Princeton Univ Fall '21	COS 521:Advanced Algorithms
Homework 1	
Out: *Sep 6*	Due: *Sep 20*

Instructions:

- Upload your solutions (to the non-extra-credit) to each problem as a **separate PDF** file (one PDF per problem) to codePost. Please make sure you are uploading the correct PDF! Please anonymize your submission (i.e., do not list your name in the PDF), but if you forget, it's OK.
- If you choose to do extra credit, upload your solution to the extra credits as a single separate PDF file to codePost. Please again anonymize your submission.
- You may collaborate with any classmates, textbooks, the Internet, etc. Please upload a brief "collaboration statement" listing any collaborators as a separate PDF on codePost (if you forget, it's OK). But always **write up your solutions individually**.
- For each problem, you should have a solid writeup that clearly states key, concrete lemmas towards your full solution (and then you should prove those lemmas). A reader should be able to read any definitions, plus your lemma statements, and quickly conclude from these that your outline is correct. This is the most important part of your writeup, and the precise statements of your lemmas should tie together in a correct logical chain.
- A reader should also be able to verify the proof of each lemma statement in your outline, although it is OK to skip proofs that are clear without justification (and it is OK to skip tedious calculations). Expect to learn throughout the semester what typically counts as 'clear'.
- You can use the style of Lecture Notes and Staff Solutions as a guide. These tend to break down proofs into roughly the same style of concrete lemmas you are expected to do on homeworks. However, they also tend to prove each lemma in slightly more detail than is necessary on PSets (for example, they give proofs of some small claims/observations that would be OK to state without proof on a PSet).
- Each problem is worth twenty points (even those with multiple subparts), unless explicitly stated otherwise.

Problems:

§1 Prove that (the natural variant of) Karger's algorithm does not work for finding the minimum s-t cut in unweighted, undirected graphs. Specifically, design an unweighted, undirected graph G (with no parallel edges), with two nodes s, t, such that repeatedly contracting a random edge **that does not contract s and t to the same supernode** outputs a minimum s-t cut with probability $2^{-\Omega(n)}$.[1]

[1] To be clear: the algorithm is guaranteed to output *an* s-t cut, it just might not be the minimum.

Hint: try to prove that the algorithm works, and see which step fails. Use this to guide your example.

§2 A cut is said to be a *B-approximate min cut* if the number of edges in it is at most B times that of the minimum cut. Show that all undirected graphs have at most $(2n)^{2B}$ cuts that are B-approximate.

Hint: Run Karger's algorithm until it has $2B$ supernodes. What is the chance that a particular B-approximate cut is still available? How many possible cuts does this collapsed graph have?

§3 Recall the max-flow problem from undergraduate algorithms: for a directed graph $G(V, E)$ with non-negative capacities c_e for every $e \in E$ and two special vertices s (source, with no incoming edges) and t (sink, with no outgoing edges), a *flow* in G is an assignment $f : E \to \mathbb{R}_{\geq 0}$ such that $f(e) \leq c_e$ for every edge and for every vertex $v \in V \setminus \{s, t\}$, the total incoming flow $\sum_{(u,v)\in E} f((u, v))$ equals the total outgoing flow $\sum_{(v,w)\in E} f((v, w))$. The task is to find a maximum flow f i.e., a flow f such that $\sum_{(s,u)\in E} f((s, u))$ is maximized.

(a) Show that the following LP is a valid formulation for computing the value of the maximum flow in G. There is a variable $f((u, v))$ for all $(u, v) \in E$. (Hint: below, the inequality between the flows is not a typo. You should show that it is w.l.o.g. to replace the equality with inequality, as this will make it easier to reason about later parts.)

$$\begin{aligned} \max & \sum_u f((u, t)) \\ & \forall e = (u, v) \in E,\ f((u, v)) \leq c_e \\ & \forall v \notin \{s, t\},\ \sum_{(u,v)\in E} f((u, v)) \geq \sum_{(v,w)\in E} f((v, w)) \\ & \forall e \in E,\ f(e) \geq 0 \end{aligned} \tag{1}$$

(b) Write the dual for the LP (1). Show that this dual LP computes the minimum *fractional* s-t cut in G (a cut that separates s and t in G and minimizes the sum of the capacities c_e of the edges going across it. You will know what a fractional s-t cut is once you take the dual: every node isn't entirely on the s side or the t side, but rather partially on each). Use strong LP duality to conclude the *fractional* max-flow min-cut theorem. That is, if the max-flow is C, there exists a fractional s-t cut of value C, and no fractional s-t cut of value $< C$.

(c) Devise a rounding scheme that takes as input a fractional min-cut of value C and outputs a true (deterministic) min-cut of value C. (Hint: there is a simple rounding scheme that works, but it is not a rounding scheme we have already seen in class. You might want to first construct a randomized min-cut.) Conclude the max-flow min-cut theorem.

§4 In class we designed a 3/4-approximation for MAX-2SAT using LP rounding. The MAX-SAT problem is similar except for the fact that the clauses can contain any

number of literals. Formally, the input consists of n boolean variables $x_1, x_2, \ldots, x_n$ (each may be either 0 (false) or 1 (true)), m clauses $C_1, C_2, \ldots, C_m$ (each of which consists of disjunction (an "or") of some number variables or their negations) and a non-negative weight w_i for each clause. The objective is to find an assignment of 1 or 0 to x_is that maximize the total weight of satisfied clauses. As we saw in the class, a clause is satisfied if one of its non-negated variable is set to 1, or one of the negated variable is set to 0. You can assume that no literal is repeated in a clause and at most one of x_i or $\neg x_i$ appears in any clause.

(a) Generalize the LP relaxation for MAX-2SAT seen in the class to obtain a LP relaxation of the MAX-SAT problem.

(b) Use the standard randomized rounding algorithm (the same one we used in class for MAX-2SAT) on the LP-relaxation you designed in part (1) to give a $(1-1/e)$ approximation algorithm for MAX-SAT. Recall that clauses can be of any length. (Hint: there is a clean way to resolve "the math" without excessive calculations).

(c) A naive algorithm for MAX-SAT problem is to set each variable to true with probability 1/2 (without writing any LP). It is easy to see that this *unbiased randomized* algorithm of MAX-SAT achieves 1/2-approximation in expectation. Show the algorithm that returns the best of two solutions given by the randomized rounding of the LP and the simple unbiased randomized algorithm is a 3/4-approximation algorithm of MAX-SAT. (Hint: it may help to realize that in fact *randomly* selecting one of these two algorithms to run also gives a 3/4-approximation in expectation).

(d) Using the previous part (and in particular, the hint) for intuition, design a direct rounding scheme of your LP relaxation to get a 3/4-approximation (that is, design a function $f(\cdot)$ which assigns a literal x_i to be true independently with probability $f(z)$ when the corresponding variable z_i in your LP relaxation is equal to z). (Hint: here, it may get messy to fully resolve the calculations. You will get full credit if you state the correct rounding scheme and clearly state the necessary inequalities for the proof. You should also attempt to show that the inequalities hold for your own benefit, but not for full credit).

§5 (Firehouse location) Suppose we model a city as an m-point finite metric space with $d(x, y)$ denoting the distance between points x, y. These $\binom{m}{2}$ distances (which satisfy triangle inequality) are given as part of the input. The city has n houses located at points $v_1, v_2, \ldots, v_n$ in this metric space. The city wishes to build k firehouses and asks you to help find the best locations $c_1, c_2, \ldots, c_k$ for them, which can be located at any of the m points in the city. The *happiness* of a town resident with the final locations depends upon his distance from the closest firehouse. So you decide to minimize the cost function $\sum_{i=1}^{n} d(v_i, u_i)$ where $u_i \in \{c_1, c_2, \ldots, c_k\}$ is the firehouse closest to v_i. Describe an LP-rounding-based algorithm that runs in poly(m) time and solves this problem approximately. If OPT is the optimum cost of a solution with k firehouses, your solution is allowed to use $O(k \log n)$ firehouses and have cost at most OPT.[2]

[2]The term for an approximation guarantee like this is *resource augmentation* — the solution is as good as the optimum, but it requires additional firehouses.

Specifically, you should design an algorithm which runs in polynomial time, and uses $O(k \log n)$ firehouses in expectation, and also has cost at most OPT in expectation.[3]

Extra Credit:

§1 (extra credit) In a *combinatorial auction* there are n bidders and m items. Bidder i has a monotone valuation function $v_i(\cdot)$ where $v_i(S)$ denotes their value for set S of items (and $v_i(S \cup T) \geq v_i(S)$ for all S, T). A *Walrasian Equilibrium* is a price for each item $\vec{p}$ such that:

- Each buyer i selects to purchase a set $B_i \in \arg\max_S \{v_i(S) - \sum_{j \in S} p_j\}$.
- The sets B_i are disjoint, and $\cup_i B_i = [m]$.

Prove that a Walrasian equilibrium exists for $v_1, \ldots, v_n$ if and only if the optimum of the LP relaxation below (called the *configuration LP*) is achieved at an integral point (i.e. where each $x_{i,S} \in \{0, 1\}$). Hint: use strong duality!

$$\max \sum_i \sum_S v_i(S) \cdot x_{i,S}$$
$$\forall i, \sum_S x_{i,S} = 1$$
$$\forall j, \sum_{S \ni j} \sum_i x_{i,S} \leq 1$$

Also, come up with an example of two valuation functions v_1, v_2 over two items where a Walrasian equilibrium doesn't exist.

[3] You may want to briefly think about how to modify your solution to run in expected polynomial time and use $O(k \log n)$ firehouses with probability one, or how to run in expected polynomial time and guarantee a solution with cost $(1 + \varepsilon)$OPT with probability one (or both). But you do not need to write this for full credit.