**WOLKITE UNIVERSITY**

**COLLEGE OF COMPUTING AND INFORMATICS**

**DEPARTEMENT OF COMPUTRE SCINCE**

**COURSE TITEL: Automata and Complexity Theory**

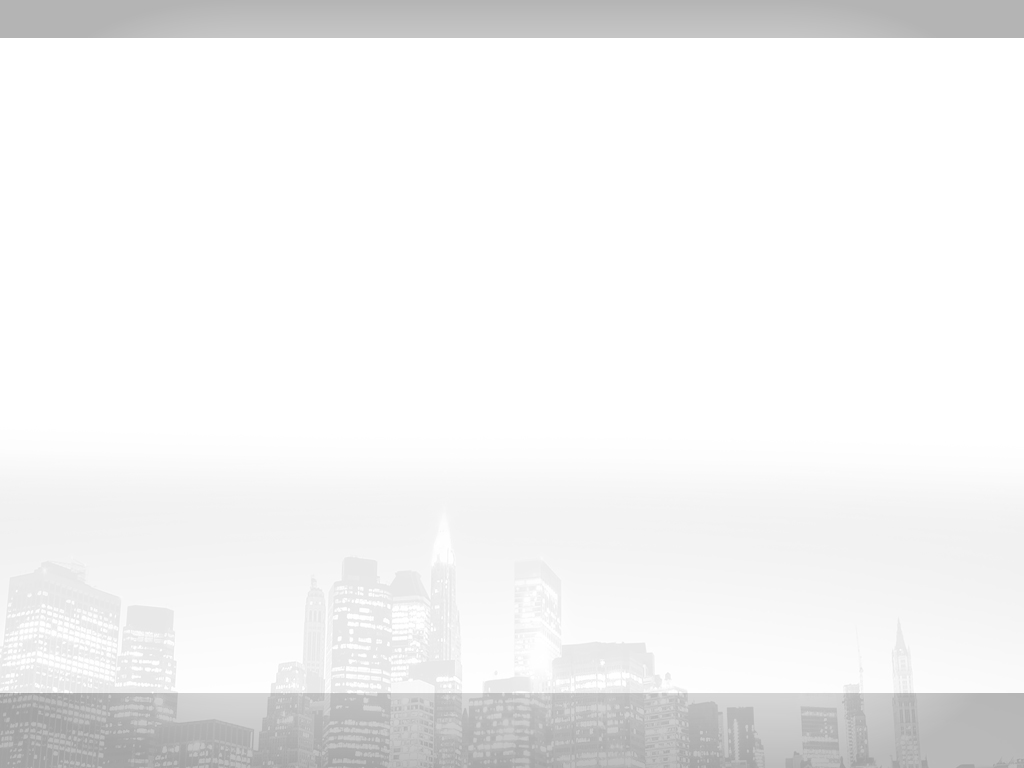
**COURSE CODE:CoSc3024**

**Name**:Ahlam Nuredin

**ID.No**:NSR/0174/13

Submitted To : Mr. Kolana.K

Submission Date : 25/05/2015 EC



**I.Some well-known problems that are NP-complete are:**

**1.N-puzzle**

N-puzzle is a type of puzzle where a set of tiles are arranged in a grid of size NxN, with one empty tile. The goal of the puzzle is to rearrange the tiles by sliding them around to reach a certain goal state. This is usually done by swapping the empty tile with a tile adjacent to it. N-puzzle is considered to be an NP-hard problem, meaning that it is very difficult to solve. There are many algorithms that can be used to solve the puzzle, such as A\* search and IDA\*.

**2.Knapsack**

The Knapsack problem is an instance of a Combination Optimization problem. One general approach to crack difficult problems is to identify the most restrictive constraint. For this, we must ignore the others and solve a knapsack problem, and finally, we must somehow fit the solution to satisfy the constraints that are ignored.

Based on the nature of the items, Knapsack problems are classified into two categories

* Fractional Knapsack
* Knapsack

Fractional Knapsack

Fractional Knapsack is an optimization problem that can be used to maximize the amount of value that can be obtained from a given set of items. In this problem, each item has a weight and a value, and the goal is to fill a knapsack of given capacity with items such that the total value of the items inside the knapsack is maximized. The difference between this problem and the 0-1 knapsack problem is that in this problem, each item can be divided and only a fraction of it can be taken, whereas in the 0-1 knapsack problem, either the entire item can be taken or none at all.

**3**. **Traveling Salesman Problem**

The traveling Salesman Problem (TSP) is a combination problem that deals with finding the shortest and most efficient route to follow for reaching a list of specific destinations.

It is a common algorithmic problem in the field of delivery operations that might hamper the multiple delivery process and result in financial loss. TSP turns out when you have multiple routes available but choosing a minimum cost path is really hard for you or a traveling person.

## Common Challenges of Traveling Salesman Problem (TSP)

* Firstly, every day, salespeople have to carry out a number of deliveries in a very limited time, so there are a lot of time constraints. To overcome this, you need to plan your routes in a way that you make the most out of them.
* Secondly, there are chances of last-minute changes. Sometimes you get extra and urgent visits to make, while sometimes, some visits are postponed or canceled due to the customer’s unavailability.
* Lastly, a math problem, a combination optimization problem, arises. A combination optimization problem is a problem that is mathematically complex to solve as you have to deal with many variables.

**4.** **Sub graph Isomorphism Problem**

 The sub graph isomorphism problem is a computational task in which two [graphs](https://en.wikipedia.org/wiki/Undirected_graph" \o "Undirected graph) G and H are given as input, and one must determine whether G contains a [subgraph](https://en.wikipedia.org/wiki/Glossary_of_graph_theory" \l "subgraph" \o "Glossary of graph theory) that is [isomorphic](https://en.wikipedia.org/wiki/Graph_isomorphism" \o "Graph isomorphism) to H. Sub graph isomorphism is a generalization of both the [maximum clique problem](https://en.wikipedia.org/wiki/Clique_problem" \o "Clique problem) and the problem of testing whether a graph contains a [Hamiltonian cycle](https://en.wikipedia.org/wiki/Hamiltonian_cycle" \o "Hamiltonian cycle), and is therefore [NP-complete](https://en.wikipedia.org/wiki/NP-complete" \o "NP-complete). However certain other cases of sub graph isomorphism may be solved in polynomial time.

1. **Subset Sum Problem**

The subset sum problem (SSP) is a [decision problem](https://en.wikipedia.org/wiki/Decision_problem" \o "Decision problem) in [computer science](https://en.wikipedia.org/wiki/Computer_science" \o "Computer science). In its most general formulation, there is a [multiset](https://en.wikipedia.org/wiki/Multiset" \o "Multiset) � of integers and a target-sum �, and the question is to decide whether any subset of the integers sum to precisely �.The problem is known to be NP-hard. Moreover, some restricted variants of it are [NP-complete](https://en.wikipedia.org/wiki/NP-completeness" \o "NP-completeness) too

* The variant in which all inputs are positive.
* The variant in which inputs may be positive or negative, and. For example, given the set {−7,−3,−2,9000,5,8}, the answer is yes because the subset {−3,−2,5} sums to zero.
* The variant in which all inputs are positive, and the target sum is exactly half the sum of all inputs, i.e., �=12(�1+⋯+��) . This special case of SSP is known as the [partition problem](https://en.wikipedia.org/wiki/Partition_problem" \o "Partition problem).

SSP is a special case of the [knapsack problem](https://en.wikipedia.org/wiki/Knapsack_problem" \o "Knapsack problem) and of the [multiple subset sum](https://en.wikipedia.org/wiki/Multiple_subset_sum" \o "Multiple subset sum) problem.

1. **Clique Problem**

The clique problem is the computational problem of finding [cliques](https://en.wikipedia.org/wiki/Clique_(graph_theory)" \o "Clique (graph theory)) (subsets of vertices, all [adjacent](https://en.wikipedia.org/wiki/Adjacent_(graph_theory)" \o "Adjacent (graph theory)) to each other, also called [complete](https://en.wikipedia.org/wiki/Complete_graph" \o "Complete graph) [subgraphs](https://en.wikipedia.org/wiki/Glossary_of_graph_theory" \l "Subgraphs" \o "Glossary of graph theory)) in a [graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)" \o "Graph (discrete mathematics)). It has several different formulations depending on which cliques, and what information about the cliques, should be found. Common formulations of the clique problem include finding a [maximum clique](https://en.wikipedia.org/wiki/Maximum_clique" \o "Maximum clique) (a clique with the largest possible number of vertices), finding a maximum weight clique in a weighted graph, listing all [maximal cliques](https://en.wikipedia.org/wiki/Maximal_clique" \o "Maximal clique) (cliques that cannot be enlarged), and solving the [decision problem](https://en.wikipedia.org/wiki/Decision_problem" \o "Decision problem) of testing whether a graph contains a clique larger than a given size.

**7**. **Vertex cover Problem**

Vertex cover (sometimes node cover) of a [graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)" \o "Graph (discrete mathematics)) is a set of [vertices](https://en.wikipedia.org/wiki/Vertex_(graph_theory)" \o "Vertex (graph theory)) that includes at least one endpoint of every [edge](https://en.wikipedia.org/wiki/Edge_(graph_theory)" \o "Edge (graph theory)) of the graph.

The problem of finding a minimum vertex cover is a classical [optimization problem](https://en.wikipedia.org/wiki/Optimization_problem" \o "Optimization problem). It is [NP-hard](https://en.wikipedia.org/wiki/NP-hard" \o "NP-hard), so it cannot be solved by a [polynomial-time](https://en.wikipedia.org/wiki/Polynomial-time" \o "Polynomial-time) algorithm if [P ≠ NP](https://en.wikipedia.org/wiki/P_%E2%89%A0_NP" \o "P ≠ NP). Moreover, it is [hard to approximate](https://en.wikipedia.org/wiki/Hard_to_approximate" \o "Hard to approximate) – it cannot be approximated up to a factor smaller

**8**.**Independent set Problem**

An independent set is a set of nodes in a binary tree, no two of which are adjacent, i.e., there is no edge connecting any two. The size of an independent set is the total number of nodes it contains. The maximum independent set problem is**finding an independent set of the largest possible size for a given binary tree**.

The idea is to traverse the binary tree, and for each node in the binary tree, there are two possible cases:

1. Exclude the current node from the maximum independent set and process its children.
2. Include the current node in the maximum independent set and process its grandchildren.
3. **Graph Coloring Problem**

Vertex coloring is the most common graph coloring problem. The problem is, given**m colors, find a way of coloring the vertices of a graph such that no two adjacent vertices are colored using same color.** The other graph coloring problems like Edge Coloring(no vertex is incident to two edges of same color) and Facing Coloring(geographical map coloring) can be transformed into vertex coloring.

1. **Minesweeper**

Is a classic puzzle game where the player attempts to identify the locations of all the mines on a grid without uncovering any of them. The game is played on a rectangular grid with some squares labeled as mines. The player must use logic to determine which squares are safe to uncover and which are mines. The game ends when all the safe squares are uncovered or when a mine is uncovered. The player wins when all the mines are identified.

**II**. **Space Complexity: PSPACE**

**PSPACE**

In computational complexity theory, a decision problem is P SPACE-complete if it can be solved using an amount of memory that is polynomial in the input length (polynomial space) and if every other problem that can be solved in polynomial space can be transformed to it in polynomial time.

**Space Complexity**

The space Complexity of an algorithm is the total space taken by the algorithm with respect to the input size. Space complexity includes both Auxiliary space and space used by input.