CS5800 Algorithms *Out: 20 November 2021*

Solutions to Problem Set 6

Name: *Due: 30 November 2021*

Group members:

Problem 1 (Reducing 2-coloring to SAT) 15

Problem (2-Coloring)**.** *Given an undirected graph $G = (V, E)$, where the graph has $|V|$ nodes and $|E|$ edges, is there a way to color the nodes V either black or white such that for every edge, it has two different colors on its endpoints?*

Problem (SAT)**.** *Given n variables $x_1, x_2, \ldots, x_n$, and m clauses $C_1, C_2, \ldots, C_m$ with each clause is an OR between some literals z_i and each literal is either a variable or the negation of a variable, does there exist an assignment of True/False to the variables such that $C_1 \wedge C_2 \wedge \cdots \wedge C_m$ is True?*

a. Reduce the 2-Coloring problem to SAT. That is, write a polynomial sized set of clauses (need not have 3 literals each) using a polynomial number of variables such that if there is a satisfying assignment if and only if there is a 2-Coloring. Briefly describe what the variables and clauses represent.

   [***Hint:*** *Note that $(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$ is satisfied when exactly one of x_1, x_2 are true. Use this fact to enforce the coloring constraint.*]

b. Does this reduction imply that 2-Coloring is NP-Complete? Why?

Solution:

The SAT formula can be constructed by running a BFS traversal over the graph. For each node $v \in V$, assign a variable x_v for it if it is unvisited and a variable x_u for node u where $(v, u) \in E$ if u is unvisited. Then add two clauses $(x_v \vee x_u)$ and $(\neg x_v \vee \neg x_u)$ to Φ. In this way, it can be ensured that the number of variables is $|V|$ and the number of clauses is $2|E|$, both of which are polynomial. Each variable represents a node and each pair of clauses represent an edge. If there exists a solution to Φ then it also guarantees a solution to the 2-coloring problem, where all nodes assigned with *true* will be white and nodes of $false$ be black. ■

Problem 2 (Reductions via Restrictions) 10+10

A common strategy to prove NP-Completeness of new problems is to show that problems already known to be NP-Complete are in fact special cases of the new problem. This is often referred to as the "restriction" strategy, where the new problem is restricted to be equivalent to a known NP-Complete problem. Use the "restriction" strategy to prove the following new problems to be NP-Complete by reducing from the given known NP-Complete problem. You only need to provide the specific manner in which these are restrictions along with a brief description of correctness.

1. Give a poly-time reduction to Subgraph Isomorphism from Clique.

   Problem (Subgraph Isomorphism)**.** *Given two graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$, does G contain a subgraph that is isomorphic to H? That is, is there a subset $V \subseteq V_G$ and a subset $E \subseteq E_G$ such that $|V| = |V_H|, |E| = |E_H|$, and there exists a one-to-one function $f : V_H \to V$ satisfying $\{u, v\} \in E_h$ if and only if $\{f(u), f(v)\} \in E$?*

Problem (Clique). *Given a graph $G = (V, E)$ and a positive integer $k \leq |V|$, does G contains a clique V' having $|V'| \geq k$? That is, is there a subset $V' \subseteq V$ such that, for every pair of vertices $u, v \in V'$, the edge $\{u, v\} \in E$.*

2. Give a poly-time reduction to Multiprocessor Scheduling from Partition.

Problem (Multiprocessor Scheduling). *Given a finite set A of "tasks", a positive integer "length" $\ell(a)$ for each "task" $a \in A$, a positive integer m of "processors", and a positive integer "deadline" D, is there a partition (of the "tasks" into the m "processors") $A = A_1 \cup A_2 \cup \cdots \cup A_m$ of A into m disjoint sets such that each "processor" finishes all of its "tasks" by the "deadline":*

$$\max_{1 \leq i \leq m} \left(\sum_{a \in A_i} \ell(a) \right) \leq D?$$

Problem (Partition). *Given a set A, a positive integral size function $s(a)$ for all elements $a \in A$, is there a partition of A into two subsets A_1, A_2 so that the sum of sizes in A_1 is equal to that in A_2, that is, $\sum_{i \in A_1} s(i) = \sum_{j \in A_2} s(j)$?*

Solution:

1. The Clique problem is reducible to Subgraph Isomorphism problem in the way that if we can find a subset $V \in V_G$ that is isomorphic to H which is fully-connected then the subset V is automatically a clique to G for $k = |V_H|$ because every pair of nodes in V is guaranteed to constitute an edge. The restriction is that a clique must be a fully-connected subgraph.

2. The Partition problem is a special case of Multiprocessor Scheduling problem by the restrictions $m = 2$ and $D = \frac{1}{2} \sum_{a \in A} \ell(a)$. If there exists a solution to the multiprocessor scheduling problem with given restrictions, then the solution also satisfies the partition problem.

■

Problem 3 (K-weight path in a DAG) 10+15+5

In class you learned that if we can reduce a NP-complete problem A to problem B through a polynomial time reduction, then problem B must be at least as hard as A, or NP-hard[1]. In this problem, we'll use a reduction to show that a problem is NP-hard using a reduction from an NP-complete problem.

The NP-complete problem you are given is Subset Sum, which you have seen (or will see) in class. In the Subset Sum problem you are given an input array A of n positive integers and an integer T, and you must decide whether there exists a subset of indicies in $\{1, ..., n\}$ such that the sum of A of these indices is exactly T. That is, given A, n, and T you should return true if and only if

$$\exists S \subseteq \{1, ..., n\} \text{ such that } \sum_{s \in S} A[s] = T.$$

We wish to prove that the K-Path problem is NP-hard using a reduction from Subset Sum. In the K-Path problem you are given a DAG (directed acyclic graph) $G = (V, E)$ with non-negative edge weights $w : E \to \mathbb{N}$ and a pair of vertices s and t in V, and you must decide whether there exists an

[1] To show NP-Completeness, you must further show that B is in NP, that is, given a solution to B, we can verify whether the solution is correct in polynomial time.

s-t path in G whose total weight is K. That is, given G, w, s, t, and K you should return true if and only if

$$\exists \text{ an } s\text{-}t \text{ path } P \text{ in } G \text{ such that } \sum_{e \in E(P)} w(e) = K.$$

Show that the K-PATH problem is NP-hard using a reduction from SUBSET SUM. That is, given an instance $SS = (A, n, T)$ for SUBSET SUM, show how to transform that into an instance $KP = (G, w, s, t, K)$ for K-PATH such that SUBSET SUM returns true on SS if and only if K-PATH returns true on KP.

a) Show how to polynomially reduce an instance of SUBSET SUM into an instance of K-PATH. Make sure that your output is a valid instance of K-PATH (i.e. all weights are non-negative and s and t are well-defined).

b) Show that your reduction is correct. That is, show that SUBSET SUM returns true on an input instance to your reduction if and only if K-PATH returns true on the output of your reduction.

c) Show that your reduction takes polynomial time.

Problem 4 (Camp Recruiting Problem) 5+10+15+5

Suppose you're helping to organize a summer sports camp, and the following problem comes up. The camp covers a set $S = \{\text{baseball, volleyball, } \dots\}$ of n sports. For each of the n sports in S, the camp is supposed to have at least two counselors who are skilled in that sport. The camp has received a set C of m job applications from potential counselors. For each sports $s \in S$, there is some subset $A_s \subseteq C$ of the counselor applicants qualified in that sport.

We define the Efficient Recruiting problem as: For a given number $1 \leq k \leq m$, is it possible to hire at most k of the counselors and have at least 2 counselors qualified in each sports in S? More formally, given a number $1 \leq k \leq m$, is there a subset $H \subseteq C$, with $|H| = k$ such that for all $s \in S$, $A_s \cap H \geq 2$ (every sports has at least two counselors hired who are skilled in it).

Show that the Efficient Recruiting problem is NP-Complete by reducing from Set Cover. In particular show each of the following:

a) Show that Efficient Recruiting is in NP.

b) Show how to polynomially reduce an instance of Set Cover into an instance of Efficient Recruiting. Make sure that your output is a valid instance of Efficient Recruiting.

c) Show that your reduction is correct. That is, show that Set Covers returns true on an input instance to your reduction if and only if Efficient Recruiting returns true on the output of your reduction.

d) Show that your reduction takes polynomial time.