Formulate travelling salesman problem and prove its NP-completeness

Formulation of the Travelling Salesman Problem (TSP)

Problem Definition: The Travelling Salesman Problem (TSP) can be formulated as follows:

- **Input**: A set of n cities, a distance function d(i,j) that gives the distance between city i and city j for all pairs of cities (i,j), and an integer k.
- **Output**: Determine whether there exists a tour (a Hamiltonian cycle) that visits each city exactly once, returns to the starting city, and has a total distance less than or equal to k.

In the decision version of TSP, we are asked if there is a tour of length at most k.

Proving NP-Completeness of TSP

To prove that the Travelling Salesman Problem (TSP) is NP-complete, we need to show two things:

1. **TSP is in NP**: Given a candidate tour, we can verify in polynomial time whether it is a valid tour and whether its total distance is less than or equal to K.
2. **TSP is NP-hard**: We need to show that any problem in NP can be reduced to TSP in polynomial time. This can be done by reducing a known NP-complete problem to TSP.

1. TSP is in NP

Given a sequence of cities (a tour), we can:

- Check if each city is visited exactly once.
- Calculate the total distance of the tour by summing up the distances d(i,j) for each consecutive pair of cities in the tour.
- Verify if this total distance is less than or equal to K.

All these steps can be performed in polynomial time, so TSP is in NP.

2. TSP is NP-hard

To show that TSP is NP-hard, we reduce the Hamiltonian Cycle (HC) problem to TSP. The Hamiltonian Cycle problem is a known NP-complete problem.

Hamiltonian Cycle Problem:

- **Input**: A graph G=(V,E)
- **Output**: Determine whether there exists a cycle in G that visits each vertex exactly once and returns to the starting vertex.

Reduction from HC to TSP:

1. **Constructing the TSP Instance**:

    - Given a graph G=(V,E) with n vertices, construct a complete graph G′ with the same set of vertices.
    - Define the distance function d′ for G′ as follows:
        - If (u,v) is an edge in G, set d′(u,v)=1.
        - If (u,v) is not an edge in G, set d′(u,v)=2.
2. **Setting the Distance Bound**:
    - Set k=n. This is because if there is a Hamiltonian Cycle in G, the total distance of the corresponding tour in G′ will be exactly n.
3. **Correctness of the Reduction**:
    - If G has a Hamiltonian Cycle, then this cycle corresponds to a tour in G′ with a total distance of exactly n (since all edges in the cycle have a distance of 1).
    - If G′ has a tour of total distance ≤n, then this tour must use only edges of distance 1 (since using any edge of distance 2 would result in a total distance greater than n). This means that the tour corresponds to a Hamiltonian Cycle in G.

Since the reduction from HC to TSP can be done in polynomial time and HC is NP-complete, it follows that TSP is NP-hard.

Suppose X is an NP-complete problem. Then X is solvable in polynomial time if and only if P = NP."

1. **P (Polynomial Time)**: This class contains decision problems that can be solved by a deterministic Turing machine in polynomial time. In simpler terms, problems for which a solution can be found quickly (in polynomial time).
2. **NP (Nondeterministic Polynomial Time)**: This class contains decision problems for which a given solution can be verified by a deterministic Turing machine in polynomial time. These are problems where we can check a given solution quickly, even if finding the solution might be difficult.
3. **NP-Complete Problems**: These are the hardest problems in NP. A problem X is NP-complete if:
    - X is in NP.
    - Every problem in NP can be reduced to X in polynomial time (X is as hard as any problem in NP).

Implications of the Statement

To understand the statement, let's consider both directions of the "if and only if" condition:

If P=NP:

- If P=NP, it means that every problem in NP can be solved in polynomial time.
- Since X is an NP-complete problem and X is in NP, if P=NP, then X can be solved in polynomial time.
- Hence, if P=NP, then X is solvable in polynomial time.

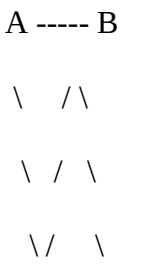
If X is solvable in polynomial time:

- Suppose X, an NP-complete problem, is solvable in polynomial time. This means there exists a polynomial-time algorithm to solve X.
- Since X is NP-complete, any problem in NP can be reduced to X in polynomial time. Let's denote any problem in NP by Y.
- Because X is solvable in polynomial time, and Y can be reduced to X in polynomial time, Y can also be solved in polynomial time.
- Therefore, if an NP-complete problem X is solvable in polynomial time, then every problem in NP can be solved in polynomial time, implying P=NP.

Vertex cover and Independent set problem with an example and proof to determine that both of the problems are NP-Complete.

Vertex Cover Problem

Definition: Given a graph G=(V,E) and an integer k, the Vertex Cover problem asks whether there is a subset of vertices $V' \subseteq V$ such that no such that every edge in E is incident to at least one vertex in V' and the size of V' is at most K



- Let k=2.
- A possible vertex cover is {B,C}, as every edge in the graph is incident to either B or C.

Independent Set Problem

Definition: Given a graph G=(V,E) and an integer k, the Independent Set problem asks whether there is a subset of vertices $V' \subseteq V$ such that no two vertices in V′V'V′ are adjacent, and the size of V′ is at least k.

Example:

- Consider the same graph G:

- Let k=2.
- A possible independent set is {A,D}\{A, D\}{A,D}, as no edge connects A and D.

Proving NP-Completeness

1. Vertex Cover is NP-Complete

Step 1: Vertex Cover is in NP

- Given a subset of vertices V′ and the graph G, we can verify in polynomial time whether every edge in G is incident to at least one vertex in V′, and whether |V′|≤k|V'| \leq k|V′|≤k.

Step 2: Reduction from 3-SAT to Vertex Cover

- We reduce an instance of 3-SAT to an instance of Vertex Cover.

Reduction:

- Given a 3-SAT formula with mmm clauses and nnn variables.
- Construct a graph GGG as follows:
  - For each variable xix_ixi, create two vertices xix_ixi and ¬xi\neg x_i¬xi.
  - Connect xix_ixi and ¬xi\neg x_i¬xi with an edge (forcing the cover to include one of them, not both).
  - For each clause (li1∨li2∨li3)(l_{i1} \vee l_{i2} \vee l_{i3})(li1∨li2∨li3), create a triangle connecting li1l_{i1}li1, li2l_{i2}li2, and li3l_{i3}li3.
- Set k=n+2mk = n + 2mk=n+2m.

Verification:

- If the 3-SAT formula is satisfiable, we can select one literal from each clause and the corresponding vertices in the graph such that each clause triangle is covered by one vertex, and the variable constraints are met by selecting exactly one vertex from each variable pair.
- Conversely, if we have a vertex cover of size kkk, it must include nnn vertices covering all variable edges and mmm vertices covering the clause triangles.

Thus, Vertex Cover is NP-Complete.

2. Independent Set is NP-Complete

Step 1: Independent Set is in NP

- Given a subset of vertices V′V'V′ and the graph GGG, we can verify in polynomial time whether no two vertices in V′V'V′ are adjacent, and whether |V′|≥k|V'| \geq k|V′|≥k.

Step 2: Reduction from Vertex Cover to Independent Set

- Show that Vertex Cover can be reduced to Independent Set.

Reduction:

- Given a graph G=(V,E)G = (V, E)G=(V,E) and an integer kkk for the Vertex Cover problem.
- Construct the same graph GGG.
- Set k′=|V|−kk' = |V| - kk′=|V|−k for the Independent Set problem.

Verification:

- If V’V'V’ is a vertex cover of size kkk, then V−V’V - V'V−V’ is an independent set of size |V|−k|V| - k|V|−k.
- Conversely, if V’’V''V’’ is an independent set of size k’k'k’, then V−V’’V - V''V−V’’ is a vertex cover of size |V|−k’=k|V| - k' = k|V|−k’=k.

Thus, Independent Set is NP-Complete.

-