CS 512: Design and Analysis of Algorithms

Assignment #4

Name: Kunal Wanikar

Roll No.: 204101070

Question 1)

Given: A graph G and an integer k

To prove: Problem X: Does G have a cycle, with no repeated nodes, of length at least k?
1) The given problem is *NP*
2) We can reduce a known *NP complete* problem into the given problem in polynomial time.

Proof:

1) In the first part of the proof we need to show that the given problem is *NP*. This can be done by showing that there exists a **polynomial time algorithm** in which we can **verify** that our problem is being solved. This will act as a **certifier** that the given problem is *NP*.
Input to the certifier will be a cyclic subgraph of graph *G* which has a length of at least k and no vertex is being repeated. Let the vertices in the cyclic subgraph be $V' = \{v_1, v_2, \ldots, v_n\}$.

Algorithm: Let us take any vertex $v \in V'$ as a starting vertex of the cycle and this is the point where the cycle will end. Let *num_of_vertices* store the total number of vertices being traversed from the cycle. Let us take *index* variable which will store the current vertex being traversed and a boolean array named *flag* which stores whether the vertex is traversed or not.

1. Select any vertex $\boldsymbol{v \in V'}$ as the first vertex of the cyclic subgraph.
2. Initializing ***num_of_vertices* = *1*, *index* = next vertex of v and *flag[v]* = 1** and all other ***flag[i]* = 0.**
3. While (*index != v*)
   a. If (***flag[index]* = = 1**) then that vertex is already being **traversed** and we get that this vertex is being **repeated** hence we **break** and come out of while loop.
   b. Else we **add 1** to *num_of_vertices* (i.e. *num_of_vertices = num_of_vertices+1*) and set ***flag[index]* = *1*.**
4. If (***num_of_vertices* < *k***) then return *false*; else return *true*.

The time complexity of the above algorithm is ***O(V)*** as in the worst case the cyclic subgraph is the given cycle graph with **longest possible cycle of all vertices**. Now as the **certifier** is taking polynomial time hence, we can safely claim that our problem is *NP*.

2) In the second part of the proof, we need to show that some known *NP complete* problem is poly-time reducible to the given problem (i.e. does G have a cycle, with no repeated nodes, of length at least k)

Let us take the closest *NP complete* problem known to us which can be reduced in to this problem as ***Hamiltonian cycle*** problem. This problem is a decision problem which outputs yes if it finds a cycle starting from any vertex without repeating any vertex.

Let us take a *Hamiltonian cycle* problem with number of vertices equal to *n* and using without loss of generality and the certifier proof algorithm we can say that the Hamiltonian cycle problem with $n = k$ (where n is the number of vertices in the Hamiltonian cycle graph and k is the number of vertices in the graph of given problem X) reduces to the given problem X in *polynomial time*. Now since we know that Hamiltonian cycle problem is a *NP complete* problem and **Hamiltonian Cycle** $\leq_P$ **X** (with the help of the certifier), Hamiltonian cycle problem is polynomial time reducible to X, therefore we can say that **X** is also a *NP complete* problem.

Conclusion: Hence by taking the advantage of certifier we proved that **Hamiltonian cycle** problem can be reduced to the given problem **X** in **O(V)** polynomial time and as we know that Hamiltonian cycle problem is an NP complete problem hence we can say that the given problem is also **NP complete**

Question 2)

Given: Family of sets $\{S_1, S_2, \dots, S_n\}$ and an integer *b*.

To prove: Problem X: Is there a set *H* with *b* or fewer elements such that *H* intersects all the sets in the family?

1) The given problem is *NP*

2) We can reduce a known *NP complete* problem into the given problem X in polynomial time.

Solution:

1) In the first part of the proof we need to show that the given problem is *NP*. This can be done by showing that there exists a **polynomial time algorithm** in which we can **verify** that our problem is being solved. This will act as a **certifier** that the given problem is *NP*.
Input to the certifier will be a set *H* with less than or equal to b elements.

Algorithm: Let us take the set $\boldsymbol{S = \{S_1, S_2, \dots, S_n\}}$ with **n** sets. We will take a variable *count* which will store the count of sets which have a common element in between them. We will take another variable *num_of_ele* which will store the number of elements in the intersection of two sets.

1. Initialize *count=0* and *i=1*
2. While (i<=n)
    a. Select a set S_i

   b. ***num_of_ele = (Si intersection H)***
   c. i = i +1
   d. If (***num_of_ele >=1***) then ***count = count + 1;***
3. If (***count != n***) return false
   Else return true

The time complexity analysis for the above algorithm will be as follows: Time for finding number of elements common to set S_i and set *H* will take at most *O(nb)* time and this will happen n times as total number of S_i*'s* are *n*. Hence the overall time complexity will be ***O(bn²).*** Now as the **certifier** is taking polynomial time hence, we can safely claim that our problem is *NP*.

2) In the second part of the proof, we need to show that some known *NP complete* problem is poly-time reducible to the given problem (i.e. Is there a set *H* with *b* or fewer elements such that *H* intersects all the sets in the family)

Let us take the closest *NP complete* problem known to us which can be reduced in to this problem as ***Vertex cover*** problem. This problem is an optimization problem which outputs the minimum vertex cover for an undirected graph.

Let us take a vertex cover problem with undirected graph $G = (V, E)$ where *V* be the set of all vertices of the graph and *S* be the set of *{u, v}* pairs where each pair is the edge between vertices *u* and *v*. Let *G (V, E), b'* be an instance of VERTEX COVER. In total we have |E| sets, and we set *b = b'*. Now the claim is as follows G has a vertex cover of size at *b'* if and only if $\{S_1, S_2, \ldots, S_n\}$ has a hitting set H of size at most b = b'.

- **Proving "➔" :** Let VC be the vertex cover of graph G having length <=*b* this implies that for every edge *{u, v}* either u or v belongs to VC. So, if we take VC set as set H and then intersect it with every set Si ∈ H we will either get u as a common or v as common. Hence VC set is a solution to a given problem X.
- **Proving "🡐" :** Let H be the set which intersects with every set *Si ∈ S*. Now, since H intersects with every *Si ∈ H* then at least one of the endpoints of every edge *{u, v}* must belong to the solution. Hence H spans at least one end of each *{u, v}* edge hence H is vertex cover.

Now, as we have proved the if and only if part, we can say that the Vertex cover problem reduces to given problem X in ***O(bn²)*** polynomial time. Now since we know that Vertex cover problem is a *NP complete* problem and **Vertex cover $\leq_P$ X** (with the help of the certifier), Vertex cover problem is polynomial time reducible to X, therefore we can say that **X** is also a *NP complete* problem.

Conclusion: Hence by taking the advantage of certifier we proved that **Vertex Cover** problem can be reduced to the given problem **X** in **O(bn²)** polynomial time and as we know that Vertex cover problem is an NP complete problem hence we can say that the given problem is also **NP complete.**