Graph algorithms exercises

Problem 1 (MCQs)

Check all statements that are true:

1. If we can solve Subset Sum in polynomial time, then `P=NP`. True. Subset Sum is NP-hard.
2. Every problem in the class `P` can be reduced to `Vertex-Cover`. True, since P is a subclass of NP.

Consider the `Longest Path Problem`:
Input: a graph $G = (V, E)$ and an integer $g > 0$.
Output: a path of length greater or equal that g.

This problem can be proved to be `NP-hard` by generalization from which of the following problems?

1. `Independent Set`.
2. `Clique`.
3. `Vertex Cover`.
4. `Rudrata Path`. Take $g = |V|$.

Problem 2 Problem 8.12 from [DPV] ($K-$spanning tree)

Solution

Let's denote the problem $K-$ST.

$K-$ST is in the class `NP`.

Given an instance $G = (V, E); k > 0$ and a candidate solution T, subgraph of G we proceed as follows:

1. Loop through the vertices to get their degree in T and check if they all have degree less than or equal to k. This is a $O(n)$ operation per vertex, resulting in a $O(n^2)$ overall runtime.
2. Run `Explore` on T from any vertex and verify if all vertices are marked *visited* (alternatively, you can run `DFS` to get the connected components). This will check that T is connected. The runtime of `Explore` is $O(n + m)$.
3. Run `DFS` on T and check if any edge is classified as *back edge*. This will tell us if the subgraph T is cycle-free. The runtime of `DFS` is $O(n + m)$.

After the three steps above, we will know if T is connected, spans all the vertices and is cycle free. Also we will check the condition on the degree of the vertices. We conclude this procedure will validate T as a solution. The overall runtime is $O(n^2)$ since the linear terms are absorbed by the first step runtime.

$K-$ST is in the class `NP-hard`.

We reduce from `Rudrata Path`.

Let $G = (V, E)$ be an input of `Rudrata Path`. To build an input of $K-$ST we pass the same graph G and set $k = 2$. Since we are using the same graph, this transformation is clearly polytime.

We show now that a solution of $K-$ST exists if and only if a solution of `Rudrata Path` also exists. The key observation is that a tree such that all vertices have degree at most two is a path. Hence, if T is a solution to $2-$ST it is a path that visits all vertices, and thus it is a solution to `Rudrata Path`. The opposite direction follows from the same reasoning. Note that recovering the solution of `Rudrata Path` can be done in polytime, $O(n+m)$, as it is the same solution of $2-$ST.

This concludes the reduction.

Problem 3 Problem 8.14 from [DPV] (Clique+IS)

Solution

We denote the problem of finding a clique and an independent set of size k in G as the Clique-IS problem. This problem is in NP since given a solution (S_1, S_2) and an input graph $G = (V, E)$ to this problem, in $O(n^2)$ time, we can check whether (S_1, S_2) is a solution to Clique-IS by checking whether S_1 is a clique by checking for all pairs $x, y \in S_1$ that $(x, y) \in E$, and that S_2 is an independent set by checking for all pairs $x, y \in S_2$ that $(x, y) \notin E$.

Now, we will reduce a known NP-complete problem, `Clique` to Clique-IS.

Recall that in the `Clique` problem, one is given a graph G and a number k, and the answer is whether the graph has a clique of size $\geq k$ or not. Note that if a graph has a clique of size $\geq k$ then it has a clique of size $= k$, so it suffices to determine if there is a clique of size $= k$ in the input graph G.

Consider an input to the `Clique` problem with a graph G and a parameter k. We will define a graph G' to run the Clique-IS problem on. To create G', we add a set I of k new vertices to G, there are no new edges added. This forms the new graph G' in time $O(n)$ since we only ned to add $k \leq n$ vertices. Note that G' always contains an independent set I of size $= k$. Moreover, this set I is not included in any cliques in G' since there are no edges from I. Hence, G' has a clique and an independent set of size $= k$ if and only if G has a clique of size $= k$. Therefore, if we can solve the Clique-IS problem on G' for parameter k then we can solve the Clique problem on G for parameter k. Furthermore, to recover the solution of `Clique` from a solution to Clique-IS we simply need to drop the independent set since the clique will be entirely made of vertices and edges of G. This operation can be done in time $O(n + m)$.

This completes the reduction and proves that the Clique-IS problem is NP-Complete.