Design and Analysis of Algorithms

1. Define Space and Time complexity.

   The space complexity of an algorithm is the amount of memory it needs to run to complete. The time complexity of an algorithm is the amount of computer time it needs to run to completion.

2. If $f(n) = a_m n^m + \ldots\ldots\ldots + a_1 n + a_0$, then prove that $f(n) = O(n^m)$

   $f(n) \leq \sum |a_i| n^i$
   $\leq n^m \sum |a_i| n^{i-m}$
   $\leq n^m \sum |a_i|$ for $n \geq 1$. So $f(n) = O(n^m)$

3. State the weighting, collapsing rules in sets.

   Weighting rule: if the number of nodes in the three with root i is less than the number in the tree with root j, then make j the parent of i; otherwise make I the parent of j.
   Collapsing rule: if j is a node on the path from I to its root and p[i] ≠ root[i] then set p[j] to root[i].

4. Define Heaps.

   A max(min) heap is a complete binary tree with the property that the value at each node is at least as large as (as small as) the vales at its children (if they exist). Call this property the heap property.

5. Explain Weighted union and collapsing find algorithms

   Weighted Union Algorithm:

```
Algorithm WeightedUnion(i, j)
{
        temp := p[i] + p[j];
        if(p[i] > p[j]) then
        {
                p[ I ] := j;
                p[ j ] := temp;
        }
        Else
        {
                p[ j ] := I;
                p[ I ] := temp;
        }
```

```
}
```

Collapsing Find Algorithm:

```
Algorithm CollapsingFind(i)
{
        r : = i;
        while(p[ r ] > 0) do r := p[ r ];
        while( i ≠ r) do
        {
                s := p[ i ];
                p[ I ] := r;
                i := s;
        }
        return r;
}
```

6. Write the control abstraction for divide and conquer.

```
Algorithm DandC(P)
{
        If Small(P) then return S(P);
        Else
        {
                Divide P into smaller instances P1, P2, …. , Pk, k ≥ 1;
                Apply DandC to each of these subproblems;
                Return Combine(DandC(P1), DandC(P2), ….. ,DandC(Pk));
        }
}
```

7. Solve: T(n) = a.T(n/b) + f(n), by considering the case in which a = 2 and b = 2. Let T(1) = 2 and f(n) = n.

$T(n) = 2.T(n/2) + n$
$= 2[2.T(n/4) + n/2] + n$
$= 4[2T(n/8) + n/4] + 2n$
$= 8T(n/8) + 3n$ ……..

In general, we see that $T(n) = 2^i.T(n/2^i) + i.n$, for and $\log_2^n \geq i \geq 1$.

Thus $T(n) = n . \log_2^n + 2n$

8. Define the terms Feasible solution, optimal solution and objective function.

Given problem have ***n*** inputs and require obtaining a subset that satisfies some constraints. Any subset that satisfies these constraints is called a ***feasible solution***. We need to find the feasible solution that either maximizes or minimizes a given ***objective function***. A feasible solution that does this is called an ***optimal solution***.

9. Write the control abstraction for greedy method

Control abstraction for greedy method:

```
Algorithm Greedy(a, n)
{
        Solution := 0;
        for i := 1 to n do
        {
                x: = Select(a);
                if Feasible(solution, x) then
                        Solution := Union(solution, x);
        }
        return solution;
}
```

10. Define Minimum Spanning Tree.

Let G = (V, E) be an undirected connected graph. A subgraph t = (V, E') of G is a spanning tree of G iff t is a tree.

A *minimum spanning tree (MST)* of an edge-weighted graph is a spanning tree whose weight (the sum of the weights of its edges) is no larger than the weight of any other spanning tree.

11. Explain The Optimal Storage on Tapes problem.

There are n programs that are to be stored on a computer tape of length *l*. associated with each program *i* is a length l_i, clearly, all programs can be stored in the tape if and only if the sum of the lengths of the programs is at most *l*. we assume that whenever a program is to be retrieved from this tape, the tape is initially positioned at the front. Hence, if the programs are stored in the order $I = i_1, i_2, \ldots, i_n$, the time t_j needed to retrieve program i_j is proportional to $\sum_{1 \le k \le j} l_{ik}$. if all programs are retrieved equally ofter, then the expected or ***mean retrieval time*** (MRT) is $(1/n)\sum_{1 \le j \le n} t_j$. in the optimal storage on tape problem we are required to find a permutation for the n programs so that when they are stored on the tape in this order the MRT is minimized.

12. Explain Traveling Salesperson Problem.

Let G = (V, E) be a directed graph with edge costs cij. The variable C_{ij} is defined such that $C_{ij} > 0$ for all I and j and $C_{ij} = \infty$ if (I, j) not belongs to E. Let $|V| = n$ and assume $n > 1$. A tour of G is a directed simple cycle that includes every vertex in V. the cost of tour is the sum of the cost of the edges on the tour. The traveling salesperson problem is to find a tour of minimum cost.

13. Differentiate between Greedy and Dynamic programming approaches.

In Greedy approach consider greedy choice property, we can make whatever choice seems best at the moment and then solve the sub-problems that arise later. The choice made by a greedy algorithm may depend on choices made so far but not on future choices or all the solutions to the sub-problem. It iteratively makes one greedy choice after another, reducing each given problem into a smaller one. **In other words, a greedy algorithm never reconsiders its choices. This is the main difference from dynamic programming, which is exhaustive and is guaranteed to find the solution**. After every stage, dynamic programming makes decisions based on all the decisions made in the previous stage, and may reconsider the previous stage's algorithmic path to solution.

14. Define articulation point and bi-connected graph.

A vertex v in a connected graph G is an articulation point if and only if the deletion of vertex v together with all edges incident to v disconnects the graph into two or more nonempty components.
A graph G is bi-connected if and only if it contains no articulation points.

15. What are the differences between Backtracking and Branch and Bound techniques?

Backtracking
[1] It is used to find all possible solutions available to the problem.
[2] It traverse tree by DFS(Depth First Search).
[3] It realizes that it has made a bad choice & undoes the last choice by backing up.
[4] It search the state space tree until it found a solution.
[5] It involves feasibility function.

Branch-and-Bound
[1] It is used to solve optimization problem.
[2] It may traverse the tree in any manner, DFS or BFS.
[3] It realizes that it already has a better optimal solution that the pre-solution leads to so it abandons that pre-solution.
[4] It completely searches the state space tree to get optimal solution.
[5] It involves bounding function.

16. What is the objective of m-colorability optimization problem?

Let G be a graph and m be a given positive integer. We want to discover wether the nodes of G can be colored in such a way that no two adhacent nodes have the same color yet only m colors are used. The m-colorability optimization problem asks for the smallest integer m for which the graph G can be colored.

17. Explain what are explicit and implicit constraints of 8-Queens problem.

8-Queen problem is to place eight queens on an 8x8 chessboard so that no two attack. That is so that no two of them are on the same row, colomn, or diagonal. Let us number the rows and columns of the chessboard 1 through 8. The queens can also be numbered 1 through 8. Since each queen must be on a different rwo, we can without loss of generality

assume queen i is to be placed on row i. all solutions to the 8-queen problem can therefore be represented as 8-tuples ($x_1,\ldots,x_8$), where x_i is the column on which queen *i* is placed. The explicit constraints using this formulation are $S_i = \{1,2,3,4,5,6,7,8\}$. The implicit constraints for this problem are that no two xi's can be the same and no two queens can be on the same diagonal.

18. What is meant by lower bound theory.

    If two algorithm for solving the problem where discovered and their times differedby an order of magnitude, the one with the smaller order was generally regarded as superior There are 3 technique to solve or compute the lower bound theory:
    1>Comparison trees
    2>Oracle and adversary argument
    3>Lower bound through reductions

19. Differentiate FIFO, LIFO branch-and-bound.

    In LIFO BnB Depth First Search Schema is used, Whereas in FIFO BnB Breadth First Search schema is used.

    (First point in description: in case DFS and BFS is not known to the Questioner) In LIFO BnB- newly formed Live Node is explored before completely exploring the current E-Node , while in FIFO BnB all the child nodes of E-Node are formed first and then Live nodes are explored further.

    In LIFO BnB State Space Tree proceeds laterally while in FIFO it proceeds horizontally.

20. Define the terms cliques, node cover.

    The clique cover problem (also sometimes called partition into cliques) is the problem of determining whether the vertices of a graph can be **partitioned** into k cliques.
    An alternative definition of the intersection number of a graph G is that it is the smallest number of cliques in G (complete subgraphs of G) that together **cover** all of the edges of G. A set of cliques with this property is known as a clique edge cover or edge clique cover.

21. What is a Hamiltonian cycle?

    A Hamiltonian path (or traceable path) is a pathin an undirected or directed graph that visits each vertex exactly once. A Hamiltonian cycle(or Hamiltonian circuit) is a Hamiltonian path that is a cycle.

22. Differentiate between NP-hard and NP-complete.

    A decision problem *H* is NP-hard when for every problem *L* in NP, there is a polynomial-time reduction from *L* to *H* An equivalent definition is to require that every problem *L*in NP can be solved in polynomial time by an oracle machine with an oracle

for *H*. Informally, we can think of an algorithm that can call such an oracle machine as a subroutine for solving *H*, and solves *L* in polynomial time, if the subroutine call takes only one step to compute.

A decision problem C is NP-complete if:
1. C is in NP, and
2. Every problem in NP is reducible to C in polynomial time.[1]

C can be shown to be in NP by demonstrating that a candidate solution to C can be verified in polynomial time.

23. What is satisfiability?

Let x1, x2, … denote Boolean variables. Let ~xi denote the negation of xi. A literal is either a variable or its negation. A formula in the propositional calculus is an expression that can be constructed using literals and the operations and & or operators. The satisfiability problem is to determine whether a formula is true for some assignment of truth values to the variables.

24. Define Reducibility.

Let L1 and L2 be problems. Problem L1 reduces to L2 if and only if there is a way to solve L1 by a deterministic polynomial time algorithm using a deterministic algorithm that solves L2 in polynomial time.

25. Define Live node, E- node, Dead node.

- Live node is a node that has been generated but whose children have not yet been generated.
- E-node is a live node whose children are currently being explored. In other words, an E-node is a node currently being expanded.
- Dead node is a generated node that is not to be expanded or explored any further. All children of a dead node have already been expanded.
- Branch-and-bound refers to all state space search methods in which all children of an E-node are generated before any other live node can become the E-node.