Introduction to Analysis of Algorithms CS4820/5820 Fall 2021	Prelim 2

Due to last-minute emergencies, it is possible that some students in the course will be taking this exam slightly late. In order to ensure that all students are provided with equitable preparation for this exam, we ask you to sign the following agreement *not to discuss the contents of this exam in the presence of anyone who hasn't taken the exam until Saturday, November 20, 2021*:

AGREEMENT: I, _____, agree not to share verbal, written, or digital information about the following exam with anyone outside of (1) the instructor, (2) the teaching assistants for the course, and (3) students in the current semester of CS4820/5820 who I know have already taken this exam before Saturday, November 20, 2021.

SIGNATURE: _____

Please clearly write your name and NetID in the box below. Also, write your NetID on the top of **each page** of the exam.

NAME: _____ NETID: _____

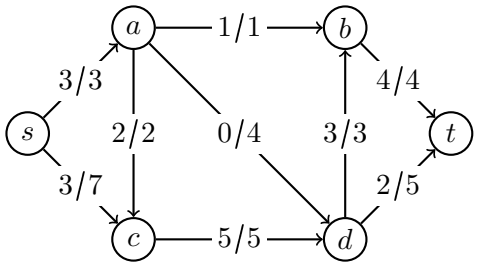
This prelim consists of 5 questions on a total of 8 pages (4 *double-sided* sheets of paper), worth a total of 50 points. Please provide the answer at the space right under the question. You may also use the back of this page or the last page for additional space, but please mark on the question page itself if you are using this as part of an answer.

This exam is closed-book and closed-notes. When you are asked to design and analyze an algorithm, you must present a full description of the algorithm and an analysis of its running time. The algorithm's running time is required to be polynomial in the input size. **You may reduce problems to ones presented in textbooks, lectures, or homework**, in which case you need not explain the details of the algorithm, and can simply state its running time. However, you still must provide the full protocol to reduce to the existing problem, as well as a correct running time analysis of the combined process of reducing the new problem to the existing one and executing the algorithm for the reduced problem. On this prelim, to save time, **please only prove what we ask for in each problem**.

Important: Follow the directions to each question carefully, **placing your answers in the designated spaces (otherwise, the autograder may ignore your answer).**

This page can be used as extra space. Please mark on the question page itself if you are using this page for part of an answer.

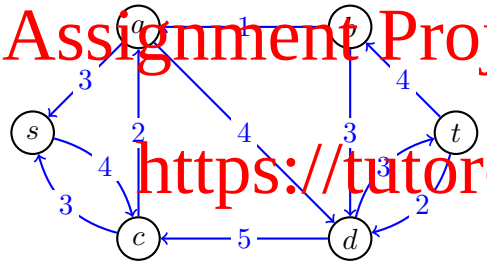
1. Consider the flow network, at an intermediary stage of the Ford-Fulkerson algorithm. Let f denote the current flow in the network. Each edge e has a label in the form "f_e/c_e", where f_e is value that f assigns to e, and c_e is the edge capacity.



(a) *(1 point)* Give the capacity of the s-t cut (A, B) where $A = \{s, a, b, c\}, B = \{d, t\}$.

$$c(A, B) = \quad 13$$

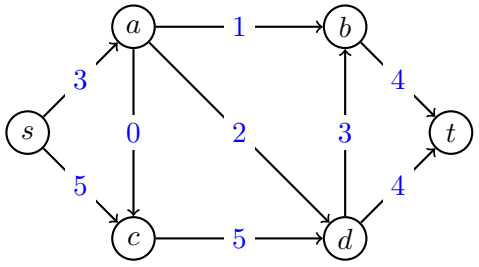
(b) *(2 points)* Draw the residual graph, G_f, below, and use it to identify an augmenting path, P (i.e., an s-t path in G_f) and its bottleneck capacity, δ.



$P = \quad s \rightarrow \quad c \rightarrow a \rightarrow d \quad \rightarrow t$

$\delta = \quad 2$

(c) *(2 points)* Let f' be the flow that results from augmenting δ units of flow onto f along P. Draw f' below. Do not label the edge capacities, only write the value f' assigns to each edge. Also, give the value of f'.



$v(f') = \quad 8$

(d) *(2 points)* Identify a minimum s-t cut (A^*, B^*) in G and its capacity.

$A^* = \quad \{s, \quad c \quad\}, \qquad B^* = \quad \{t, \quad a, b, d \quad\}, \qquad c(A^*, B^*) = \quad 8$

(e) *(1 point)* Fill in the the appropriate circle below to indicate whether the following statement is true or false. No explanation is required.

○ **True** / ○ **False** : f' is a maximum flow in G.

Solution: True. f' is a maximum flow in G by the Max-Flow Min-Cut theorem, since it has value equal to the capacity of cut (A, B).

2. For each of the following statements, indicate whether it is true or false by filling in the appropriate circle. You do **not** need to explain your choices.

   (a) *(2 points)* Consider a directed graph $G = (V, E)$ with source s and sink t, and integer capacity $c_e > 0$ on each edge $e \in E$. Let f be an s-t flow in G, and let (A, B) be an arbitrary s-t cut.

   ○ **True** / ○ **False** : The value of f equals the net flow across (A, B):

   $$v(f) = \sum_{\substack{(v,w)\in E \\ v\in A, w\in B}} f(v,w) - \sum_{\substack{(v,w)\in E \\ v\in B, w\in A}} f(v,w).$$

   Solution: True; we proved this in lecture using the fact that flow conservation must hold at every node except s and t.

   (b) *(2 points)* Suppose that $X \in \mathsf{NP}$ and that Y is a decision problem of unknown complexity. Suppose we prove that $Y \in \mathsf{NP}$, and that we find a polynomial-time reduction $X \leq_P Y$.

   ○ **True** / ○ **False** : Y is NP-complete.

   Solution: False. It's possible that both $X, Y \in \mathsf{P}$ (meaning Y is not NP-hard if P≠NP). It's also possible that Y is not a decision problem, meaning $Y \notin \mathsf{NP}$.

   (c) *(2 points)* Let UniqueStableMatching denote the problem of deciding if a stable matching instance has a unique stable matching. (You showed on Homework 1 how to solve this problem by running the Gale-Shapley algorithm twice.)

   ○ **True** / ○ **False** : Assuming $\mathsf{P} \neq \mathsf{NP}$, there is no polynomial-time reduction from UniqueStableMatching to SAT.

   Solution: False; UniqueStableMatching is in P and thus it is also in NP. SAT is NP-complete, and thus any problem in NP can be reduced to it in polynomial time.

3. Given a flow network $G = (V, E, s, t, c)$, and a maximum s-t flow f, the UNIQUEMINCUT problem asks whether G has a unique minimum-capacity s-t cut.

   (a) *(2 points)* Select the correct answer: UNIQUEMINCUT is

   ○ **solvable in polynomial-time** / ○ **NP-complete**.

   (b) *(6 points)* Briefly give the reasoning why the answer you circled is correct.

   - If you believe UNIQUEMINCUT is solvable in polynomial time, describe a polynomial time algorithm that would decide it. You do **not** need to give the running time or prove that your algorithm is correct.
   - If you believe UNIQUEMINCUT is NP-complete, describe a reduction from an NP-complete problem to UNIQUEMINCUT. You do **not** need to prove that the UNIQUEMINCUT is in NP, nor prove that your reduction is correct.

Solution: UNIQUEMINCUT $\in$ P. Consider the following algorithm.

- Construct the residual graph G_f.
- Perform BFS on G_f from s to compute a min-cut (A_1, B_1) with A_1 = {nodes reachable from s}, and $B_1 = V \setminus A_1$.
- Perform BFS on the reversed residual graph G_f^R from t to compute a min-cut (A_2, B_2) with B_2 = {nodes reachable from t}, and $A_2 = V \setminus B_2$.
- If $A_1 = A_2$, output "yes," otherwise output "no."

4. *(16 points)* A boolean formula $\phi(x_1, \cdots, x_n)$ is called *doubly satisfiable* if there are two distinct truth assignments to its variables $x_1, \cdots, x_n$ for which ϕ is true.

   The decision problem DoubleSat takes as input a boolean formula ϕ in conjunctive normal form[1] and outputs whether ϕ is doubly satisfiable. For example, on input

   $$\phi(x_1, x_2, x_3) = (x_1 \vee x_2) \ \wedge \ (\overline{x_2} \vee \overline{x_3}) \ \wedge \ (\overline{x_1} \vee x_3),$$

   DoubleSat would return "yes", since we can make ϕ true by setting $x_1 = T$, $x_2 = F$, $x_3 = T$, and also by setting $x_1 = F$, $x_2 = T$, $x_3 = F$.

   Prove that DoubleSat is NP-complete. *Include all steps of an NP-completeness proof.*

Solution: First, we argue that DoubleSat $\in$ NP. Let a certificate consist of the two truth assignments $\mathbf{x}$ and $\mathbf{y}$. This certificate has length $O(n)$, where n is the number of variables in ϕ, which is polynomial in the input length. The verifier should first check that $\mathbf{x}$ and $\mathbf{y}$ are distinct via an $O(n)$ element-wise comparison. Next, it must verify that both $\mathbf{x}$ and $\mathbf{y}$ are satisfying truth assignments for ϕ. This can be done for each assignment by iterating first over the clauses of ϕ and then over the literals in each clause, checking that the assignment sets at least one literal per clause to true. This requires $O(mn)$ time, where m is the number of clauses, for an overall polynomial runtime of the verifier.

Next, we argue that DoubleSat is NP-Hard via a reduction from Sat. Let ϕ be a boolean formula input for Sat. We transform ϕ into a new boolean formula ϕ_2 for DoubleSat as follows:

- Let x_{n+1}, x_{n+2} be variables that do not appear in ϕ.

- Take $\phi_2 = \phi \wedge (x_{n+1} \vee x_{n+2})$.

ϕ_2 has length $O(mn)$, so can be constructed in $O(mn)$ time. We argue that ϕ is satisfiable $\iff$ ϕ_2 is doubly satisfiable.

For the forward direction, suppose $\mathbf{x}$ is a satisfying assignment of ϕ. Then, $\mathbf{x} \cup \{x_{n+1} = T, x_{n+2} = F\}$ and $\mathbf{x} \cup \{x_{n+1} = F, x_{n+2} = T\}$ are both satisfying assignments of ϕ_2, as the variables assignments in $\mathbf{x}$ ensure that the first m clauses in ϕ_2 are true, and both possible assignments of x_{n+1}, x_{n+2} ensure the last clause in ϕ_2 is true.

For the reverse direction, suppose that ϕ_2 is doubly satisfiable. Notably, it has some satisfying truth assignment $\mathbf{x}$. The first m clauses in ϕ_2, that is ϕ, depends only on the first n variables, $x_1 \cdots x_n$ in $\mathbf{x}$. But then, these assignments in $\mathbf{x}$ give a satisfying assignment of ϕ.

[1] In other words, the input is the same as the input to SAT.

5. *(12 points)* A shipping company is deciding whether to do business in a country. The company has information about potential shipping opportunities organized as a graph $G = (V, E)$. The vertex set V is a list of potential ports. For each port $v \in V$, the company estimates a cost $b_v > 0$ to build a terminal.

   Each edge $e = (u, v) \in E$ is labeled with an integer $c_e > 0$, representing an estimated amount of cargo that can be shipped between u and v. The company can handle all of this cargo if and only if it has built terminals at **both** ports u and v. Each unit of cargo that they ship earns the company a profit of p.

   The company will do business in the country if there is a set of ports $S \subseteq V$ for which the cost to build terminals at these ports is less than the profits from handling all traffic between them. Give an algorithm that decides whether there is such a set S (so your algorithm should return either "yes" or "no"). The runtime of your algorithm should be polynomial in $n = |V|$, $m = |E|$, and $B = \sum_{v \in V} b_v$.

   You must fully explain how your algorithm works, and analyze the running time, but you do ***not*** *have to provide a proof of correctness.*

Solution:

Solution 1: Reduction to Project Selection

This is a project selection problem, where the positive-profit projects are the edges of G (with profit pc_e for edge e) and the negative-profit projects are the nodes of G (with profit $-b_v$ for node v); doing project $e = (u, v)$ requires us to also do projects u and v. Find the project selection A of maximum profit, and answer yes if and only if the profit of A is greater than 0.

(The set S of ports where the company should build terminals is the set of nodes for which the corresponding project is in A.)

Solution 2: Reduction to Minimum s-t Cut

We can solve this problem by reducing it to a minimum s-t cut problem.

Create a flow graph $G' = (V', E')$ where:

- $V' = \{s, t\} \cup \{n_v : v \in V\} \cup \{n_e : e \in E\}$, i.e., in addition to a source and sink, there is node n_v for every node v in G and a node n_e for every edge in G;
- E' has three types of edges:
  - an edge from s to n_e for every $e \in E$ with capacity equal to pc_e;
  - edges from n_e to n_u and n_v for every edge $e = (u, v)$ in E with capacity ∞;
  - an edge from n_v to t for every $v \in V$ with capacity b_v.

Let $P = \sum_{e \in E} pc_e$. Given a minimum s-t cut (A, B) in G', the answer to the company's question is yes if and only if $c(A, B) < P$.

Solution 3: Reduction to Project Selection/Min Cut a la Prelim Practice Question 7

It is also possible to only consider the ports as potential projects. Each v has a potential total benefit of $\sum_{e \text{ incident to } v} pc_e$, and a cost of b_v. For an edge $e = (u, v)$, if we only do v and not u, we lose the benefit pc_e, and similarly if we only do u and not v. So we create directed edges (u, v) and (v, u) both with a cost of pc_e which is incurred if we only do one of the two. Using the ideas from the practice prelim, we can turn this into a minimum cut problem by adding a source s and sink t, and for each $v \in V$ adding an edge (s, v) of capacity $\sum_{e \text{ incident to } v} pc_e$, and an edge (v, t) of capacity b_v. Consider an s-t cut (A, B) in the resulting graph. Observe that for an edge $e = (u, v)$ either both u, v are in A (and we cut edges $(u, t), (v, t)$ with capacity $b_u + b_v$), both are in B (and we cut edges $(s, u), (s, v)$ whose capacities include pc_e twice) or one of them is in A, say $u \in A, v \in B$ (in which case we cut (u, t) with capacity b_u, and $(s, v), (u, v)$ whose capacities include pc_e twice). So if we build terminals in the locations whose corresponding nodes are in A, our total profit is $2\sum_{e \in E} pc_e - c(A, B)$. We should answer yes if the minimum cut capacity is less than $2\sum_{e \in E} pc(e)$.

Running time:

The running time is the same for the first two ways to solve the problem, because project selection is solved by reducing it to minimum s-t cut.

Let $|V| = n, |E| = m$. The s-t flow graph G' has $N = m + n + 2$ nodes and $M = m + 2m + n$. Finding the maximum s-t flow in G' using the Ford-Fulkerson algorithm takes time $O(MC)$ where $M = 3m + n$ and C is the sum of the edge capacities into t, i.e., $C = \sum_v b_v$. Once we have the (value of the) maximum flow, we also know the capacity of the minimum s-t cut, and we compare it to P in $O(1)$ time to give the output. So the total running time is $O(m + n) \sum_v b_v)$.

The third approach has a flow graph of $N = n + 2$ nodes and $M = 2n$ edges. Constructing it takes $O(n + m)$ time because we need to compute $\sum_{e \text{ incident to } v} pc_e$ for every v. Finding the maximum s-t flow using the Ford-Fulkerson algorithm takes time $O(MC)$ where $M = 2n$ and C is the sum of the edge capacities into t, i.e., $C = \sum_v b_v$.

Bonus question: What did you bring to the prelim as a good luck charm?

The remainder of this page can be used as extra space. Please mark on the question page itself if you are using this page for part of an answer.