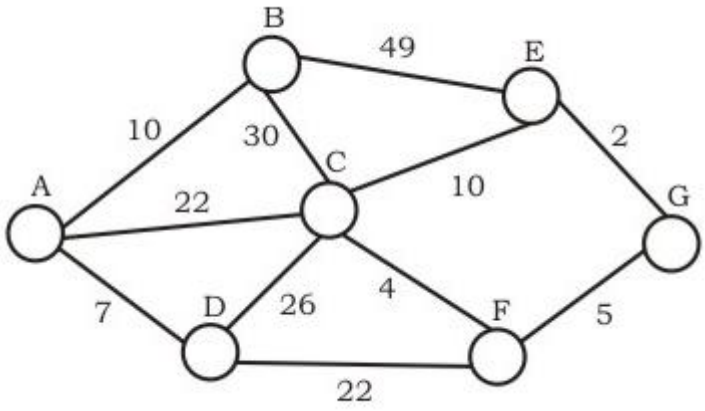
CONCEPT BUILDER 1-7

Q1.
Consider the undirected graph below:



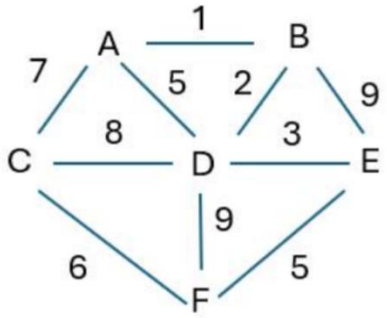
Using Prim's algorithm to construct a minimum spanning tree starting with node A, which one of the following sequences of edges represents a possible order in which the edges would be added to construct the minimum spanning tree?

- (A, B), (A, D), (D, F), (F, G), (G, E), (F, C)
- (E, G), (C, F), (F, G), (A, D), (A, B), (A, C)
- (A, D), (A, B), (A, C), (C, F), (G, E), (F, G)
- **(A, D), (A, B), (D, F), (F, C), (F, G), (G, E)**

Q2.
Prim’s algorithm is a _____.

- Divide and conquer algorithm
- **Greedy algorithm**
- Dynamic Programming
- Approximation algorithm

Q3.
Consider the following graph:

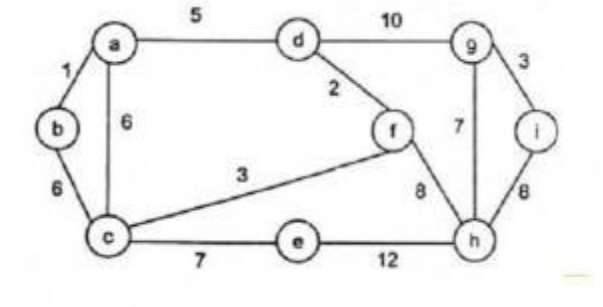


Which edges would be included in the minimum spanning tree using Prim's algorithm starting from vertex A?

- **AB, BD, DE, EF, FC**
- AC, CD, DE, EB, BF
- AB, BD, DE, EC, CF
- AC, CD, DE, EB, FE

Q4.

For the undirected, weighted graph given below, which of the following sequences of edges represents a correct execution of Prim's algorithm to construct a Minimum Spanning Tree?



- (a, b), (d, f), (f, c), (g, i), (d, a), (g, h), (c, e), (f, h)
- (c, e), (c, f), (f, d), (d, a), (a, b), (g, h), (h, f), (g, i)
- **(d, f), (f, c), (d, a), (a, b), (c, e), (f, h), (g, h), (g, i)**
- (h, g), (g, i), (h, f), (f, c), (f, d), (d, a), (a, b), (c, e)

Q5.

Which of the following is true?

- **Prim's algorithm initializes with a vertex**
- Prim's algorithm initializes with a edge
- Prim's algorithm initializes with a vertex which has smallest edge

Prim’s algorithm initializes with a forest

Q6.

A graph having an edge from each vertex to every other vertex is called a _____

Tightly Connected

Strongly Connected

Weakly Connected

Loosely Connected

Q7.

Let G = (V, E} be a directed graph where V Ts is the set of vertices and E the set of edges. Then which one of the following graphs has the same strongly connected component

where E = ((u, v)l(u, v) e E}

G2 (V, E2)t here Et = t(u, v}|(v, u) e E}

G3 - (V, Et) where E3 - t(•, v)| there is a path of length‹ 2 from u to v in E}

Gt = (V4. E) where Vt is the set of vertices in G which are not isolated

Q8.

What are the two main components of a graph data structure?

Elements and Links

Points and Lines

Vertices and Edges

Nodes and Arrays

Q9.

How are graphs commonly utilized in representing real-life networks?

Telecommunication circuits

Social networks like LinkedIn

Paths in a city

All of the mentioned options

Q10.

Which of the following statements is true for a **strongly connected** directed graph?

There exists at least one path between any two vertices in the graph.

Every vertex has an outgoing edge to at least one other vertex.

There is a directed path from every vertex to every other vertex in the graph.

The graph has no cycles but still maintains connectivity between vertices.

Q11.
0/1 knapsack is based on _____method.

Greedy method

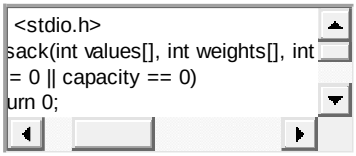
Branch and bound

Dynamic programming

Divide and conquer

Q12.
Consider the following dynamic programming code snippet for solving the 0/1 Knapsack problem:

Given the values [60, 100, 120] and weights [10, 20, 30], what would be the output of calling knapsack(values, weights, 50, 3)?




```
#include <stdio.h>
int knapsack(int values[], int weights[], int capacity, int n) {
   if (n == 0 || capacity == 0)
      return 0;
   if (weights[n - 1] > capacity)
      return knapsack(values, weights, capacity, n - 1);
   else
      return (values[n - 1] + knapsack(values, weights, capacity - weights[n - 1], n - 1)) >
            knapsack(values, weights, capacity, n - 1)
            ? (values[n - 1] + knapsack(values, weights, capacity - weights[n - 1], n - 1))
            : knapsack(values, weights, capacity, n - 1);
}
```

180

220

280

300

Q13.
The time complexity of solving the 0-1 Knapsack Problem using dynamic programming with a bottom-up approach (tabulation) is:

O(n)

O(n log n)

O(n * capacity)

O(n * capacity^2)

Q14.
Given 0-1 knapsack problem and fractional knapsack problem and the following statements:

S1: 0-1 knapsack is efficiently solved using the Greedy algorithm.
S2: Fractional knapsack is efficiently solved using Dynamic programming.

Which of the following is true?

S1 is correct and S2 is not correct

Both S1 and S2 are correct

Both S1 and S2 are not correct

S1 is not correct and S2 is correct

Q15.
A 0-1 knapsack problem has four items and a knapsack capacity of 11. The weight and profit of each item are given in the below
Pi (Rs.) 50 30 32 27 W = 11 Wi (Kg) 5 6 4 3 Which of the following greedy strategy gives the maximum profit?

Lightest item first

Largest profit first

Largest profit per unit weight first

Heaviest item first

Q16.
What is the parenthesization strategy in Matrix Chain Multiplication?

- To multiply matrices from left to right
- To multiply matrices from right to left
- **To explore all possible ways to parenthesize and choose the optimal one**
- To divide the matrices into equal halves and multiply

Q17.
In Matrix Chain Multiplication, which of the following represents the recursive subproblem?

- Multiplying matrices from index i to j
- Calculating the matrix product of matrices i and j
- **Finding the minimum number of scalar multiplications for a matrix chain**
- Storing the matrix dimensions

Q18.
Which approach is generally preferred for solving the Matrix Chain Multiplication problem for large numbers of matrices?

- Brute-force approach
- **Dynamic Programming**
- Greedy algorithm
- Divide and Conquer

Q19.
In the Matrix Chain Multiplication problem, what is the significance of splitting the chain at position k?

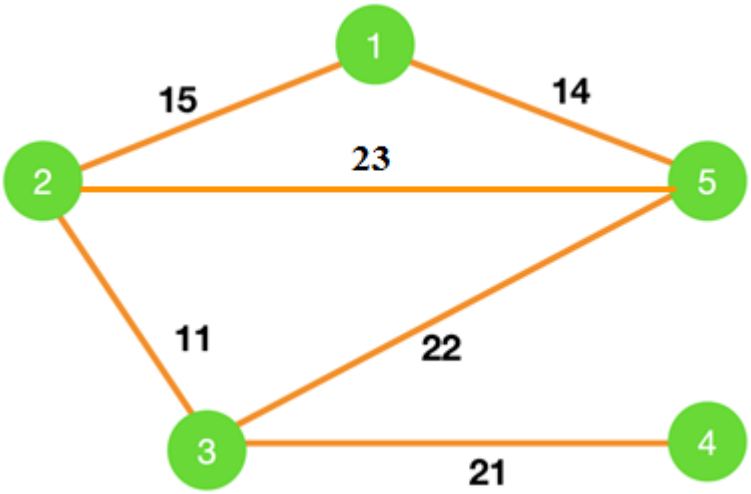
- It helps to find the matrix with the least number of rows
- **It determines the optimal order of matrix multiplication**
- It divides the chain into equal halves
- It ensures that no matrix is multiplied more than once

Q20.
What is the minimum number of matrices required for Matrix Chain Multiplication to make sense?

- 1

2
3
4

Q21.
Consider the graph shown below.



Which of the following edges form the MST of the given graph using Prim's algorithm, starting from vertex 4.

(4-3)(5-3)(2-3)(1-2)
(4-3)(3-5)(5-1)(1-2)
(4-3)(3-5)(5-2)(1-5)
(4-3)(3-2)(2-1)(1-5)

Q22.
Prim's algorithm can be efficiently implemented using _____ for graphs with greater density.

d-ary heap
linear search
fibonacci heap
binary search

Q23.
Which of the following is false about Prim's algorithm?

It is a greedy algorithm

It constructs MST by selecting edges in increasing order of their weights

It never accepts cycles in the MST

It can be implemented using the Fibonacci heap

Q24.
Worst case is the worst case time complexity of Prim's algorithm if adjacency matrix is used?

O(log V)

$O(V^2)$

$O(E^2)$

O(V log E)

Q25.
Which of the following is true about Kruskal and Prim MST algorithms?

Assume that Prim is implemented for adjacency list representation using Binary Heap and Kruskal is implemented using union by rank.

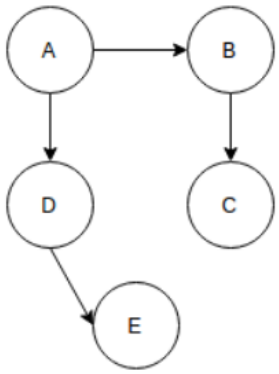
Worst case time complexity of both algorithms is the same.

Worst case time complexity of Kruskal is better than Prim

Worst case time complexity of Prim is better than Kruskal

None of the mentioned options

Q26.
What would be the DFS traversal of the given Graph by choosing the node A as the starting node?

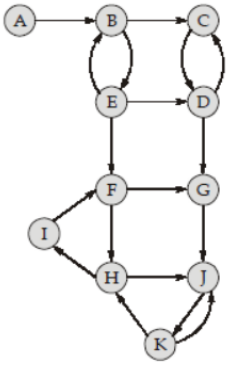


ADEBC
AEDCB
EDCBA
ABDCE

Q27.
If a vertex is deleted from a random connected graph G containing n vertices, then what is the maximum number of components in the resultant graph?

1
n
n-1
n-2

Q28.
Consider the following graph:



The number of strongly connected components of the graph are

4
5
3
2

Q29.
If, between any two vertices, there exists a path, then the graph is said to be

Strongly Connected
Connected

- Weakly Connected
- All of the mentioned options

Q30.
Let G = (V, E} be an undirected, unweighted connected graph. The diameter of G is defined as: diam(G) = t" (the length of the shortest path between u and v}

Let M be the adjacency matrix of G.Define graph G2 on the same set of vertices with adjacency matrix N, where

$$N_{ij} = \begin{cases} 1 \text{ if } M_{ij} > 0 \text{ or } P_{ij} > 0, \text{ where } P = M^2 \\ 0, \quad \text{otherwise} \end{cases}$$

Which one of the following statements is true?

- diam(G) < diam(G2)<diam(G)
- diam(G/2) < diam(G2)<diam(G)
- **diam(G/2)<[diam(G)/2)]**
- diam(G2)=diam(G)

Q31.
A 0-1 Knapsack is a well-known problem where it is desired to get the maximum total profit by placing n items (each item has some weight and associated profit) into a knapsack of capacity W. The table given below shows the weights and associated profits for 5 items, where one unit of each item is available to you. It is also given that the knapsack capacity w is 8. If the given The 0/1 knapsack problem is solved using Dynamic Programming, which one of the following will be the maximum earned profit by placing the items into the knapsack of capacity 8.

Item:1 2 3 4 5
Weight:1 2 4 5 8
Associated Profit:3 5 9 11 18

- **19**
- 18
- 17
- 20

Q32.
What does "0/1" represent in the "0/1 Knapsack problem"?

- It denotes the number of items available.

It indicates whether items are completely filled (1) or no item(0) in the knapsack.

It represents the maximum weight capacity of the knapsack.

It signifies the number of knapsacks available for packing items.

Q33.
Which of the following methods can be used to solve the Knapsack problem?

Brute-force algorithm

Recursion

Dynamic programming

Brute force, Recursion and Dynamic Programming

Q34.
In the 0-1 Knapsack Problem, if an item's weight is greater than the remaining capacity of the knapsack, what action is typically taken?

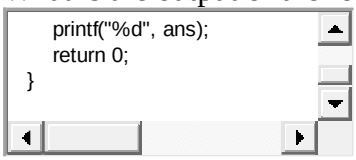
Ignore the item and move to the next one

Remove the least valuable item from the knapsack to make space

Add the fractional part of the item that can fit into the knapsack

Remove the most valuable item from the knapsack to make space

Q35.
What is the output of the following code?




```
#include <stdio.h>
int max(int a, int b) {
   return (a > b) ? a : b;
}
int knapsack_bruteforce(int W, int wt[], int val[], int n) {
   if (n == 0 || W == 0)
      return 0;
   int exclude = knapsack_bruteforce(W, wt, val, n - 1);
   int include = 0;
   if (wt[n - 1] <= W)
      include = val[n - 1] + knapsack_bruteforce(W - wt[n - 1], wt, val, n - 1);
```

```
    return max(include, exclude);
}
int main() {
    int wt[] = {10, 20, 30};
    int val[] = {60, 100, 120};
    int W = 50;
    int n = sizeof(wt) / sizeof(wt[0]);
    int ans = knapsack_bruteforce(W, wt, val, n);
    printf("%d", ans);
    return 0;
}
```

- 120
- **220**
- 180
- 100

Q36.
What is the primary objective of the Matrix Chain Multiplication problem?

- To multiply matrices in sequential order.
- To maximize the number of scalar multiplications.
- **To minimize the number of scalar multiplications.**
- To find the determinant of the matrices.

Q37.
If there are n matrices, what is the size of the dimensions array used in Matrix Chain Multiplication?

- 2n
- n-1
- **n+1**
- n

Q38.
What is the time complexity of the Matrix Chain Multiplication algorithm using dynamic programming?

- O(n log n)

- $O(n^4)$
- **$O(n^3)$**
- $O(n^2)$

Q39.
If there are 4 matrices to be multiplied, how many ways can they be parenthesized?

- 6
- **5**
- 14
- 4

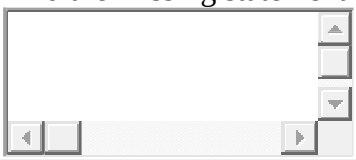
Q40.
Which of the following is used to solve the Matrix Chain Multiplication problem efficiently?

- Backtracking
- **Dynamic Programming**
- Divide and Conquer
- Greedy Algorithm

Q41.
The Floyd-Warshall algorithm for all-pair shortest paths computation is based on

- Greedy paradigm
- Divide-and-Conquer paradigm
- **Dynamic Programming paradigm**
- neither Greedy nor Divide-and-Conquer nor Dynamic Programming paradigm

Q42.
Find the missing statement in the if loop of the Floyd algorithm.



```
1
2
3
4
5
```

```
6
7
8
9
10
11
12
13
14
15
```

```
Procedure Floyd: (var A: array[1...n,1...n] of real; C: array [1...n,1...n] of real);
Var
i, j, k: integer;
begin M
for i:=l to n do
for j:=l to n do
 A[i,j]: =C[i,j]
for i:=l to n do
A[i,j ]:=0 ;
for k:= l to n do
for i:=l to n do
for j:= l to n do
if A[i,j] > A[i,k]+A[k,j] then
_____
end ;
```

A[i,j] := A[i,k]+A[k,j]

A[i,j] := A[i,j]+A[k,j]

A[i,j] := A[j,k]+A[j,i]

A[i,j] := A[i,k]+A[i,j]

Q43.

An all-pairs shortest-paths problem is efficiently solved using:

Dijkstra's algorithm

Bellman-Ford algorithm

Kruskal algorithm

Floyd-Warshall algorithm

Q44.

Floyd Warshall's Algorithm can be applied to _____.

- Undirected and unweighted graphs
- Undirected graphs
- **Directed graphs**
- Acyclic graphs

Q45.
The Floyd Warshall algorithm does not work for

- Negative edge weights
- **Negative edge weights and negative cycles**
- Negative edge weights and positive cycles
- Only positive edge weights

Q46.
What is the running time of the Floyd-Warshall Algorithm?

- Big-oh(V)
- Theta(V^2)
- Big-Oh(V^E)
- **Theta(V^3)**

Q47.
Floyd-Warshall Algorithm can be used for finding

- Single source shortest path
- Topological sort
- Minimum spanning tree
- **Transitive closure**

Q48.
What happens when the value of k is 0 in the Floyd-Warshall Algorithm?

- 1 intermediate vertex
- **0 intermediate vertex**
- N intermediate vertices

N-1 intermediate vertices

Q49.
What procedure is being followed in the Floyd-Warshall Algorithm?

Top down

Bottom up

Big bang

Sandwich

Q50.
What approach is being followed in Floyd Warshall Algorithm?

Greedy technique

Dynamic Programming

Linear Programming

Backtracking

Q51.
When constructing an OBST using dynamic programming, what does the recurrence relation primarily depend on?

The in-order traversal of the tree

The depth-first search traversal of the tree

The number of keys present in the tree

The left and right subtrees cumulative search cost

Q52.
Which algorithm is commonly used to construct an Optimal Binary Search Tree?

Backtracking

Divide and Conquer

Dynamic Programming

Greedy Algorithm

Q53.
What is the primary objective of constructing an Optimal Binary Search Tree (OBST)?

To ensure all elements are equally distributed

To minimize the expected search cost based on access probabilities

To maximize the height of the tree

To minimize the number of nodes in the tree

Q54.
What is the main difference between a regular Binary Search Tree (BST) and an Optimal Binary Search Tree (OBST)?

OBST requires more space than BST

OBST minimizes the search cost, considering probabilities of access, whereas BST does not

OBST is balanced, whereas BST is not

OBST allows duplicate keys, while BST does not

Q55.
In an OBST, what additional information is required apart from the keys and their values?

Pre-order traversal sequence

Hash values of keys

Parent-child relationships

Frequency of access (probabilities)

Q56.
The cost function of an OBST is computed based on which factor?

The number of edges in the tree

The in-order traversal of the tree

The depth of nodes weighted by their probabilities

The number of child nodes

Q57.
What happens if the frequency of some keys in an OBST is significantly higher than others?

The tree will become a complete binary tree

The search cost will increase

The tree will always remain balanced

The tree will become unbalanced

Q58.
What is the primary benefit of an OBST in practical applications?

It eliminates the need for rebalancing

It reduces the average search time compared to a regular BST

It can handle infinite keys efficiently

It always results in a perfectly balanced tree

Q59.
In the dynamic programming approach for OBST, which table is used to store the computed minimum cost for subproblems?

Adjacency matrix

Probability matrix

Cost matrix

Prefix sum table

Q60.
Which of the following is NOT a property of an Optimal Binary Search Tree?

It considers access probabilities for optimal structure

It is computed using a dynamic programming approach

It is always a balanced binary search tree

It minimizes the expected search cost

Q61. How does the Backtracking approach solve the Knapsack problem?
Correct Answer: By pruning branches that do not lead to a feasible or optimal solution.

Q62. Which strategy is commonly used in the Branch and Bound method to decide which node to explore next?
Correct Answer: Best-first search

Q63. What is the key difference between Backtracking and Branch and Bound in solving the Knapsack problem?

Correct Answer: Backtracking does not use any bounding function, while Branch and Bound does.

Q64. What is the worst-case time complexity of the Backtracking approach for the 0/1 Knapsack problem?
Correct Answer: $O(2^n)$

Q65. In the Backtracking method for the Knapsack problem, when do we decide to backtrack?
Correct Answer: When adding the next item exceeds the weight capacity.

Q66. What is the primary advantage of using the Branch and Bound approach for the Knapsack problem?
Correct Answer: It systematically explores possible solutions while pruning non-promising ones.

Q67. Which approach is generally more efficient for solving the 0/1 Knapsack problem when using Backtracking?
Correct Answer: Sorting items by value-to-weight ratio and then applying Backtracking

Q68. Which of the following methods is used to prune the search space in the Branch and Bound method for the Knapsack problem?
Correct Answer: Ignoring nodes with a weight exceeding the capacity.

Q69. Which data structure is commonly used to implement the Branch and Bound approach for solving the Knapsack problem?
Correct Answer: Priority Queue

Q70. In the Branch and Bound approach, what does the term “bounding function” refer to?
Correct Answer: A function that estimates the upper bound on the maximum value that can be achieved from a node.