RITIKA KUMARI
A20414073

HOMEWORK - 6

Problem 1:

Sol^n:

Given a collection of m finite sets $S_1, S_2, \ldots, S_m$, a cover is a subcollection of $S_1, S_2, \ldots, S_m$ such that the union of these sets equals the union of all sets.

Now, our goal is to find a polynomial time algorithm that finds a cover with minimum number of sets for any instance.

It is given that there is a black-box polynomial time algorithm for the decision-version of MINIMUM-SET-COVER which will take m finite sets $S_1, S_2, \ldots, S_m$ and a number k as imputs which returns a cover with k sets as output.

For finding a cover with minimum no. of sets, we are writing a polynomial time algorithm below:

(Assume $S = \{S_1, S_2, \ldots, S_m\}$

```
Find_Number_K (m)
    for (k = 1 to n)          where n = no. of elements
        BlackBox (S, k)
            if "YES"
            {
```

```
        return k ;
    } findset (m, k) ;
}

findset (m, k)      // function to get Minimum SET-COVER
  for ( i = 1  to  m)
  {
        remove $S_i$ from S
        // where $S = \{S_1, S_2, S_3, ---, S_m\}$
        BlackBox ( $S - S_i$ , k)
            if "No"
            {
                add $S_i$ back to S;
                $S = S_i + S$ ;
            }
  }
  return S ;
```

<u>Running time:</u> It will take polynomial time since BlackBox function is taking polynomial time and Find-Number-k will take $O(n)$ time, and FindSet will take $O(m)$ time. Therefore, ~~ove~~ overall algorithm will take polynomial time.

Proof of Correctness : We can proof the correctness by Induction : Lets' consider that Initially the given set i's empty , therefore in this Case none of the loop will get executed and we will get empty set from our algorithm. Therefore , we can say that, for base case, our algorithm is true. Now, considering the case when given sets are not empty , then the function "Find-Number-K" will given you ~~minimum~~ minimum value of K for which the given input will have a ~~vertex~~ cover. Now, In the function "FindSet" , the "Black Box" function will return you the Minimum set cover for the minimum value of K. Removing and adding sets in "FindSet" function will not affect the final output. Therefore, we can say that if the given input has multiple set-cover , then our algorithm will chose one of them depending on the set removal ordering. Hence, we can say that our algorithm produce correct result at each and every steps. Proved

Problem 2:

Solⁿ: For proving SET-COVERAGE is NP-hard. In we will reduce Vertex-cover to Set-cover first. If Vertex-cover can be reduced to Set-cover then in that case, we ~~case~~ can say that SET-COVERAGE is NP-hard, because it is known that Vertex-Cover is NP-hard.

For reducing Vertex-cover, we will follow certain steps :-

1. Consider a set U, which will have all the edges of Graph $G(V, E)$.
2. For each vertex $v \in V$, create subset S_v such that it will have all the incident edges of vertex v.
3. Now, take K (positive integer) indices ~~to~~ to find the vertex cover of size $K \in 1, \ldots, m$ such that the vertices corresponding to the Subsets in the subset cover of U, will be the vertex cover of G.

4. Now, we will check the eq^n given in the question ie, $\boxed{| U_{j=1}^{k} \; S_{i_j} | \geq R}$

This is the extra step which we are following for Set-Coverage which where we are taking the union of sets and comparing with the input R (positive integer) and if

$$| U_{j=1}^{k} \; S_{i_j} | \geq R$$

then we can say that there is a sol^n to the problem of SET COVERAGE.

For example, let's consider

$$S = \{ S_1, S_2, S_3, ----, S_k \}$$

$\therefore$ If we will do the union of $S_1, S_2, S_3, ---, S_k$ ie,

$$S_1 \cup S_2 \cup S_3 \cup --- S_k = U$$

where $U = \{ S_1, S_2, S_3, -----, S_n \}$

At the same time, we are also checking ~~$U \geq R$~~

$$S_1 \cup S_2 \cup S_3 \cup .. \cup S_k \geq R$$

Since S covers all the edges E of G, we can say that E = U

Now, for each $S_v \in S$ are having edges incident to vertex $v \in V$ and also verifying the condition

$\left| \bigcup_{j=1}^{K} S_j \right| \geq R$, then we can say that there is a sol^n to the problem of SET-COVERAGE, since there is a sol^n to the problem of Vertex-cover.

Hence, we can say that Vertex-cover can be reduced to Set-cover and Set-cover can be reduced to Set-coverage, therefore, Set-coverage is NP-hard, because it is known that Vertex-cover is NP-Hard. Proved

Problem 3.

Sol^n: In this problem, a directed graph $G(V, E, w)$ is given where $w(e)$ is defined as (possibly -ve) integer for each edge $e \in E$.

Our goal is to prove Shortest simple s-t path problem is NP-hard.

To prove this, we have to reduce the Hamiltonian path problem to Simple-shortest path problem, Since it is known that the Hamiltonian-path problem is NP-Hard.

Now, For reducing Hamiltonian Path problem to Shortest-simple path problem, we will follow the below steps:-

1. Let's consider a graph $G(V, E)$. Now we will convert original graph G to graph G'. We will add an extra vertex to G' called t' and also add a path from t to t' where we will assign value of $t \to t'$ edge as $|V| - 2$.

2. Now, all the existing edges of G is assigned a value of -1 in G'.

3. We will argue that G has a hamiltonian path s-t, if and only if, G' has -ve value path from s-t'.

4. Let's consider P is an Hamiltonian path s-t in G, therefore $PU(t \to t')$ is an hamiltonian path s-t' in G'. So, the value of s-t' will be $(-1 \times |V| + 1) + (|V| - 2) = -1$.

Likewise, if we will consider the reverse direction in G' then $t' \to t$ has a wt of $|V| - 2$, therefore, the ~~[illegible]~~ weight of hamiltonian path to become -ve, it need to accumulate $1 - |V|$ weight in the rest of the hamiltonian path. To acheive this, it need to visit ∀ vertices once and only ~~[illegible]~~ once ie, there must exist a simple-shortest path in the graph G'.

Hence we can say that Hamiltonian Path problem can be reduced to Simple-shortest-path problem.