CS 6363: Homework 4, Part 2

Version 1.0

Instructor: Benjamin Raichel

Due Tuesday December 4, at 1:00 pm

Homework is to be submitted online through eLearning by the above deadline. Typeset solutions (in LaTeX) are strongly preferred, but not required. If you chose to hand wright and scan your solutions, if what is written is illegible you will be marked down accordingly. Please make sure your name is on each page of the submission. You do not need to restate the problem, just the problem number.

Explanations are to be given for each problem, unless the problem specifically states it is not necessary. Explanations should be clear and concise. Rambling and/or imprecise explanations will be marked down accordingly.

Problem 5. Flow and Cut Problems (22 points)

For each part your answer should be clearly explained, though a formal proof of correctness is not required.

(a) **(6 points) Class Scheduling:** Let S denote the set of students at UTD, and let C denote the set of classes. For a student $x \in S$, let $C(x)$ denote the set of classes that student x is interested in taking. The school policy is that each student can take at most 5 classes per semester. Additionally, each class can have at most 20 enrolled students. Naturally, a student can register for a specific class at most once.

The University's goal is to maximize class enrollment, i.e. the sum over all classes of the enrolled total in each class. (Note this may mean some students won't be able to register for any courses, and some courses may be empty.)

Describe how to use maximum flow to meet this goal, that is you need to specify the vertices, edges, and capacities in a flow network.

(b) **(9 points) Unique Cuts:** Suppose you have already computed a maximum flow f^*. Recall from class that in a flow network G any maximum flow saturates all minimum cuts in G. Using this fact and the precomputed f^*, give a linear time algorithm to determine whether there is a unique minimum cut in G.

(c) **(7 points) Double Target Flow:** In a standard flow network, we are given a directed graph $G = (V, E)$, which has a non-negative capacity $c(e)$ on each edge $e \in E$, a source vertex s and a target vertex t. A flow is an assignment of non-negative values to the edges, such that there is flow conservation at all $V \setminus \{s, t\}$, and no capacity is violated. A max flow is a valid flow maximizing the net amount of flow leaving s.

A double target flow network is the same except now there are two targets, t_1 and t_2, and instead we require flow conservation at the vertices in $V \setminus \{s, t_1, t_2\}$ (and capacity constraints are still enforced). Just as in the single target case, a max flow in a double target flow network is a valid flow maximizing the net flow leaving s.

Give an algorithm to compute the max flow in a double target flow network, assuming you have a black box which correctly computes the max flow in any standard flow network.

NP-Completeness

The following definitions and problems are from class, and are listed here as a reference.

Definitions:

- A boolean formula is in kCNF form if it is the AND of several clauses, where each clause is the OR of exactly k literals, where a literal is a variable or its negation.
  For example, the following is a 3CNF formula:
  $$(a \vee b \vee \overline{c}) \wedge (d \vee \overline{b} \vee \overline{e}) \wedge (\overline{a} \vee b \vee \overline{e}) \wedge (c \vee \overline{d} \vee e)$$
- Given an undirected graph $G = (V, E)$, an *independent set* is a subset $I \subseteq V$ such that for each pair $u, v \in I$, $\{u, v\}$ is *not* an edge in E (i.e. the induced subgraph of I has no edges)
- A vertex cover is a subset $W \subseteq V$ such that for each edge $\{u, v\} \in E$, either $u \in W$, $v \in W$, or both. (i.e. a set of vertices which "covers" all the edges).

The following problems are NP-complete.

Problem: 3SAT

Instance: A boolean formula $f(x_1, \ldots, x_n)$ in 3CNF form.
Question: Is there an assignment to $x_1, \ldots, x_n$ s.t. $f(x_1, \ldots, x_n) = 1$?

Problem: Independent-Set

Instance: An undirected graph $G = (V, E)$, and an integer k.
Question: Does G contain an independent set of size $\geq k$?

Problem: Vertex-Cover

Instance: An undirected graph $G = (V, E)$, and an integer k.
Question: Does G have a vertex cover which uses $\leq k$ vertices?

Problem: Subset Sum

Instance: A set $X = \{x_1, \ldots x_n\}$ of positive integers and an integer t.
Question: Is there a subset of X whose sum is equal to t?

Problem 6. NP-completeness (32 Points)

(a) (7 points) The Independent-Set problem described above is an NP-complete decision problem. On the other hand in the optimization problem Max-Independent-Set you are asked to find the largest possible independent set in G (i.e. the output is no longer binary and is instead a set of vertices, and there is no input value k).

You are given a black box solving the Independent-Set problem on any instance (G, k), i.e. you can query the black box on any graph G and value k and it will tell you if there is an independent set of size $\geq k$ in G. Give an algorithm to solve the Max-Independent-Set problem using only a polynomial number of calls to this black box.

For the remaining problems, your reductions should be clearly explained, though a formal proof of correctness in not required.

(b) (7 points)

Problem: 4SAT

Instance: A boolean formula $f(x_1, \ldots, x_n)$ in 4CNF form.	
Question: Is there an assignment to $x_1, \ldots, x_n$ s.t. $f(x_1, \ldots, x_n) = 1$?	

Prove that 4SAT is NP-hard. Remember each clause must contain *exactly* 4 literals.

(c) (9 points)

Problem: Set-Cover

Instance: A collection C of subsets of a finite set S, and an integer k.
Question: Are there k subsets $S_1, \ldots, S_k$ from C such that $\cup_{i=1}^{k} S_i = S$.

Prove that Set-Cover is NP-hard. [Hint: reduce from Vertex-Cover.]

(d) (9 points)

Problem: Knapsack

Instance: An array of values $V[1 \ldots n]$, an array of sizes $S[1 \ldots n]$, a bag size B, and a number k.
Question: Is there a subset of indices I such that $\sum_{i \in I} V[i] \geq k$ and $\sum_{i \in I} S[i] \leq B$.

Prove that Knapsack is NP-hard. [Hint: reduce from Subset Sum.]