CS 6363: Homework 4

Version 1.0

Instructor: Benjamin Raichel

Due November 28, at 11:59 pm

Homework is to be submitted online through eLearning by the above deadline, and on paper in the following class. 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 1. Running the algorithms (15 points)

In this problem you will simulate Dijkstra and Bellman-Ford on a graph.
For clarity, these algorithms are as follows:

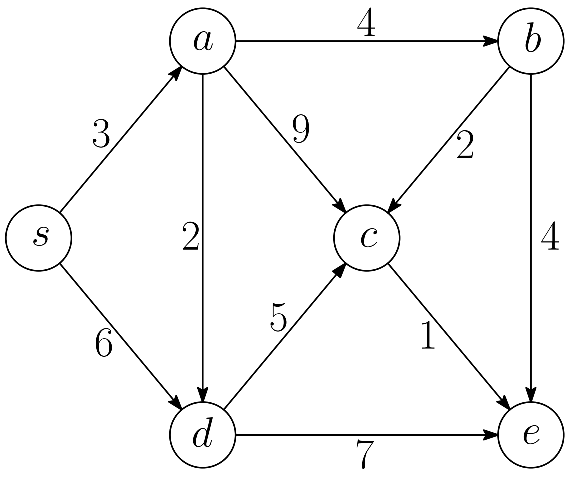
```
procedure RELAX($u \to v$)
    $dist(v) = dist(u) + w(u \to v)$
    $pred(v) = u$
```

```
procedure INITSSSP($s$)
    $dist(s) = 0$
    $pred(s) = NULL$
    for all vertices $v \neq s$ do
        $dist(v) = \infty$
        $pred(v) = NULL$
```

```
procedure DIJKSTRA($s$)
    InitSSSP($s$)
    Init min $dist()$ priority queue, $Q$
    Put $s$ in $Q$
    while $Q$ not empty do
        $u = Q.Pop()$
        for all edges $u \to v$ do
            if $u \to v$ tense then
                Relax($u \to v$)
                Put $v$ in $Q$
```

```
procedure BELLMAN-FORD($s$)
    InitSSSP(s)
    loop $|V|$ times
        for every edge $u \to v$ do
            if $u \to v$ tense then
                Relax($u \to v$)
    for every edge $u \to v$ do
        if $u \to v$ tense then
            return "Negative cycle!"
```

The input graph is the following:



(a) (9 points) First simulate Dijkstra's algorithm by filling out the following table. Each column represents a single round of Dijkstra's algorithm. In the top row write the "active" vertex of that round, i.e. the one popped from the queue which is the one being set as the predecessor. Then for the other column entries, write the current distance values, where you only need to fill in the entry if it changed since the previous round. The first round, where s is popped off the queue, has already been filled in. No explanation required.

Active	Null	s					
s	0						
a	∞	3					
b	∞						
c	∞						
d	∞	6					
e	∞						

(b) (6 points) Bellman-Ford does not specify which order it loops over the edges, but if the right order is chosen Bellman-Ford only does a single loop. Give an ordering of the edges in G such that after a single round of the Bellman-Ford algorithm the $pred(v)$ and $dist(v)$ values correctly describe a shortest path tree. No explanation required.

Problem 2. SSSP (15 points)

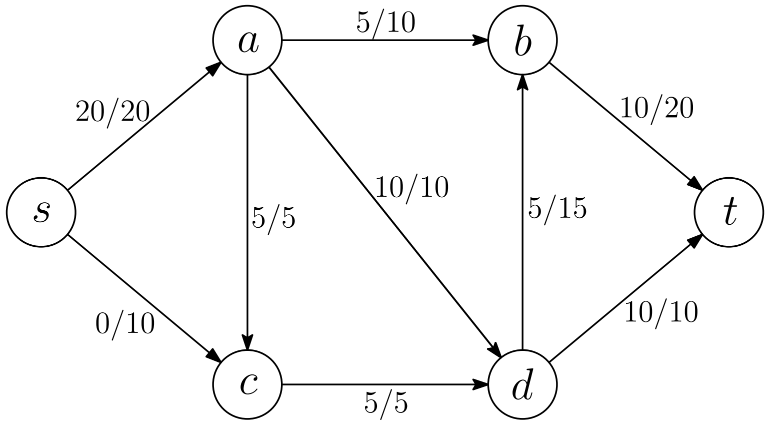
Let $G = (V, E)$ be a directed graph with positive edge weights, and let s be an arbitrary vertex of G. Your roommate claims to have written a program which computes the single source shortest path tree from s. Specifically, for every vertex $v \in V$, your roommate tells you the $pred(v)$ and $dist(v)$ values output by the program. Unfortunately, your roommate is not that reliable and so you want to check their solution. Give an $O(|V| + |E|)$ time algorithm to check whether the given values correctly describe some shortest path tree of G.

For simplicity you can assume that all vertices in the graph are reachable from s, and hence for every vertex in your roommates solution $dist(v) \neq \infty$, and moreover s is the only vertex such that $pred(s) = NULL$.

Problem 3. Computing Flows and Cuts (10 points)

No explanation required for any part of this problem.

(a) Draw the residual graph of the following flow network.



(b) Give the ordered list of vertices of an augmenting path in your residual graph.

(c) Give a minimum s-t cut (i.e. give the vertex bi-partition).

Problem 4. Flow and Cut Problems (25 points)

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

(a) **(8 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 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) **(8 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.

(c) **(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.

NP-Completeness

The following definitions will be covered in class, but you will need them now to answer the remaining questions on this homework.

Definitions:

- A boolean formula is in kCNF form if it is the OR of several clauses, where each clause is the AND of exactly k literals, where a literal is a variable or its negation.
  For example, following is a 3CNF formula:

$$(a \wedge b \wedge \overline{c}) \vee (d \wedge \overline{b} \wedge \overline{e}) \vee (\overline{a} \wedge b \wedge \overline{e}) \vee (c \wedge \overline{d} \wedge e)$$

- 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. We will prove this in class, though you do not need to know those proofs to answer the questions on this homework.

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 5. NP-completeness (35 Points)

(a) (5 points) On the second homework you developed an $O(nb)$ time dynamic programming algorithm for the (non-fractional) jewel thief problem. This is more commonly referred to as the knapsack problem, and is a well known NP-complete problem. Explain why having an $O(nb)$ time algorithm for knapsack does not imply P=NP. (Your answer should be very short.)

(b) (5 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 determined the size of the largest possible independent set in G (i.e. the output is no longer binary, and there is no input value k).

Given a black box which solves the Independent-Set problem on G, explain how to solve the Max-Independent-Set problem. Your algorithm should use as few calls to the black box as possible. (Your answer should be very short.)

(c) (12 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-Complete. Remember each clause must contain *exactly* 4 literals.

(d) (13 points)

Problem: Hitting-Set

Instance: A collection $C = \{S_1, \ldots, S_m\}$ of non-empty subsets of a set S, and a positive integer k.
Question: Does S contain a hitting set for C of size $\leq k$, that is a subset $S' \subseteq S$ with $\lvert S' \rvert \leq k$ and such that for all i, $S' \cap S_i \neq \emptyset$.

Prove that Hitting-Set is NP-Complete. [Hint: to prove NP-hardness, reduce from Vertex-Cover.]