A20551908
Vadlamudi Manogha

1) Assume that P=NP.
Take any language except $A=\phi$ and $A=\Sigma^*$ where $A \in P$.
we want to show that A is NP-complete
To do this, we need to show that A is in NP and that every language B in NP can be reduced to A in polynomial time.
Since $A \in P$, it means that there exist a polynomial-time algorithm that decides whether a given input belongs to A or not.
This algorithm can be used as certificate for A, so A is in NP.
Now, let's consider any language B in NP.
Since P=NP, it means that there exists a polynomial-time algorithm that decides whether a given input belongs to B or not.
We can use this algorithm to reduce B to A in polynomial time.
Specifically, we can modify the algorithm for B to first check if the input belongs to A, and if it does, output "yes". otherwise, run the original algorithm for B.
This reduction can be done in polynomial time because both the algorithm for A & the algorithm for B run in polynomial time.
Therefore, A is NP-complete, as every language B in NP can be reduced to polynomial time.
Thus, if P=NP, then every language $A \in P$, except $A=\phi$ and $A=\Sigma^*$, is NP-complete.

2) Let's define the language for the KNAPSACK problem,
Language: KNAPSACK Input: A set of items $\{1,2,\ldots,n\}$ each with an integer size S_i and an integer profit P_i, and integers k and B.
Output: YES if there exists a subset S of items such that the sum of their sizes is less than or equal to B and the sum of their profits is greater than or equal to k.
NO otherwise.

To prove that KNAPSACK is NP-complete, we need to show that it is in NP and that it is NP-hard.

To show that KNAPSACK is in NP, we need to show that given a

potential solution (subset S), we can verify it in polynomial time.
Given a potential solution S, we can calculate the sum of sizes of the items in S and check if it is less than or equal to B. This can be done in O(n) time, where n is the number of items.
We can also calculate the sum of the profits of the items in S and check if it is less than or equal to K. This can be done in O(n) time.
Therefore, we can verify a potential solution in polynomial time, and KNAPSACK is in NP.
To show that KNAPSACK is NP-hard, we need to reduce a known NP-complete problem to KNAPSACK.
Let's choose the subset sum problem as our known NP-complete problem. The subset sum problem is defined as follows:
language: subset sum
Input: A set of integers {a1, a2, ..., an} and an integer B.
Output: YES if there exists a subset S of the integers such that the sum of the integers in S is equal to B. No, otherwise.
We will now see the reduction from subset sum to KNAPSACK.
Given an instance of the subset sum problem, we can create an instance of the KNAPSACK problem as follows:- let K = B (the target sum in the subset sum problem).
- For each integer a_i in the subset sum problem, create an item with size $S_i = a_i$ and profit $P_i = a_i$.
Now, we need to show that the reduction is correct, i.e, if there exists a subset S of the integers in the subset sum problem that sums to B, then there exists a subset S' of item in the KNAPSACK problem such that the sum of their sizes is less than or equal to B and the sum of their profits is greater than or equal to K.
If there exists a subset of S of the integers in the subset sum problem that sums to B, then the sum of the sizes of the corresponding items in the KNAPSACK problem is also equal to B. This is because we set the size of each items to be equal to the corresponding integer in the subset sum problem.
The sum of the profits of the corresponding items in the KNAPSACK problem is also equal to the sum of the integers in the subset sum problem. This is because we set the profit of each item to be equal to the corresponding integer in the subset sum problem.

Therefore, if there exists a subset S of the integers in the subset sum problem that sums to B, then there exists a subset S' of items in the KNAPSACK problem such that the sum of their sizes is less than or equal to B and the sum of profits is greater than or equal to k.

Conversely, if there exists a subset S' of items in the KNAPSACK problem such that the sum of their sizes is less than or equal to B and the sum of their profits is greater than or equal to k, then the sum of the sizes of the corresponding items in the subset sum problem is also equal to B. This is because we set the profit of each item to be equal to the corresponding integer in the subset sum problem.

The sum of the profits of the corresponding items in the KNAPSACK problem is also equal to the sum of the integers in the subset sum problem. This is because we set the profit of each item to be equal to the corresponding integer in the subset sum problem.

Therefore, if there exists a subset S' of items in the KNAPSACK problem such that the sum of their sizes is less than or equal to B and the sum of their profits is greater than or equal to k, then there exist a subset S of the integers in the subset sum problem that sums to B.

Thus, the reduction is correct.

Since, the subset sum problem is known to be NP-complete, and we have shown a reduction from subset sum to KNAPSACK, we can conclude that KNAPSACK is also NP-complete.

Therefore, the language KNAPSACK is NP-complete.

3) The SHORTEST SIMPLE S-t PATH problem involves finding a simple path from vertex s to vertex t in a directed graph, such that the total weight of the path does not exceed a given positive integer k. A path is considered simple if it does not revisit any vertices, and the weight of the path is the sum of the weights of its edges.

Let's formulate this problem as a language L_s. We want to determine whether there exists a simple path from s to t with a total weight at most k. The input for L_s consists of a directed graph $G = (V, E, W)$, where:

V represents the set of vertices
E represents the set of edges
W(e) is the weight associated with each edge e∈E.
Now, let's prove that L_s is NP-complete. We'll achieve this by demonstrating a reduction from the HAMPATH problem:
Given an undirected graph $G=(V,E)$ and two vertices s and t, determine whether there exists a Hamiltonian path (a path that visits each vertex exactly once) from s to t.
The HAMPATH problem is known to be NP-complete.

Reduction:-

We'll show that if we can solve L_s efficiently, we can also solve HAMPATH efficiently.
Given an instance of HAMPATH with graph $G=(V,E)$ and vertices s and t, we construct an instance of L_s as follows:
create a new directed graph $G'=(V',E',w')$:
For each edge $(i,j) \in E$, add two directed edges: (i,j) and (j,i) to E'.
Assign a weight of -1 to all edges in E'.
Set $s'=s$ and $t'=t$.
choose $k'=-(n-1)$, where n is the number of vertices in G.

Explanation:-

If G has a Hamiltonian path from s to t, then there exists a simple path in G' from s' to t' with total weight k' (since the sum of weights is $-(n-1)$).
conversely, if G' has a simple path from s' to t' with total weight k', then this path corresponds to a Hamiltonian path in G.
We've shown a polynomial-time reduction from HAMPATH to L_s.
Since HAMPATH is NP-complete, L_s must also be NP-complete.
Therefore, we've established that the SHORTEST SIMPLE s-t PATH problem (language L_s) is NP-complete, using a reduction from HAMPATH.