UNIT - V

NP Hard and NP Complete Problems: Basic Concepts

NP Hard Graph Problems: Clique Decision Problem (CDP), Traveling Salesperson Decision Problem (TSP)

NP Hard Scheduling Problems: Job Shop Scheduling

NP Hard and NP Complete Problems: Basic Concepts

Decision Problems: Any problem for which the answer is either yes or no is called decision problem. The algorithm for decision problems is called decision algorithm.

Ex: Sum of subsets problem.

Optimization Problems: Any problem that involves the identification of an optimal value (maximum or minimum) is called optimization problem.

Ex: Knapsack problem, travelling salesperson problem.

Polynomial Time Algorithms:

A polynomial time algorithm is an algorithm whose running time grows at most as a polynomial function of the input size. In other words, if the size of the input is ‘n’, the maximum time taken by the algorithm is some polynomial expression of n, like $O(n)$,$O(n^2)$,$O(n^3)$, $O(n \log n)$ etc.

Ex: Linear search, quick sort, all pairs shortest path, etc.

Non- Polynomial Time Algorithms:

Non-polynomial time algorithms are algorithms whose running time grows faster than any polynomial function of the input size n. They are generally considered inefficient and impractical for large inputs due to their exponential growth rate like $O(n!)$, $O(2^n)$, etc.

Ex: Sum of subsets, Travelling salesman problem, etc.

NP (Nondeterministic Polynomial time):

NP (Nondeterministic Polynomial time) is a class of decision problems (problems with a yes/no answer) in computational complexity theory. It refers to the set of problems for which a solution can be verified in polynomial time.

Ex: Sub of subsets

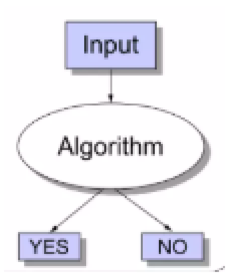
Deterministic and Non-Deterministic Algorithms:

Class P:

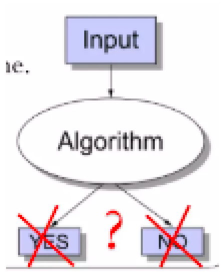
P: P time problems contains all decision problems that can be solved deterministically using a polynomial time. P is the class of computational problems that are efficiently solvable.

Ex: Searching, Minimum Cost Spanning Tree (Prims and Kruskals)

Deterministic Algorithms: The algorithm in which every operation is uniquely defined is called deterministic algorithm.



Non-Deterministic Algorithms: The algorithm in which the operations are not uniquely defined but are limited to specific set of possibilities for every operation, such an algorithm is called non-deterministic algorithm.



The non-deterministic algorithms use the following functions.

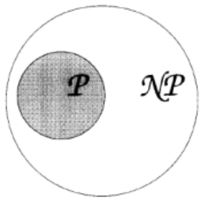
1. Choice: Arbitrarily chooses one of the elements from the given set.
2. Failure: It indicates an unsuccessful completion.
3. Success: It indicates successful completion.

Ex: Subset Sum Problem:

- Given a set of integers, does a subset exist that sum to a particular target value?
- Verifying a proposed subset (checking that its elements add up to the target value) can be done in polynomial time, even though finding that subset might be much harder.

A non-deterministic algorithm terminates unsuccessfully it and only if there exists no set of choices leading to a success criterion. Whenever there is a set of choices that leads to a successful completion, then one such set of choices is selected and the algorithm terminates successfully.

A non-deterministic algorithm is considered as a subset for deterministic algorithm. i.e. the deterministic algorithm are a special case of non-deterministic algorithms.



Satisfiability Problem (SAT): It is used to determine whether a Boolean formula is true or not.

Ex:

- F = A or B, is satisfiable because A= True and B= False makes F = True
- F = A and B, is unsatisfiable because A= True and B= False makes F = False

Reducibility:

A problem A is said to be reducible to another problem B if and only if;

- There exists a polynomial-time algorithm that transforms instances of A into instances of B.
- A solution to the transformed instance of B can be used to solve the original instance of A.

NP Hard and NP Complete:

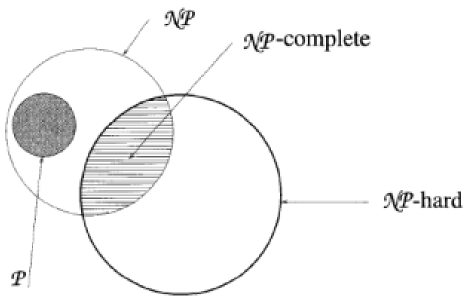
Let P denote the set of all decision problems solvable by deterministic algorithm in polynomial time. NP denotes set of decision problems solvable by nondeterministic algorithms in polynomial time. The nondeterministic polynomial time problems can be classified into two classes. They are

1. NP Hard
2. NP Complete

NP-Hard: Any nondeterministic polynomial time problem is satisfiable and reducible then the problem is said to be NP-Hard.

Ex: Halting Problem, Flow shop scheduling problem

NP-Complete: A problem L is NP-Complete iff L is NP-Hard and L belongs to NP (nondeterministic polynomial).

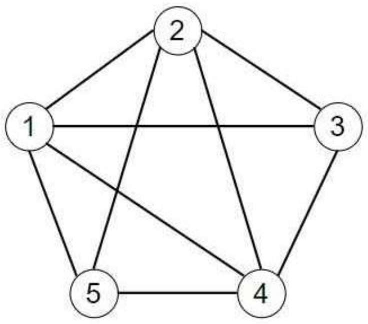


Clique Decision Problem:

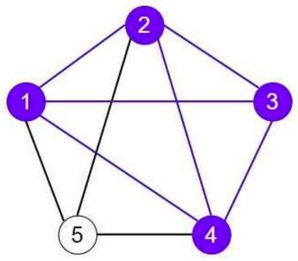
Let G be a non-directed graph consisting of a set of vertices V and set of edges E. A subgraph that forms a complete graph is called a clique. The size of the clique is the number of vertices in it. The max clique problem is an optimization problem that has to determine whether graph has a clique of size atleast k for some integer k.

A graph is said to be complete graph if every pair of distinct vertices are connected by an edge. The number of edges in a complete graph is n9n-1)/2, where 'n' is the number of vertices.

Ex:



The vertex 5 is not connected to vertex 3. Hence, it is said that this graph not a complete graph. But a subgraph {1, 2, 3, 4} is said to be a complete graph in which all the pairs are connected by an edge.



A subgraph that forms a complete graph is called a **clique**. In this case, using the brute force algorithm, **we have identified a 4-clique within the 5-vertex graph**. This means that the subgraph formed by 4 vertices is fully connected, with every pair of vertices having an edge between them, making it a complete graph.

We can find that the 4 vertices are fully connected, which means it forms a **maximum clique.** This maximum clique includes all four vertices. Additionally, there are several fully connected 3-vertex combinations, such as **{(1, 2, 3), (2, 3, 4), (1, 2, 4), (1, 3, 4)}**. Furthermore, we can identify fully connected pairs of 2-vertices **{(1, 2), (2, 3), (3, 4), (1, 3), (1, 4), (2, 4)}.**

Clique Optimization Problem (COP) Statement

Given an undirected graph **G = (V, E)**, the objective of the Clique Optimization Problem is to **find the largest clique in the graph**. A clique is a subset of vertices where every two distinct vertices are connected by an edge. The goal is to maximize the number of vertices in this fully connected subset.

The **Clique Optimization Problem** belongs to the class **NP-Hard**. This means that finding the largest clique is difficult (no known polynomial-time algorithm exists to solve it).

Clique Decision Problem (CDP) Statement

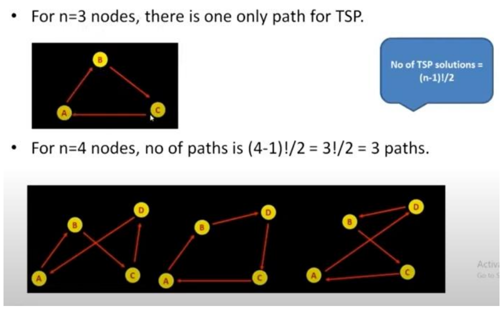
Given an undirected graph **G = (V, E)** and an integer **k**, the problem is to determine whether there exists a clique of size k in the graph. In other words, the task is to decide if there is a subset of k vertices such that every pair of vertices in this subset is connected by an edge.

This problem asks for a **yes** or **no** answer to whether the graph contains a clique of at least **k** vertices and is classified as **NP-complete.**

Travelling Salesman Decision Problem:

TSP is a NP-Hard problem. Here TSP NP-Hard problem is converted into NP-Complete problem. For checking all the permutation, the time complexity is O(n!). In dynamic programming the complexity is O $(n^2 2^n)$.

Ex;



- For n=3 nodes, there is one only path for TSP.

No of TSP solutions = (n-1)!/2

- For n=4 nodes, no of paths is (4-1)!/2 = 3!/2 = 3 paths.

If n=31 then no. of paths = 31!/2. It is impossible to find all possible solutions and then get the optimal solution. Hence, it is NP Hard.

For solving this problem reduction of NP-Hard problem i.e. NP-Complete. Hamiltonian cycle problem is reduced to an instance of TSP.

1. Convert unweighted graph to weighted graph.
2. If the graph ‘G’ has a Hamiltonian cycle, then it is same for TSP.
3. If the graph ‘G’ does not have Hamiltonian cycle,, then there can be no such TSP.

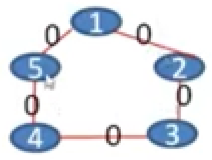
To form a TSP:

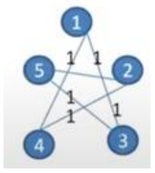
Step1 : Construct a complement graph G’. It takes O(n) time.

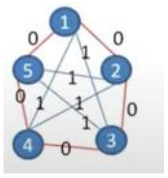
Step2 : Construct a complete graph by combining G and G’.

Define Cost function: C(i,j) = 0 if (i,j) E G and 1, if(i, j) E G’.

Graph G: **Compliment Graph G’:** **G&G’**







	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	0	0
3	0	1	0	1	0
4	0	0	1	0	1
5	1	0	0	1	0

	1	2	3	4	5
1	0	0	1	1	0
2	1	0	0	1	1
3	1	0	0	0	1
4	1	1	0	0	0
5	0	1	1	0	0

	1	2	3	4	5
1	0	1	1	1	1
2	1	0	1	1	1
3	1	1	0	1	1
4	1	1	1	0	1
5	1	1	1	1	0

T(i,j)=0 T(i,j)∈G

T(i,j)=1 T(i,j)∈G’

Now, suppose that a Hamiltonian cycle h exists in G. It is clear that the cost of each edge in h is 0 in G' as each edge belongs to E. Therefore, h has a cost of 0 in G'. Thus, if graph G has a Hamiltonian cycle, then graph G' has a tour of 0 cost.

Conversely, we assume that G' has a tour h' of cost at most $\mathbf{0}$. The cost of edges in E' are **0** and **1** by definition. Hence, each edge must have a cost of **0** as the cost of h' is **0**. We therefore conclude that h' contains only edges in E.

We have thus proven that G has a Hamiltonian cycle, if and only if G' has a tour of cost at most **0**. TSP is NP-complete.

NP Hard Scheduling Problems: Job Shop Scheduling

In a job shop there are 'n' jobs and 'm' machines are available. Each job has a unique order of visit for their completion. A job can visit subset of the machines. The objective is to find the sequence of jobs which minimizes the total delay of all jobs.

Short Answer Questions

Essay Questions

1. Explain the classes of NP-hard and NP-complete and specify its properties.
2. Explain in detail non-deterministic algorithm with proper example.
3. Distinguish between deterministic and non-deterministic algorithms.
4. Explain about Clique Decision Problem with an example.
5. Explain about TSP with an example.

Short Answer Questions

1. What is NP-Hard?
2. What is Np-Complete?
3. List the difference between deterministic and non-deterministic algorithm.
4. Define decision problem and optimization problem.
5. What is meant by clique decision problem?
6. Show the relation between P and NP problem.
7. Define satisfiability problem.
8. Define reducibility.