COMPSCI330 Design and Analysis of Algorithms
Assignment 9

Due Date: Wednesday April 24, 2019

Guidelines

- **Reductions** For any reduction from problem A to problem B, you need to first clearly specify the direction of reduction (if it was not given in the problem), then follow 3 steps: (a) show how you can transform an instance X of A to an instance Y of B; (b) show if answer to X is YES, answer to Y is also YES; (c) show if answer to Y is YES, answer to X is also YES.
- **Typesetting and Submission** Please submit the problems to GradeScope. You will be asked to **label your solution for individual problems**. Failing to label your solution can cost you 5% of the total points (3 points out of 60 for this homework).
- LaTeX is preferred, but answers typed with other software and converted to pdf is also accepted. Please make sure you submit to the correct problem, and your file can be opened by standard pdf reader. **Handwritten answers or pdf files that cannot be opened will not be graded.**
- **Timing** Please start early. The problems are difficult and they can take hours to solve. The time you spend on finding the proof can be much longer than the time to write. If you need more time for your homework please use this form and submit a STINF.
- **Collaboration Policy** Please check this page for the collaboration policy. You are **not** allowed to discuss homework problems in groups of more than 3 students. **Failure to adhere to these guidelines will be promptly reported to the relevant authority without exception.**

Problem 1 (BALANCED KNAPSACK)**.** (15 points) In this problem, we are going to consider two problems related to knapsack. In the KNAPSACK problem, we are given a knapsack with capacity $c > 0$, and n items with weights $w_1, w_2, ..., w_n \geq 0$ (you can assume $\sum_{i=1}^{n} w_i \geq c$). The answer to the KNAPSACK problem is YES, if and only if there exists a subset $S \subset \{1, 2, ..., n\}$ of items such that the total weight of this subset is exactly equal to the capacity c. That is, $\sum_{i \in S} w_i = c$.

KNAPSACK problem is known to be NP-complete. (Of course, you may wonder why that is the case since we have designed a dynamic programming algorithm for this problem. However, the running time of the DP algorithm is $O(nc)$, but in this problem the input c is given in binary, so the actually value of c might be exponential in the input length. The DP algorithm is therefore not considered to be a polynomial time algorithm.)

Now consider a related problem called BALANCED KNAPSACK. In this problem, you are given n items with weights $w_1, w_2, ..., w_n \geq 0$. The answer to the BALANCED KNAPSACK problem is YES, if and only if you can partition the items into two knapsacks so that the two knapsacks have exactly the same weight.

Use induction to show BALANCED KNAPSACK is NP-hard.

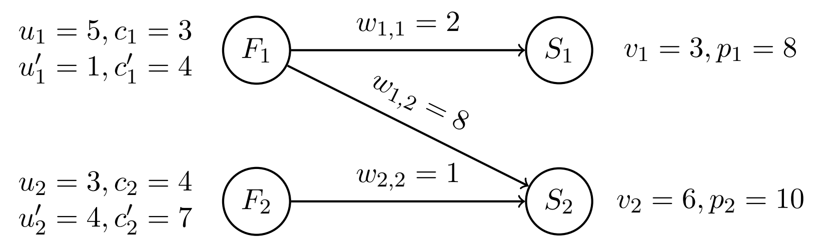
Problem 2 (SET COVER and DOMINATING SET)**.** (25 points) SET COVER is a classical NP-complete problem. In this problem, there are n sets $S_1, S_2, ..., S_n$. All of these n sets are subsets of $\{1, 2, ..., m\}$. The input of the problem is $n, m, k (1 \leq k \leq n)$ and a list of elements for each set. The answer to SET COVER is YES, if and only if there is a way to select k out of the n sets $(S_{i_1}, S_{i_2}, ..., S_{i_k})$ such that the union of these sets covers every element in $\{1, 2, ..., m\}$. That is, $\cup_{j=1}^{k} S_{i_j} = \{1, 2, ..., m\}$. You can assume that every element appears in at least one set.

DOMINATING SET is a graph problem. The input is an undirected graph with n vertices and a number k $(1 \leq k \leq n)$. A set D of vertices is called a dominating set, if every vertex $v \in V$ is either in the dominating set D, or is adjacent to another vertex u that is in the dominating set D.The answer to DOMINATING SET is YES if and only if there is a dominating set of size k.
(a)(5 points) Prove that DOMINATING SET is in NP.
(b) (20 points) Reduce SET COVER to DOMINATING SET problem.

Problem 3 (Shipping Products Revisited)**.** (20 points) As promised we are looking at the problem in the midterm 2 again. Let's first recall the description of the problem (we will now be in the setting of part (c)).

Company X is trying to figure out how to ship their product from factories to shops. There are n_1 factories and n_2 shops. Factory F_i $(i = 1, 2, ..., n_1)$ can produce at most u_i products at the cost of $c_i \geq 0$ per unit. After factory F_i has produced u_i items, it can produce u'_i more items at the cost of c'_i per unit (note that $c'_i \geq 0$ can be smaller than c_i. Shop S_j $(j = 1, 2, ..., n_2)$ can sell at most v_j products at the price of $p_j \geq 0$ per unit. However, transportation is only available for a limited pairs of factories and shops - you are given a list of possible transportation routes $E = \{(i_1, j_1), (i_2, j_2), ..., (i_m, j_m)\}$. Each route $(i, j) \in E$ has a cost of $w_{i,j} \geq 0$ per unit, where transporting each product from factory F_i to shop S_j costs $w_{i,j}$. See Figure below for an example.

In this case the optimal solution would produce 3 products at F_1 and 6 products at F_2. The 3 products at F_1 will be shipped to S_1 and 6 products at F_2 will be shipped to S_2. The total profit for this solution is 30.



The decision version of this problem (which we call SHIPPING PRODUCTS) is: given the input graph, $u_i, c_i, u'_i, c'_i, v_j, p_j, w_{i,j}$, and a profit threshold V, the answer is YES if and only if there is a way to make a profit of exactly V.

Based on the fact that SET COVER is NP-complete, prove that SHIPPING PRODUCTS is NP-hard.