Princeton Univ Fall '23 COS 521:Advanced Algorithms

Homework 1

Out: *Sep 11* Due: *Sep 24*

Instructions:

- Upload your non-extra solutions to Gradescope in a single PDF file, and mark your solution to each problem. 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 Gradescope. 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 Gradescope (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 In class, we showed that for a *given* min-cut $(S, \overline{S})$, the probability that Karger's algorithm outputs $(S, \overline{S})$ is at least $\Omega(n^{-2})$. For sufficiently large n, give a graph G (possibly with multi-edges) such that the probability that Karger's algorithm outputs *any* min-cut is $\Theta(n^{-2})$.

§2 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]

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

§3 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?

§4 In class we saw that when hashing m items into a hash table of size $O(m^2)$, the expected number of collisions was < 1. In particular, this meant we could easily find a "perfect" hash function of the table that has no collisions.

Consider the following alternative scheme: build two tables, each of size $O(m^{1.5})$ and choose a separate hash function for each table independently. To insert an item, hash it to one bucket in each table and insert it only in the emptier bucket (tie-break lexicographically).

(a) Show that, if we're hashing m items, with probability 1/2, there will be no collisions in either table (a collision occurs when multiple distinct elements are inserted into the same bucket in the same table). You may assume access to a fully random hash function (see definition in lecture notes).

(b) Modify the above scheme to use $O(\log m)$ tables, each of size $O(m)$. Prove again that with probability 1/2, there will be no collisions in any table. Again, you may assume a fully random hash function.

§5 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.

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

(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 $(u,v) \in E$.

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

(b) Write the dual for the LP (1). Show that this dual LP computes the minimum *fractional* s-t cut in G. A fractional cut places each node v at some point y_v on the unit interval $[0,1]$, with s placed at 0 and t placed at 1. The value of the fractional cut is $\sum_{(u,v)\in E(G)} c_e \cdot \max\{0, y_v - y_u\}$ (where c_e is the weight of edge e). Observe that if instead each $y_v \in \{0,1\}$, that this is simply an s-t cut. 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.

Extra Credit:

§1 Given a undirected *unweighted* simple graph $G = (V,E)$, prove that there is a sparse *weighted* graph $G' = (V, E')$ such that the weight of every cut is close in G and G'. That is, prove that for any $\varepsilon > 0$, there is a graph G' with $O(\varepsilon^{-2} \cdot n \cdot \operatorname{poly}\log n)$ *weighted* edges such that for every cut $(S, \overline{S})$, we have

$$\mathrm{CUT}_{G'}(S,\overline{S}) = (1 \pm \varepsilon)\mathrm{CUT}_G(S,\overline{S}).$$

Hint: divide-and-conquer when min-cut is “small”, random sampling when min-cut is “large”. An upper bound on the number of B-approximate cuts might help!