CS 170 Homework 14

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, explicitly write "none".

2 Hitting Set

In the Hitting Set Problem, we are given a family of finite integer sets $\{S_1, S_2, \ldots, S_n\}$ and a budget b, and we wish to find an integer set H of size $\leq b$ which intersects every S_i, if such an H exists. In other words, we want $H \cap S_i \neq \emptyset$ for all i.

Show that the Hitting Set Problem is NP-complete. (Hint: Hitting Set generalizes one of the problems covered in Chapter 8 of the textbook).

3 Orthogonal Vectors

In the 3-SAT problem, we have n variables and m clauses, where each clause is the OR of (at most) three of these variables or their negations. The goal of the problem is to find an assignment of variables that satisfies all the clauses, or correctly declare that none exists.

In the orthogonal vectors problem, we have two sets of vectors A, B. All vectors are in $\{0,1\}^m$, and $|A| = |B| = n$. The goal of the problem is to find two vectors $a \in A, b \in B$ whose dot product is 0, or correctly declare that none exists. The brute-force solution to this problem takes $O(n^2 m)$ time: compute all $|A||B| = n^2$ dot products between two vectors in A, B, and each dot product takes $O(m)$ time.

Show that if there is a $O(n^c m)$-time algorithm for the orthogonal vectors problem for some $c \in [1, 2)$, then there is a $O(2^{cn/2} m)$-time algorithm for the 3-SAT problem. For simplicity, you may assume in 3-SAT that the number of variables must be even.

Hint: Try splitting the variables in the 3-SAT problem into two groups.

4 Local Search for Max Cut

Sometimes, local search algorithms can give good approximations to NP-hard problems. Recall that in the Max-Cut problem, we have an unweighted graph $G = (V, E)$ and we want to find a cut (S, T) that maximizes the number of edges "crossing" the cut (i.e. with one endpoint in each of S, T). Consider the following local search algorithm:

1. Start with any cut (e.g. $(S, T) = (V, \emptyset)$).
2. While there is some vertex $v \in S$ such that more edges cross $(S \setminus \{v\}, T \cup \{v\})$ than (S, T) (or some $v \in T$ such that more edges cross $(S \cup \{v\}, T \setminus \{v\})$ than (S, T)), move v to the other side of the cut.

Now, let us prove a couple of guarantees that this algorithm achieves.

(a) Give an upper bound on the number of iterations this algorithm can run for (i.e. the total number of times we move a vertex).

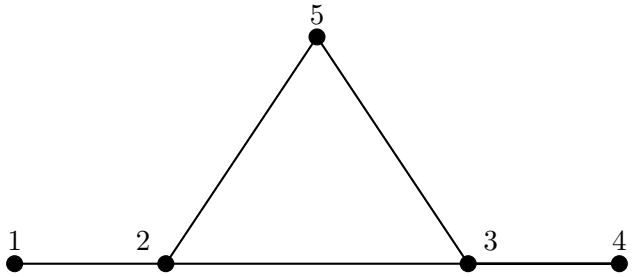
(b) Show that when this algorithm terminates, it finds a cut where at least half the edges in the graph cross the cut.

Hint: when we move v from S to T, v must have more neighbors in S than T. What does this observation suggest about the neighbors of each vertex once the algorithm terminates? Then, what can we say about the number of edges crossing the cut vs. the number of edges within each side of the cut?

5 Second Smallest Global Min-cut

Recall that a *cut* of an undirected, unweighted graph (V, E) is a partition of the vertices into two non-empty sets S and $V \backslash S$. The *weight* of such a cut is the number of edges crossing the cut, which we denote $|E(S, V \backslash S)|$. Karger's contraction algorithm from class finds a cut of minimum weight with probability $1 - p$, with runtime $O(poly(n) \cdot \log(1/p))$, where $n := |V|$ and $m := |E|$.

In this problem, we will be concerned with finding not the smallest, but rather the second smallest minimum cut. Ties are broken arbitrarily. For example, consider the graph below:



It has two cuts of weight one, given by $S = \{1\}$ or $S = \{4\}$. All other cuts have weight at least two. Since we break ties arbitrarily, we would say in this example both the smallest *and* second smallest cuts have weight equal to 1.

(a) Give a general formula for the number of cuts in a graph with n vertices and m edges. **Note:** we are asking for the number of cuts, not necessarily minimum cuts.

(b) Focus on a particular second smallest mincut S^* of G. Consider running just the first step of Karger's contraction algorithm, where we contract a uniformly random edge. Compute a tight upper bound α for the probability that the contracted edge crosses $(S^*, V \backslash S^*)$. Your answer is allowed to depend on $k := |S^*|$ or n, but not on m.

(c) Starting with n vertices, imagine doing $n - 3$ random contraction steps in sequence. How many "supervertices" are left?

(d) Suppose your answer in part (c) is c. Imagine then picking a uniformly random cut (not necessarily a mincut) on this graph of c supervertices to get a resulting cut of the original graph. Show that the probability that this resulting cut is S^* is $\Omega(1/n^2)$.

(e) Give a $O(poly(n) \cdot \log(1/p))$ time algorithm for finding the second smallest mincut in a graph with success probability at least $1 - p$. You need not calculate the precise exponent of n in the $poly(n)$ term; any polynomial dependence suffices.

Hint: $(1 + x)^T \leq e^{xT}$ *for all* $x \in \mathbb{R}$ *and* $T \geq 0$.

6 How to Gamble With Little Regret

Suppose that you are gambling at a casino. Every day you play at a slot machine, and your goal is to minimize your losses. We model this as the experts problem. Every day you must take the advice of one of n experts (i.e. play at a slot machine of their choosing). At the end of each day t, if you take advice from expert i, the advice costs you some c_i^t in $[0, 1]$. You want to minimize the regret R, defined as:

$$R = \frac{1}{T}\left(\sum_{t=1}^{T} c_{i(t)}^t - \min_{1\le i\le n} \sum_{t=1}^{T} c_i^t\right)$$

where $i(t)$ is the expert you choose on day t. Notice that in this definition, you are comparing your losses to the best expert, rather than the best overall strategy.

Your strategy will be probabilities where p_i^t denotes the probability with which you choose expert i on day t. Assume an all powerful adversary (i.e. the casino) can look at your strategy ahead of time and decide the costs associated with each expert on each day. Let C denote the set of costs for all experts and all days. Compute $\max_C(\mathbb{E}[R])$, or the maximum possible (expected) regret that the adversary can guarantee, for each of the following strategies, with justification.

(a) Any deterministic strategy, i.e. for each t, there exists some i such that $p_i^t = 1$.

(b) Always choose an expert according to some fixed probability distribution at every time step. That is, fix some $p_1, \ldots, p_n$, and for all t, set $p_i^t = p_i$.

What distribution minimizes the regret of this strategy? In other words, what is $\text{argmin}_{p_1,\ldots,p_n} \max_C(\mathbb{E}[R])$?

This analysis should conclude that a good strategy for the problem must necessarily be randomized and adaptive.

7 Variants on the Experts Problem

Consider the *realizable* experts problem: We have n experts who predict it will either rain or shine tomorrow. At least one of the experts is perfect and will always make the correct prediction. Every day we guess based on the experts if it will rain or shine tomorrow. Our goal is to make as few mistakes in our predictions as possible.

(a) Suppose we always guess the prediction of the majority of experts who have been correct so far, breaking ties arbitrarily (Note that this set is always non-empty since one expert is always correct). Show that we make at most $\log n$ mistakes using this strategy.

Hint: if you're stuck, look at the Halving Algorithm from lecture

(b) Suppose there are now k experts who are always correct instead of just one. Give a better upper bound on the number of mistakes the algorithm from the previous part makes makes.

(c) Suppose instead of one expert always being correct, we are only guaranteed that there is one expert who makes at most k mistakes. Consider the following algorithm: Let m be the least mistakes made by any expert so far. Use the prediction of the majority of experts who have made at most m mistakes so far.

Give an upper bound on the number of mistakes made by this algorithm.

8 Informed Predictions, Randomized!

Define the *realizable* experts problem to be as follows: there are n experts who each make predictions every day and we need to use the expert predictions to make our own daily predictions; additionally, there is guaranteed to be at least one *perfect* expert that is always correct. In class, we found that the Halving Algorithm (which is deterministic) makes at most $\log_2 n$ mistakes.

Describe a randomized algorithm that, in expectation, makes at most $\log_e n = \ln n$ mistakes. Prove that your algorithm, indeed, makes $\ln n$ mistakes in expectation.

Hint: you may use the identity that $\sum_{i=1}^{n} \frac{1}{i} \leq \ln n + 1$.