Name: Mansi Jainendra Tandel

SCU id: W1606463

1.True or false

1. True
2. True
3. False
4. False
5. True
6. False
7. False
8. False
9. False
10. True
11. False
12. True
13. False

2. (a) Prove that all leaf nodes of the tree belong to a solution.

Using greedy choice property, we can get the globally optimal solution by making locally optimal choices. So here, we will make a choice that looks best in the current problem. So we must prove that greedy choice at every step yields a globally optimal solution. The leaf nodes are derived from the root nodes so it contains the solution of root node.

Greedy choice property is given below:

(a) Greedy-choice property: a globally optimal solution can be arrived at by making a locally optimal choice.

(b) Optimal substructure states that a problem exhibits optimal substructure if an optimal solution to the problem contains within its optimal solutions to subproblems.

However, if this undirected graph is a tree, then finding maximum size of such a set is polynomial-time solvable. For example, the following tree (left) has a maximum independent set shown on the right.

So according to the greedy choice property, We can only get a solution if we have the solution of its subproblems which are leaf nodes. So for the undirected graph shown in the example, the leaf nodes must be locally optimal solution for their parent nodes, which will give us the globally optimal solution. So, we can say that all the leaf nodes of the tree belong to a solution.

2. (b).

Greedy(G):

S = {}

While G is not empty:

Let v be a node with minimum degree in G

S = union(S, {v})

```
            remove v and its neighbors from G
 return S
```

3.

Psuedo –Code(Top-down approach)

```
 int func(int n)
 {
            for i -> n+1 {
            lookup[i] = -1
           }
            if (lookup[n] == -1)
            {
                      if (n <= 1)
                               lookup[n] = 5
                                else
                                          lookup[n] = 2 * func(n-1) * func(n-2)
           }
           return lookup[n];
            }
```

~~The~~

4) The vertex-cover problem is to find a vertex cover of minimum size a given graph. Restating this optimization problem as a decision problem, we wish to determine whether a graph has a vertex cover of a given size k. As a language we define,

VERTEX_COVER = {<G, k> : graph G has a vertex cover of size k}.

The following shows that CLIQUE is not harder to solve than VERTEX COVER

Proof:

(1) We first show that VERTEX_COVER ∈ NP. Suppose we ~~have~~ are given a graph $G = (V, E)$ and an integer k. The certificate we choose is the vertex cover $V' \subseteq V$ itself. The verification algorithm affirms that $|V'| = k$, and then it checks, for each edge $(u, v) \in E$, that $u \in V'$ or

$v \in V'$. We can easily verify the certificate in polynomial time.

(2)
We prove that the vertex cover is NP-Hard by showing that CLIQUE $\leq_p$ VERTEX COVER.
This reduction relies on the notion of the "complement" of a graph. Given an undirected graph $G=(V, E)$, we define the complement of G as $\bar{G}=(V, \bar{E})$, whose $\bar{E}=\{(u,v): u,v \in V, u \neq v,$ and $(u,v) \notin E\}$.
In other words, $\bar{G}$ is the graph ~~and that its complement and~~ containing exactly those edges that are not in G.

The reduction algorithm takes an input $\langle G, k \rangle$ of the clique problem.

It will compute $\bar{G}$, which is easily done in polynomial time.

The output of the reduction algorithm is the instance $\langle \bar{G}, |V|-k \rangle$ of the vertex cover problem.

→ To complete the proof, we show that this transformation is indeed a reduction, which is: the graph G has a clique of size k if and only if the graph $\bar{G}$ has a vertex cover of size $|V|-k$.

→ Suppose that G has a clique $V' \subseteq V$ with $|V'| = k$. We claim that $V - V'$ is a vertex cover in $\bar{G}$.
→ Let (u, v) be any edge in $\bar{E}$. Then, $(u, v) \notin E$, which implies that at least one of u or v does not belong to V', since every pair of vertices in V' is connected by an edge of E.
→ Equally, at least one of u or v is in $V - V'$, which means that edge (u, v) is covered by $V - V'$. Since (u, v) was chosen from $\bar{E}$ arbitrarily, every edge of $\bar{E}$ is covered by a vertex in $V - V'$.

Hence, the set $V - V'$, which has a size $|V| - k$, forms a vertex cover for $\bar{G}$.

Conversely, suppose that $\bar{G}$ has a vertex cover $V' \subseteq V$, where $|V'| = |V| - k$.
Then for all $u, v \in V$, if $(u, v) \in \bar{E}$, then $u \in V'$ or $v \in V'$ or both. The contrapositive for this implication is that for all $u, v \in V$, if $u \notin V'$ and $v \notin V'$, then $(u, v) \in E$. So, $V - V'$ is a clique and it has size $|V| - |V'| = k$.

Since VERTEX_COVER is NP_complete, we don't expect to find a polynomial time algorithm for finding a minimum size vertex cover.

y From above explanation we can say that vertex cover problem is NP-complete.

We have used instance from Clique to prove that NP-completeness of vertex_cover.

∴ We have proved that CLIQUE can be reduced to VERTEX_COVER, so we can say that CLIQUE problem is not harder ~~so~~ to solve than VERTEX_COVER.