3	3	4	5	6	7	8	9
v	3	3	2	2	3	4	4	5	6	7	8	9
e	4	4	3	3	3	3	4	5	6	6	7	8
n	5	5	4	4	4	4	3	4	5	6	7	7
s	6	6	5	5	5	5	4	3	4	5	6	7
h	7	7	6	6	6	6	5	4	4	5	6	7
t	8	8	7	7	7	7	6	5	4	5	6	7
e	9	9	8	8	8	7	7	6	5	4	5	6
i	10	10	9	8	9	8	8	7	6	5	4	5
n	11	11	10	9	9	9	8	8	7	6	5	4

N - L E V E N S H T E I N – No Action
I - L E V E N S H T E I N – No Action
E - L E V E N S H T E I N – No Action
T - L E V E N S H T E I N – No Action
H - L E V E N S ~~H~~ T E I N – **Remove H**
S - L E V E N S T E I N – No Action
N - L E V E N S T E I N – No Action
E - L E V E N S T E I N – No Action
V - L E V L E N S T E I N – **Insert L**
V - L E I L E N S T E I N – **Replace V with I**
E - L E I L E N S T E I N – No Action
L - M E I L E N S T E I N – **Replace L with M**