CS6402 DESIGN AND ANALYSIS OF ALGORITM

2mark Questions with Answer

UNIT-III

DYNAMIC PROGRAMMING AND GREEDY TECHNIQUE

1. What is Dynamic programming?

Dynamic programming is an algorithm design technique for solving problem with overlapping subprograms. The smaller subprograms are solved only once and recording the results in a table from which the solution to the original problem is obtained.

Procedure;

1. Solution to the problem is carried out by results of sequence of decisions.
2. Enumerate all decision sequences and then pick out the best.
3. Optimal sequence of decision is obtained.

Example:

1. Computing Binomial coefficient
2. Knapsack Problem and Memory Functions
3. Optimal Binary Search Trees
4. Warshall's and Floyd's Algorithms

2. What is greedy method? / Write a short note on greedy approach. ®

The greedy method is the most straight forward design, which is applied for change making problem**.** The greedy technique suggests constructing a solution to an optimization problem through a sequence of steps, each expanding a partially constructed solution obtained so far, until a complete solution to the problem is reached. On each step, the choice made must be feasible, locally optimal and irrevocable.

Some greedy techniques are:

1. Prim's Algorithm
2. Kruskal's Algorithm
3. Dijkstra's Algorithm
4. Huffman Trees and Codes

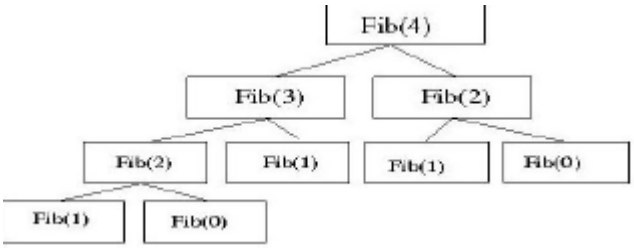
3. How do you find the nth Fibonacci number? Write the algorithm.

Starting from x(0) = 0 and x(1) = 1, the series progresses infinitely as: 0,1,1,2,3,5,8,13...

```
// nth Fibonacci number finding algorithm by Dynamic programming
Fib(n)
        if (n == 0)
                return 0;
        if (n == 1)
                return 1;
        else
                return Fib(n-1) + Fib(n-2);
```

Let T(n) be the complexity of the above algorithm. T(n) can be written in the form of

a recurrence relation as: T(n)=T(n−1)+T(n−2) => $\lambda n=\lambda^{n-1}+\lambda^{n-2}$
=> $\lambda^2 = \lambda+1$. Solving this, we get $\lambda = (1+\sqrt{5})/2$
=> $\lambda^n = \lambda^{n-1}+\lambda^{n-2}$. Solving this, we get $\lambda = ((1+\sqrt{5})/2)^n$



4. List the proving methods of Greedy Technique.

Proofs for Greedy Technique are listed below:

1. The first way is one of the common ways by mathematical induction.
2. The second way to prove optimality of a greedy algorithm is to show that on each step it does at least as well as any other algorithm could in advancing toward the problem's goal.
3. The third way is simply to show that the final result obtained by a greedy algorithm is optimal based on the algorithm's output rather than the way it operates.

5. List the advantage of greedy algorithm.

1) Greedy algorithm produces a feasible solution.
2) Greedy method is very simple to solve a problem.
3) Greedy method provides an optimal solution directly.

6. Write an algorithm to compute Binomial Coefficient.
Write an algorithm to compute c(n,k) using dynamic program.

Algorithm Binomial(n, k)

for $i \leftarrow 0$ to n do // fill out the table row wise
  for $i = 0$ to min(i, k) do
    if j==0 or j==i then $C[i, j] \leftarrow 1$ // Initial Condition
    else $C[i, j] \leftarrow C[i\text{-}1, j\text{-}1] + C[i\text{-}1, j]$ // recursive relation
  return $C[n, k]$

$(a + b)^n = C(n, 0)a^n b^0 + \ldots + C(n, k)a^{n-k}b^k + \ldots + C(n, n)\, a^0 b^n$

7. Define Adjacency matrix.

An Adjacency matrix A = {aij} of a directed graph is the boolean matrix that has 1 in its ith row and jth column if and only if there is a directed edge from the ith vertex to the jth vertex.

8. Define Transitive closure

The transitive closure of a directed graph with n vertices can be defined as the n x n boolean matrix T = {tij }, in which the element in the ith row and the jth column is 1 if there exists a nontrivial path (i.e., directed path of a positive length) from the ith vertex to the jth vertex; otherwise, tij is 0.

9. Define Distance matrix

A distance matrix is a matrix (two-dimensional array) containing the distances, taken pairwise, between the vertices of graph.

10. Define warshall's algorithm?

Warshall's algorithm is an application of dynamic programming technique, which is used to find the transitive closure of a directed graph.

The transitive closure of a directed graph with n vertices can be defined as the n x n boolean matrix T = {tij }, in which the element in the ith row (1≤i≤n≥) and the jth column(1≤j≤n) is 1 if there exists a nontrivial path (i.e., directed path of a positive length) from the ith vertex to the jth vertex; otherwise, tij is 0.

11. What is the complexity of warshall's algorithm?

Warshall's algorithm's time efficiency:

ALGORITHM Warshall(A[1..n, 1..n])
//Implements Warshall's algorithm for computing the transitive closure
//Input: The adjacency matrix A of a digraph with n vertices
//Output: The transitive closure of the digraph
$R^{(0)} \leftarrow A$
for k←1 to n do
    for i ←1 to n do
        for j ←1 to n do
            R(k)[i, j]←R(k−1)[i, j] or (R(k−1)[i, k] and
        R(k−1)[k, j])
return $R^{(n)}$
The algorithm requires 3 nested for loops of size n = n*n*n
The order of complexity = $\Theta(n^3)$.

Warshall's algorithm's Space efficiency

Space for a nxn adjacency matrix = n2 locations (. i.e matrix size0
For control variables i, j, k = 3
Total space = n^2 +3 = $\Theta(n^2)$

12. Define Floyd's algorithm?

Floyd's algorithm is an application, which is used to find all the pairs shortest paths problem. Floyd's algorithm is applicable to both directed and undirected weighted graph, but they do not contain a cycle of a negative length.

13. What is the complexity of Floyd's algorithm?

Floyd's algorithm's time efficiency:

ALGORITHM Floyd(W[1..n, 1..n])
//Implements Floyd's algorithm for the all-pairs shortest-paths problem
//Input: The weight matrix W of a graph with no negative-length cycle
//Output: The distance matrix of the shortest paths' lengths
D ←W //is not necessary if W can be overwritten
for k←1 **to** n **do**

```
for i ←1 to n do
        for j ←1 to n do
                D[i, j ]←min{D[i, j ], D[i, k]+ D[k, j]}
return D
```

The algorithm requires 3 nested for loops of size n = n*n*n
The order of complexity = Θ(n3).

Floyd's algorithm's Space efficiency

Storage Space for a nxn weighted matrix = n^2 locations(. i.e matrix size0
For control variables i, j, k = 3
Total space = n^2 +3 = Θ(n^2)

14. Differentiate Floyd and warshall.

Factors	Warshall	Floyd
Input	Directed graph	Weighted directed graph
Matrix	Adjacency matrix	Distance matrix
Computation	Transitive closure	Shortest path distance matrix
Purpose	All-Pairs Path Existence Problem	All-Pairs Shortest Paths Problem

15. Differentiate prim's and kruskal's algorithm.

prim's algorithm	Kruskal's algorithm
This algorithm is for obtaining minimum spanning tree by selecting the adjacent vertices of already selected vertices.	This algorithm is for obtaining minimum spanning tree but it is not necessary to choose the adjacent vertices of already selected vertices.
Sorting of all edge weights is not required.	Sorting of all edge weights is must to construct minimum spanning tree.
The efficiency of Prim's algorithm is O(|E| log |V |)	The efficiency of Kruskal's algorithm is O(|*E*| log |*E*|).
Prim's algorithm is significantly faster in the limit when you've got a really dense graph with many edges than vertices.	Kruskal performs better in typical situations (sparse graphs) because it uses simpler data structures.

16. Define Binary Search Trees.

Binary Search tree is a binary tree in which each internal node x stores an element such that the element stored in the left subtree of x are less than or equal to x and elements stored in the right subtree of x are greater than or equal to x.

17. Define Optimal Binary Search Trees.

An optimal binary search tree (BST), sometimes called a weight-balanced binary tree, is a binary search tree which provides the smallest possible search time for a given sequence of accesses (or probabilities of searching elements). Optimal BSTs are generally

divided into two types: static and dynamic.

In the static optimality problem, the tree cannot be modified after it has been constructed. In the dynamic optimality problem, the tree can be modified at any time, typically by permitting tree rotations.

18. What is Catalan number? How it is related with OBST?

The total number of binary search trees with *n* keys is equal to the *n*th Catalan Number,

$$c(n) = \frac{1}{n+1}\binom{2n}{n} \quad \text{for } n > 0, \quad c(0) = 1$$

c(n)=(2n)!/(n+1)!n!

19. What is the complexity of Optimal Binary Search Trees?

The algorithm's space efficiency is clearly quadratic, ie, **:** Θ(*n*3). The time efficiency of this version of the algorithm is cubic. It is Possible to reduce the running time of the algorithm to Θ(n2)

20. Define Knapsack Problem by dynamic programming.

- Given n items of known weights $w_1, \ldots, w_n$ and values $v_1, \ldots, v_n$ and a knapsack of capacity W, find the most valuable subset of the items that fit into the knapsack.
- Assume that all the weights and the knapsack capacity are positive integers; the item values do not have to be integers.
- 0 / 1 knapsack problem means, the chosen item should be either null or whole.
- To design a dynamic programming algorithm, we need to find a recursive relation from the smaller sub problems to larger problem.

21. Define Memory functions.

Dynamic programming (DP) solves problems that have a recurrence relation. Using the recurrences directly in a recursive algorithm is a top-down technique. It has the disadvantage that it solves common sub problem multiple times. This leads to poor efficiency, exponential. The dynamic programming technique is bottom-up, and solving all the sub-problems only once. It is natural to try to combine the strengths of the top-down and bottom-up approaches. The goal is to get a method that solves only subproblems that are necessary and does so only once. Such a method exists; it is based on using memory functions**.**

22. State the uses of memory functions to solve knapsack problem

- Memory function solves a given problem in top-down manner but maintains a table used in a bottom-up DP algorithm.
- It does less number of Computations.
- It does computation in less time
- It requires minimum storage space (less memory).

23. List out some Greedy Techniques and their uses.

- Prim's algorithm: To find minimum spanning tree.
- Kruskal's Algorithm: To find minimum spanning tree.

- Dijkstra's Algorithm: To find single source shortest path.
- Huffman Trees: To minimize encode by frequency of symbols.

24. Define Directed graph.

A directed graph (or digraph) is a graph, or set of vertices connected by edges, where the edges have a direction associated with them.

25. What is Prim's algorithm?

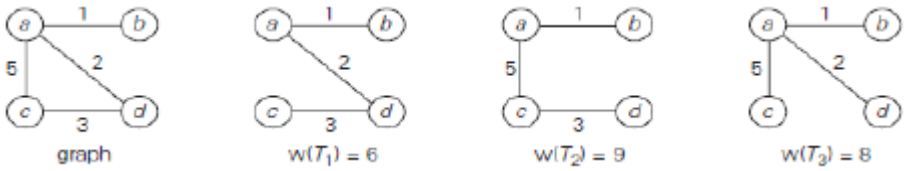
Prim's algorithm is a greedy and efficient algorithm, which is used to find the minimum spanning tree of a weighted connected graph.

26. What is spanning tree?

A ***spanning tree*** of an undirected connected graph is its connected acyclic subgraph (i.e., a tree) that contains all the vertices of the graph.
Let G= {V, E} be an undirected connected graph. A sub graph G'= {V, E'} of G is a spanning tree of G, if t is a tree.

27. What is minimum spanning tree?

A minimum spanning tree is its spanning tree of the smallest weight, where the weight of a tree is defined as the sum of the weights on all its edges. The minimum spanning tree problem is the problem of finding a minimum spanning tree for a given weighted connected graph.



W(T1)=6 , i.e., minimum spanning tree cost is 6.

28. List the applications of minimum spanning tree?

Spanning tree are used to obtain an independent set of circuit equations for an electric network. Another application of spanning tree arises from the property that a spanning tree is a minimal sub graph G' of G such that V(G')=V(G) and G' is connected.

29. What is Kruskal's Algorithm?

Kruskal's algorithm is another greedy algorithm for the minimum spanning tree problem. Kruskal's algorithm constructs a minimum spanning tree by selecting edges in increasing order of their weights provided that the inclusion does not create a cycle. Kruskals algorithm provides a optimal solution.

30. What is Dijkstra's Algorithm? / What is the purpose of Dijkstra's Algorithm?

Dijkstra's algorithm solves the single source shortest path problem of finding shortest paths from a given vertex(the source), to all the other vertices of a weighted graph or digraph.

Dijkstra's algorithm provides a correct solution for a graph with non negative weights.

31. What is path compression?

1. The better efficiency can be obtained by combining either variation of quick union with path compression.
2. Path compression makes every node encountered during the execution of a find operation point to the tree's node.

32. What is Huffman Trees?

A Huffman tree is binary tree that minimizes the weighted path length from the root to the leaves containing a set of redefined weights. The most important application of Huffman trees are Huffman code.

33. What is Huffman code?

A Huffman code is a optimal prefix tree variable length encoding scheme that assigns bit strings to characters based on their frequencies in a given text.

34. Define prim's algorithm.

Prim's algorithm is greedy and efficient algorithm, which is used to find the minimum spanning tree of weighted connected graph.

35. How efficient is prim's algorithm?

The efficiency of the prim's algorithm depends on data structure chosen for the graph. If a graph is represented by its adjacency lists and the priority queue is implemented as a minheap, the running time of the algorithm is O(|E| log |V |) in a connected graph.

36. Define Kruskal's algorithm.

Kruskal's algorithm is a minimum-spanning-tree algorithm where the algorithm finds an edge of the least possible weight that connects any two trees in the forest.It is a greedy **algorithm** in graph theory as it finds a minimum spanning tree for a connected weighted graph adding increasing cost arcs at each step.

37. What is path compression?

The better efficiency can be obtained by combining either variation of quick union with path compression. Path compression makes every node encountered during the execution of a find operation point to the tree's node.

38. Define Dijkstra's Algorithm?

Dijkstra's algorithm solves the single source shortest path problem of finding shortest paths from a given vertex (the source), to all the other vertices of a weighted graph or digraph. Dijkstra's algorithm provides a correct solution for a graph with non negative weights.

39. Define Huffman trees?

A Huffman tree is binary tree that minimizes the weighted path length from the root to the leaves containing a set of predefined weights. The most important application of Huffman trees are Huffman code.

40. Write short note on degree of tree.

- The number of subtrees of a node is called the *degree* of the node. In a binary tree, all nodes have degree 0, 1, or 2.
- A node of degree zero is called a *terminal node* or *leaf node*.
- A non-leaf node is often called a *branch node*.
- The degree of a tree is the maximum degree of a node in the tree. A binary tree is degree 2.

41. Write short note on path and height of tree.

- A *directed path* from node n_1 to n_k is defined as a sequence of nodes n_1, n_2, ..., n_k such that n_i is the parent of n_i+1 for 1 <= i < k. An undirected path is a similar sequence of undirected edges. The length of this path is the number of edges on the path, namely *k* –1 (i.e., the number of nodes – 1). There is a path of length zero from every node to itself. Notice that in a binary tree there is exactly one path from the root to each node.
- The *level* or *depth* of a node with respect to a tree is defined recursively: the level of the root is zero; and the level of any other node is one higher than that of its parent. Or to put it another way, the level or depth of a node n_i is the length of the unique path from the root to n_i.
- The *height* of n_i is the length of the longest path from n_i to a leaf. Thus, all leaves in the tree are at height 0.
- The *height of a tree* is equal to the height of the root. The *depth of a tree* is equal to the level or depth of the deepest leaf; this is always equal to the height of the tree.
- If there is a directed path from n_1 to n_2, then n_1 is an ancestor of n_2 and n_2 is a descendant of n_1.

42. What do you mean by Huffman code?

A Huffman code is a optimal prefix tree variable length encoding scheme that assigns bit strings to characters based on their frequencies in a given text.

43. What is meant by compression ratio in Huffman code?

Huffman's code achieves the compression ratio, which is a standard measure of compression algorithms effectiveness of

(3-2.25)/3*100 = 0.75/3*100 .

= 0.25 *100

= 25%.

44. List the advantage of Huffman's encoding?

- Huffman's encoding is one of the most important file compression methods.
- It is simple
- It is versatility
- It provides optimal and minimum length encoding

45. What is dynamic Huffman coding / encoding?

In dynamic Huffman coding, the coding tree is updated each time a new character is read from the source text. Dynamic Huffman n codings used to overcome the drawback of simplest version.

46. What is articulation point?

A vertex of a connected graph G is said to be in articulation point, if its removal with all edges incident to it breaks the graph into disjoint pieces.

47. What is a Biconnected Graph?

A biconnected graph is a connected graph on two or more vertices having no articulation vertices. Such graphs are also sometimes called blocks or non-separable graphs. A biconnected undirected graph is a connected graph that is not broken into disconnected pieces by deleting any single vertex (and its incident edges). A biconnected directed graph is one such that for any two vertices v and w there are two directed paths from
v to w which have no vertices in common other than v and w.

48. Tabulate the complexity of graph algorithms.

Algorithm	**Time complexity**	**Space complexity**
Warshall's Algorithm	$\Theta(n^3)$	$\Theta(n^2)$
Floyd's Algorithm	$\Theta(n^3)$	$\Theta(n^2)$
Optimal Binary Search Trees	$\Theta(n^3)$ $\Theta(n^2)$	$\Theta(n^2)$
knapsack problem (n-items, W-weight)	O(nW)	O(nW)
Prim's Algorithm	O(|E| log |V |)	$\Theta(n^2)$
Kruskal's Algorithm	O(|E| log |E |)	$\Theta(n^2)$
Dijkstra's Algorithm	O(|E| log |V|)	Θ ($\|V\|^2$)
Huffman tree code Algorithm	O(*n log n*)	O(n)