Aman Gupta
2019CS10673
Arpit Chauhan
2019CS10332

Assignment - 4
COL351 - Analysis and Design of Algorithms
Term: Fall 2021
Due Date: 13^{th} November, 2021

Problem 1
Let G = (V,E) be a directed graph with source s and $T = \{t_1, ..., t_k\} \subseteq V$ be a set of terminals. For any $X \subseteq E$, let r(X) denote the number of vertices $v \in T$ that remains reachable from s in G - X.

Give an O(|T|·|E|) time algorithm to find a set X of edges that minimizes the quantity r(X)+|X|. (Note that setting X equal to the empty-set is allowed).

Solution: Consider an auxiliary graph H = (V',E') which has all the vertices in set V plus an additional sink vertex t, say. Set E' has all the edges in set E plus k additional edges, $\{(t_1, t), (t_2, t) ..., (t_k, t)\}$. Hence, |V'| = |V| + 1 and |E'| = |E| + k. Let each edge e in the auxiliary graph H have capacity c(e) = 1.
Observe that for each vertex $v \in T = \{t_1, t_2, ..., t_k\}$ that is reachable from s in G, there exists a corresponding s-t path via v in the auxiliary graph H.

Algorithm 1: To find a set X of edges that minimizes the quantity r(X)+|X|

---

1. Construct auxiliary graph H = (V',E') by adding vertex t and edges $\{(t_1, t), (t_2, t) ..., (t_k, t)\}$ to the initial graph G = (V,E).
2. Apply the Ford-Fulkerson Algorithm for s-t Max-flow on the auxiliary graph H to obtain the Max-flow vector f and residual graph H_f
3. Now, find the set of all vertices that are reachable from s in the residual graph H_f by running DFS from the source vertex s. Let us call this set to be A. Let the set of remaining vertices H\A be B.
4. Min-cut C will be then, the set of all edges in H which have one end-point in A and other end-point in B, i.e. C = Set of all A-B edges
5. Return X = $C \cap E$, intersection of Min-cut C with the edge set E of the initial graph G

Runtime Analysis

Construction of the auxiliary graph H takes O(|E| + |T|) time. Next, we perform the Ford-Fulkerson algorithm which is know to take O((m+n)*Value of s-t Max-flow) time. The Max-flow is bound by |T| as the auxiliary graph H contains |T| edges to the sink t each with capacity 1. Since (m+n) is bound by O(|E|) for any connected graph, total time taken for Step 2 of the algorithm will be O((|E|+|T|)*|T|) which can be said to be O(|E|*|T|) for |T| = O(|E|).
In Step 3, we perform DFS which takes O(m+n) time which will be O(|E|) in our case. Step 4 also takes O(|E|) as we just need to find the set of A-B edges.
Finally in Step 5, we need to perform the intersection of sets C and E. We know that size of C is bound by O(|T|) by Max-flow Min-cut Theorem. So, even a naive O(1) space algorithm can find the intersection in O(|E|*|T|) time.
Hence, the algorithm takes O(|E|*|T|) time overall to find the required set X.

Proof of Correctness

Claim 1: $r(X) + |X| \geq Max - flow$ in the auxiliary graph H
Proof: Let F be the Max-flow in the auxiliary graph H. Since each edge was assigned a capacity of 1, there must exist F edge-disjoint paths from the source s to sink t. Now if we remove the edges that belong to the set X, then there will be still atleast F-|X| edge-disjoint paths from the source s to sink t and each such path corresponds to a distinct vertex in T being reachable from s in G-X. Hence, the number of vertices $v \in T$ that remain reachable from s in G - X will be atleast F-|X| i.e. $r(X) \geq F - |X|$ which implies that $r(X) + |X| \geq F$. Hence, the claim is proved.

Claim 2: Algorithm 1 returns X such that $r(X) + |X| = Max - flow$ in H

Proof: Consider the Min-cut C obtained in the step 4 of the algorithm. It consists of 2 parts, set X $= C \cap E$ and the set C$\backslash$X which correspond to those edges of C which are common with the set of additional edges $\{(t_1, t), (t_2, t)..., (t_k, t)\}$ that were added to G to obtain auxiliary graph H. Each of these edges in the set C$\backslash$X corresponds to a distinct edge in T being reachable from source s in G-X, hence r(X) $=$ |C$\backslash$X| $=$ |C| - |X|. Thus, r(X) $+$ |X| $=$ |C|.

From the Max-flow Min-cut Theorem, we know that (s,t) Max-flow value(say F) is equal to the capacity of the Min-cut. Since, each edge has a capacity of 1, |C| $=$ F. But we have already shown that r(X) $+$ |X| $=$ |C| for our case, hence we have proved that $r(X) + |X| = F$, the Max-flow value in H. Thus, the claim is correct.

Claim 1 suggests that the quantity r(X) $+$ |X| is atleast equal to the Max-flow in the auxiliary graph H and claim 2 tells us that our algorithm returns X such that r(X) $+$|X| $=$ Max-flow in H. Hence, our Algorithm is correct as it finds a set X of edges that minimizes the quantity r(X)$+$|X|.

Problem 2.1
Consider a set U $= \{u_1, ..., u_n\}$ of n elements and a collection $A_1, A_2, ..., A_m$ of subsets of U. That is, $Ai \subseteq U$, for $i \in [1, m]$. We say that a set $S \subseteq U$ is a hitting-set for the collection $A_1, A_2, ..., A_m$ if $S \cap A_i$ is non-empty for each i. The Hitting-Set Problem (HS) for the input $(U, A_1, ..., A_m)$ is to decide if there exists a hitting-set $S \subseteq U$ of size at most k.

Prove that Hitting-Set problem is in NP class.

Solution: To prove that the Hitting-Set problem is in NP class, we need to show that for every instance I of the HS problem, there is a polynomial time algorithm A with output in $\{"yes", "no"\}$, such that for any proposed solution S of length $O(|I|^c)$, A runs in $O(|I|^d)$ time over (I,S) to verify whether S is a valid solution to I.
Consider the following algorithm, A -

Algorithm 2: To check if S is a valid solution to the HS instance I

1 Let I $= (U, A_1, ..., A_m)$ and S be a proposed solution set.
2 Check whether $S \subseteq U$ and whether $|S| \leq k$. If not, Return "no".
3 For $i \in [1, m]$,
4 Check if A_i has atleast one element common with set S. If not , Return "no".
5 (Simply iterate over the set S for each element in A_i to check for common element.)
6 Return "yes"

Runtime Analysis

Naive way to perform Step 2 will take O(|S||U|) time as for each element in set S, we will check if it is present in U. Then for each $i \in [1, m]$, we can naively check for the presence of atleast one common element in O(|S|$|A_i|$) time. Hence, the for-loop takes total O(|S|*$\sum_{i=1}^{m} |A_i|$) time.
Thus, the overall algorithm takes O(|S|*(|U|+$\sum_{i=1}^{m} |A_i|$)) time which is polynomial in the input size.
We know that |U| = n and each of the sets $S, A_1, ..., A_m$ have their sizes bound by n as they are subsets of U. Hence, the algorithm can be said to be taking O(m*n^2) time.

Proof of Correctness

The algorithm A described above correctly verifies any proposed solution S as we are just checking the necessary conditions as per the definition of Hitting-Set described in the problem statement.

Since, we have found a polynomial time solution verifier for the HS problem, we have proved that the Hitting-Set problem is in NP class.

.

Problem 2.2
Prove that Hitting Set is NP-complete by reducing Vertex-cover to Hitting Set.

Solution: Consider an instance I_{VS} of the Vertex Cover problem, specified by the Graph G = (V,E) and the number k i.e. we need to find if there exists a set $W \subseteq V$ of size atmost k such that for each $(a, b) \in E$, one end-point of (a,b) lies in W.
Now, to prove that Hitting Set is NP-complete, we need to reduce this Vertex Cover instance to a Hitting Set instance.
Let us define the set U in the Hitting-set instance to be the Set V of the Vertex-cover instance with n = |V|. Let m = |E|, then for each edge $e_i = (a, b) \in E$, consider $A_i = \{a, b\}$ in the Hitting-set instance. Hence, we have obtained a collection $A_1, A_2, ..., A_m$ of subsets of U. We need to find a set $S \subseteq U$ of size atmost k such that $S \cap A_i$ is non-empty for each i. In this way, for every instance I_{VS} of the Vertex-cover problem, we get a corresponding instance I_{HS} of the Hitting-set problem.

Claim 1: There exists a Hitting set of size atmost k for instance I_{HS} if and only if there exists a Vertex Cover of size atmost k for instance I_{VS}.
Proof: If we have a Vertex Cover $W \subseteq V$ of size atmost k for instance I_{VS}, then consider Hitting set S to be equal to set W. Since $W \subseteq V$, $S \subseteq U$ and $|S| \leq k$. Also, since each edge has atleast one vertex in W, each set A_i will have atleast one element common with the Set S. Hence, S is a valid Hitting Set of size atmost k for I_{HS} .
Conversely, if we have a Hitting set $S \subseteq U$ of size atmost k for instance I_{HS}, then consider Vertex Cover W to be equal to set S. Since $S \subseteq U$, $W \subseteq V$ and $|W| \leq k$. Also since each set A_i has atleast one element common with the set S, each edge e_i will have atleast one vertex in W. Hence, W is a valid Vertex Cover of size atmost k for I_{VC}.
Thus, the claim is proved.

Using claim 1, we can say that any Vertex-cover instance can be reduced to a corresponding Hitting Set instance. As discussed in class, the Vertex-cover problem is an NP-complete problem. Therefore, for any problem X in the NP class, we know that X $\leq_P$ Vertex-cover problem. So by transitivity of polynomial reductions, any problem X in NP class is reducible to HS problem, i.e. X $\leq_P$ HS problem.
Hence,we have proved that Hitting Set is NP-complete problem.

Problem 3.1
Given an undirected graph G = (V, E), a feedback-set is a set $X \subseteq V$ satisfying that $G \setminus X$ has no cycle. The Undirected Feedback Set Problem (UFS) asks: Given G and k, does there exist a feedback set of size at most k.
Prove that Undirected Feedback Set Problem is in NP class.

Solution:
The Undirected Feedback set problem is a decision problem which will have a yes answer iff there exists a set of vertices X such that the sub-graph $G \setminus X$ has no cycles.
We need to prove that the problem lies in the NP class of problem, for which we must show that there exists a polynomial time verifier algorithm for the UFS problem. As discussed in lectures, an algorithm A is called polynomially verifiable iff for any instance of the problem I (let |I| = n) and a proposed solution "s" of polynomial (w.r.t n) size, algorithm A should be able to run in polynomial (w.r.t n) time to determine the decision of (I,s).

In the context of this problem the above result boils down to the following, we are given an undirected graph G = (V,E) and a vertex set X, $X \subseteq V$. We need to show that there exists an algorithm that can show that the sub-graph $G \setminus X$ has no cycle in polynomial time.
We will use the following algorithm:

Algorithm 3: Check whether $G \setminus X$ has a cycle or not, given graph G and vertex set X.

```
Remove all the vertices in the set X from the graph G. Let the remaining subgraph be T.
Choose a random vertex v in T.
Perform DFS(v) (Depth-First-Search)
If a back edge is found during the DFS traversal return "No"
Else return "Yes"
```

We note that the above algorithm uses DFS to find whether or not there exists a cycle in the sub-graph or not. A cycle can occur in an undirected graph if and only if there are back edges in the graph. The above algorithm runs in $\mathbf{O(|V|+|E|)}$ time. Since the number of edges in a graph can only be of the order of n^2, the overall time complexity of the algorithm is $\mathbf{O}(n^2)$. Hence, the UFS problem belongs to NP set.

Problem 3.2
Prove that Undirected Feedback Set Problem is NP-complete by reducing Vertex-cover to Undirected Feedback Set Problem.

Solution: We will show that Undirected Feedback set problem is an NP-complete problem by reducing Vertex cover problem to UFS problem.

Claim 1: Vertex-Set Problem $\leq_P$ Undirected Feedback Set Problem
Proof:
To show that a problem A is reducible to problem B, A$\leq_P$B we need to show the following,
There exists a polynomial function f such that,
i) An instance of problem A, I should map to an instance f(I) of problem B
ii) Instance I of A is "yes" iff instance f(I) of B is "yes"

In the context of vertex set and UFS problems consider the function f to be the following:
For a given graph G=(V,E) consider a graph H such that H contains all the edges and vertices of G, additionally for every edge (x,y) in G, H has an additional vertex z, which is connected to vertices x and y. One can imagine it as an additional triangle on every edge of graph G.

Now, that we have obtained a mapping from instances of Vertex set problem (G) to instances in UFS problem (H) we need to show than a proposed solution S of size k for the vertex set problem is decided as "yes" iff there exits an UFS set of size k in the H.

Claim 1a: If S is vertex cover set in G then S is an UFS in graph H.
Proof: Consider a cycle C in the graph H. There will exist two vertices x,y on the cycle C such that (x,y) is an edge in G, because all the new vertices added during making graph H are unconnected to each other. Since (x,y) is an edge in G, one of its end-points must be present in set S, as its a vertex cover. So we have shown that the vertex set S in H passes though any arbitrary cycle C. Hence after removing S from H, no cycle will be left. Therefore, S is a valid UFS of graph H.

Claim 1b: If S is an UFS of size k in graph H then there exist a vertex set of size k in G.
Proof: Suppose that S contains one of the vertices z added during making graph H. Vertex z corresponds to edge (x,y). Any cycle that passes through z also passes through vertices x and y, because z is only connected to those two vertices. Hence we can make another UFS in H by replacing z with either x or y. In a similar way we can replace all the newly added vertices from S to form a new UFS S' of H that contains only the common vertices between G and H. Since x,y,z is a cycle S' must contain one out of these three vertices. Since it contains only the common vertices it should have either x or y in it. This is true for any arbitrary edge of G. Therefore, S' is a vertex cover set of graph G of size k.

Using the claims 1a and 1b we can say any proposed solution S of size k for the vertex set problem is decided as "yes" iff there exits an UFS set of size k in the graph H. Therefore the vertex set problem is reducible to the UFS problem. Hence the claim 1 is proved.

The UFS problems belongs to the NP class as proved in the problem 3.1 . Also, according to the claim 1 the vertex set problem is reducible to the UFS problem. As discussed in class, the Vertex-cover problem is an NP-complete problem. Therefore, for any problem X in the NP class, we know that X $\leq_P$ Vertex-cover problem. So by transitivity of polynomial reductions any problem X in NP class is reducible to UFS problem, i.e. X $\leq_P$ UFS problem.
Hence, we have proved that the Undirected Feedback Set problem belongs to the NP-complete set of problems.
