ASSIGNMENT 6:

Design Analysis of Algorithm

Animesh Patni
A20403240

Problem 1:→

In this problem, I first prove that the problem of Minimum Set Cover is in NP. Suppose, we have sets $S = \{S_1, S_2 \dots S_m\}$ and an integer k. The certificate I chose is the set cover $S' \subseteq S$ itself. The verification algorithm confirms that $|S'| = Q$. and then checks for each set in S'. We can easily verify this in polynomial time.

Subroutine 1:→ [BLACK BOX DECISION PROBLEM]

Using the SET-COVER decision problem

Input → (S, K)

Output → YES OR NO.

$S \rightarrow S_1, S_2, \dots, S_m$.

$K \rightarrow$ Given size of the sets

Certificate for this problem is the list of sets used in covering.

Subroutine 2:→

Computing the size of the Minimum-set cover :→

```
1) Q = 1
2) while (SET-COVER (S, Q) == "NO"):
3)     Q = Q + 1
```

The above module starts from 1 and keep on increasing Q by 1 till the condition in while loop returns YES, which means we have found the minimum value of Q, that is the size of min-set-cover.

The idea I am going to use is, If we have a decision problem which we can for surely solve in polynomial time, we can find the optimal solution algorithm using that decision problem in polynomial time

MINIMUM-SET-COVER

Input → S, Q

Output → MINS → VALUES OF SETS that satisfies the definition in the problem

⇒ Finding the minimum set that satisfies the definition in the problem:

```
1   So = S
2   Q' = Q
3   FOR i in So:
4       Sj = So
5       IF (SET-COVER (Sj - i, Q' - 1) == "YES" AND Q' > 0):
6                   MINS ← MINS ∪ {i}
7                   So ← So - i
8                   Q' = Q' - 1
9       ELSE:
10                  So ← So - i
11  RETURN MINS
```

We know the value of Q which we calculated above. In line 3, I run the for through the sets which would run time atmost m, then in line

5, I am checking if the set-cover decision black-box algorithm return "YES" after removing that set from the original sets and even decreasing the value of Q' (= Q). If it is YES, then we union that set (i) in MINS, decrease the value of Q' and remove that set from So. We keep on repeating it, till all sets are done and value of Q is greater than 0. The output mins would contain the minimum set cover

Running time Analysis :→

Line 3 of the above algorithm is the major part of the algorithm and that for loop would run atmost

m times, in the worst case scenario. It is given in the problem that the black-box decision algorithm would run in polynomial time. So, the total running time of the algorithm would be O(m · time for black box decision algorithm).

Black-box decision algorithm in my case is SET-COVER.

Proof of Correctness :→

So, for this correctness proof, I am going to assign a cost of 1 to each set S_i and distribute this cover over the elements ——→. This cost will only be applied for the first time we cover element x in S_i. And this

would repeat till the algorithm terminates

To make this above part clear, suppose the first set that is added to mins is S_1 & S_1 has 10 distinct elements belonging X. So, we distribute cost 1 assigned to S_1, to every element in S_1 covered for the first time. Repeat this, till all the sets that satisfies line 5 is covered.

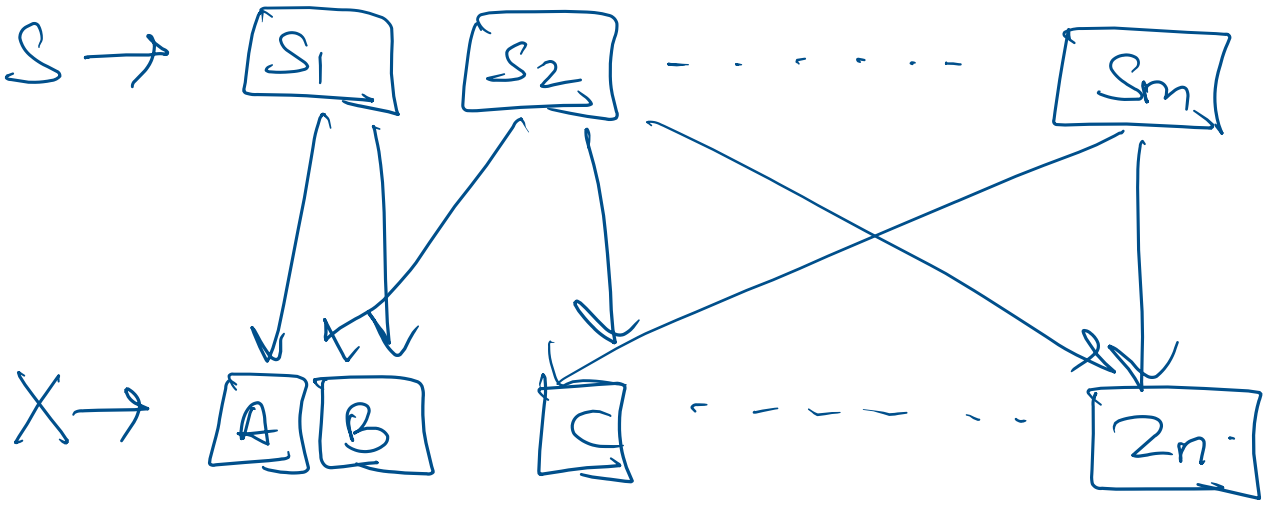
Now, suppose we have optimal solution mins*, we have list of sets as well as the elements covered in the optimal solution. Therefore,

$$|mins| = \sum_{S \in mins^*} \cdot \sum_{x \in S} cost.$$

where the cost of element x in

S can be defined as follows :→

$$cost_x = \frac{1}{|S_i - (S_1 \cup S_2 \cup \cdots \cup S_{i-1})|}$$



P.T.O. →

Problem 2:→

This problem needs reduction from vertex cover $\leq_p$ set cover and then from set cover $\leq_p$ set coverage. As theorm 7 discussed in the handout vertex cover $\leq_p$ set coverage

We run M_1 for vertex-cover to set-cover and M_2 for set cover to set-coverage

Suppose we have an instance of vertex-cover (G, z) and $G = (V, E)$ We can construct an instance of SET-COVER as follows using M_2:

1. $U = E$ (taking all the edges)
2. $z = z$ (find the vertex cover of size z
3. Creating a subset S_i for every

vertex i and taking all the edges that are incident to i.

Also constructing an instance of set-coverage using M_2 method on M_1

1. $U' = U$
2. $K = |U|$
3. Creating a subset S_{ij} for every set i and taking edges $j = 1$ to K. that are incident to i

This can all be done in polynomial time, as suppose m_1 reduces a problem instance w time being $O(w^k)$ and then m_2 reduces $m_2(w)$ in $O(w^k) + O(M_1(w)^{k'})$ which comes to $O(w^{k \cdot k'})$.

Now proving that the reduction is correct. Suppose we have a solution to vertex-cover (G, z) ie V' with size z. Then vertices in V' covers all the edges in E by definition of vertex cover, since S_i is the set of edges incident to i, so when we combine $S_1, S_2 \cdots S_m$ and the S_i where i belong to V' must contains all the edges E. Since our construction for set cover says all edges for the set U.

$\therefore$ S_i in which all i belong to V is a set cover of U with size z.

Now reducing from set cover

to set-coverage we use a polynomial time reduction algorithm that constraints S_i such that it takes only elements from 1 to k for i and $|\cup_{j=1}^{k} S_{ij}| \geq R$. S_{ij} contains some sets $U' \in U$ of size R.

Conversely :-

Suppose there is a solution for SET-COVERAGE $(U', \{S_1, S_2 \cdots S_m\}, R, z)$ be S_{ij} in which $i = 1$ to m and $j = 1$ to k. Union of the above sets = some sets in U, and size $>= R$.. As $S_{ij} \in S_i$ and $U' \in U$, S_i is a set-cover.

Then, we union $S_1 \cup S_2 \cdots \cup S_m = U$

Since $E = U$, S covers all the edges in E. As $S_i \in S$ and the edges are incident on vertex i.

$\therefore \{i : S_i \in S\}$ form a vertex cover of size z. Hence proved.

P.T.O. →

Problem 3:→

PART A:→

Given in the problem a directed graph $G = (V, E, w)$

$w(e) \rightarrow$ weight of each edge in G (positive & negative).

$s, t \rightarrow$ source and destination vertex that belongs to V.

$k \rightarrow$ total weight threshold.

So, we start by saying that Shortest Simple Path belongs to NP

Given an instance of problem, sequence of n vertices can be used as a certificate to verify the $s \rightarrow t$ path is shortest.

So, the verification algorithm goes like this:→

1. Checks whether the given sequence of vertices, with the edge cost & no vertex is repeated exists exactly once or not.

2. If point 1 is "yes" then, we sum all the weights in that path and check whether it is atmost k or not. If the sequence cost is greater than k we return NO else if,

→ sum <= k return YES:

```
if sum <= k:
    YES.
else
if sum > k:
    NO.
```

3. Input → {G, c, k}

   Output → YES OR NO

   G → {V, E}

   c → cost (s → t sequence)

   k → certificate

Sequence of vertices here means s → t

path, with no vertex repeated. The above verification problem can easily be done in polynomial time.

PART B:→

To prove shortest simple path is NP-hard. I am going to use hampath to reduce the shortest simple path problem.

HAMPATH $\leq_p$ SHORTEST-SIMPLE-PATH.

$G(V, E, w(e)), s, t, k \rightarrow G(V', E', w'(e)), s', t', k$

There is a function f that in polynomial time reduces to hampath to shortest simple path. Let G be an instance of hampath and from

that we create an instance where

$E' = \{(i, j) : i, j \in E \text{ and } i \neq j\} + 1$

V' = We add a supersource ss to vertex v.

$S' = \text{Supersource}(ss)$

$K = 1$ (because we take $(|E| - |E| + 1)$ we add 1 to E' for the edge from ss to S.

$t' = t$.

$$w'(e) = \begin{cases} -1 & \text{if } (e) \in E \\ 0 & \text{if } (e) \notin E \end{cases}$$

Then the instance for shortest simple path is then $\{G', w'(e), 1\}$ this above instance be created in polynomical time.

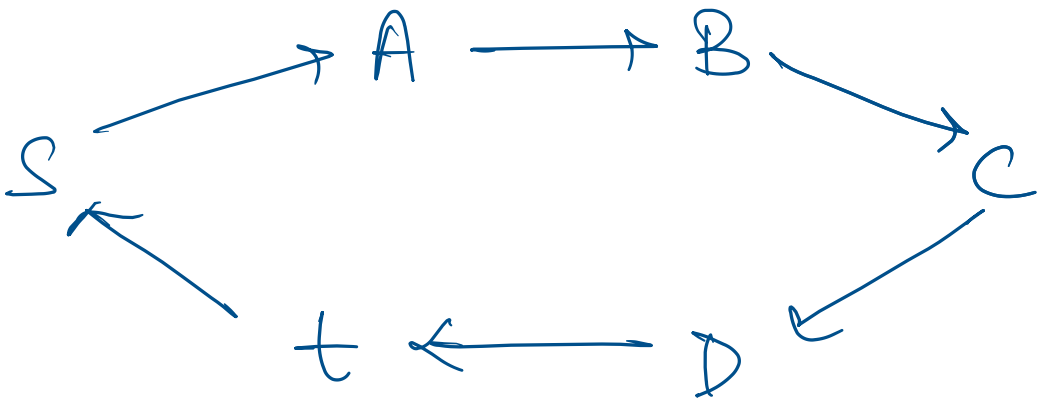
Now proving that G has a hampath h if and only if G' has a path from $s \to t$ of cost k

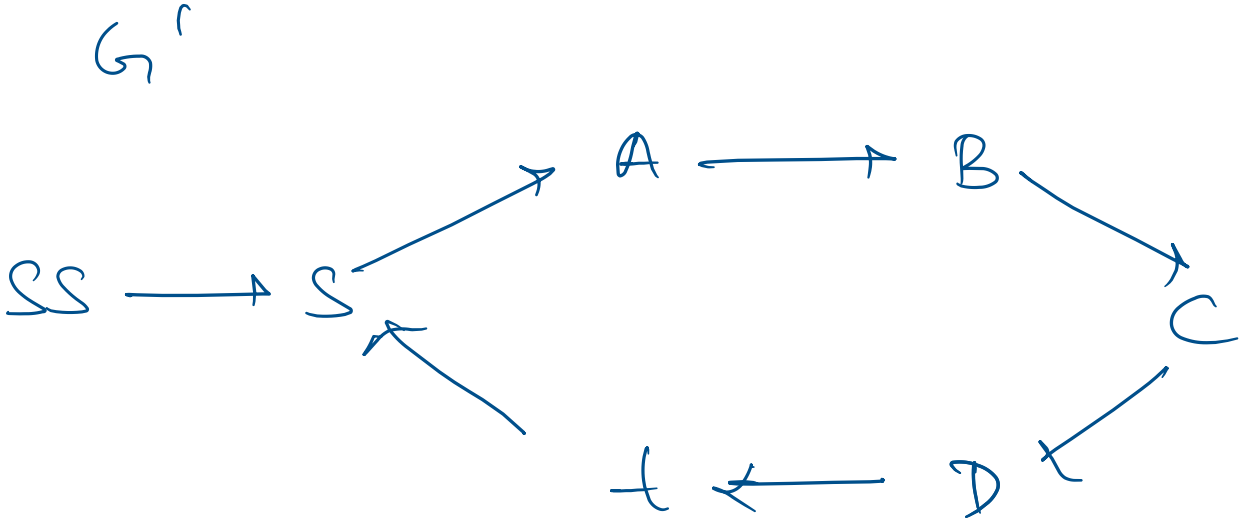
a.) Suppose G has a hampath h. So, each edge in h belongs to E. Each edge in h would also belong to E' plus the edge from superource s' to source. The cost of each edge in h would be -1 in G' and the total cost would be k (ie = 1). Thus h is the shortest simple path from $s \to t$; and h contains all the vertices in V exactly once. Proving conversely: Suppose G' has a shortest simple path h' from s' to t. So all

edges $s' \rightarrow t$ would belong E' and all the edges in h' removing the supersource belongs to E. And h' contains all the vertices in V' and h' contains all the vertices in V except S' exactly once

Hence proved.

G_1:




G'
A
B
SS
S
C
t
D