CS 787: Advanced Algorithms	Homework 3
Out: 11/10/15	**Due: 11/24/15**

Guidelines

- This homework consists of 6 problems. Only the problems marked with an asterisk (two in all) will be graded. You are not required turn in solutions to the unmarked problems but you are highly encouraged to solve and write up your solutions to all of them. We will provide solutions for all the problems.
- Some of the problems are difficult, so please get started early. Late submissions do not get any credit.
- Please typeset your solutions.
- Homework should be done in pairs. Please write your names clearly on your homework.

Problems

1. Consider adapting Karger's randomized min-cut algorithm from class to finding an s-t min-cut in a graph as follows: at every stage, we have an s-supernode and a t-supernode. Of all the edges that do not go between the s-supernode and the t-supernode, we pick a random edge and contract it; the algorithm ends when we are left with only the s- and t-supernodes; we output the remaining cut.

   (a) Show that there are graphs for which this algorithm produces an s-t min-cut with exponentially low probability.

   (b) What is the largest number of distinct s-t min-cuts that a graph can have? Give a graph that achieves this number.

2. (*) The **online bin-packing** problem is a variant of the knapsack problem. We are given an unlimited number of bins, each of size 1. We get a sequence of items one by one, each of a certain size no more than 1, and are required to place them into bins as we receive them. Our goal is to minimize the number of bins we use, subject to the constraint that no bin to should be filled to more than its capacity. In this question we will consider a simple online algorithm for this problem called **First-Fit** (FF). FF orders the bins arbitrarily, and places each item into the first bin that has enough space to hold the item.

   Recall that the competitive ratio of an online algorithm for a minimization problem is the maximum over all arrival sequences of the cost of the algorithm (in this case, the number of bins used by the algorithm) to the cost of the hindsight OPT (in this case, the minimum number of bins necessary to pack the items).

   (a) Give an instance of bin-packing for which FF does not obtain the optimal packing.

   (b) Prove that FF has competitive ratio no more than 2, that is, on every instance, it uses no more than twice as many bins as necessary.

   (c) **Extra credit:** Can you prove a better bound on the competitive ratio of FF?

3. Consider an undirected graph $G = (V, E)$, where each vertex v has a list $S(v)$ of allowed colors. A **list-coloring** χ of G assigns each vertex $v \in V$ a color from its list $S(v)$. A proper list-coloring is one that ensures that all edges are bichromatic, that is, their end points are assigned different colors.

   Suppose each vertex has a list of size k for some parameter k. Moreover, suppose that for each $v \in V$ and $c \in S(v)$, there are at most $k/10$ neighbors u of v that contain c in their color sets $S(u)$. (There is no bound on the degree of the underlying graph, though.) Show that there exists a proper list-coloring of G with these parameters.

4. Call two vectors near-orthogonal if their inner product has small absolute value compared to the product of their lengths; in this problem we will show that while there are at most d orthogonal vectors in $\Re^d$, there can be exponentially more near-orthogonal vectors.

   Given a parameter $\epsilon > 0$, two vectors $\vec{x}, \vec{y} \in \Re^d$ are called ϵ-orthogonal if

   $$|\vec{x} \cdot \vec{y}| \leq \epsilon \|x\| \|y\|.$$

   Show that there exist $N = \exp(\Omega(\epsilon^2 d))$ vectors in $\{-1, 1\}^d$ that are mutually ϵ-orthogonal. (Hint: consider a random such set.)

5. Here is a variation on the deterministic Weighted-Majority algorithm, designed to make it more adaptive.

   (a) Each expert begins with weight 1 (as before).

   (b) At every step, we predict the result of a weighted-majority vote of the experts (as before).

   (c) At every step, for every expert that makes a mistake, we penalize it by dividing its weight by 2, but only if its weight was at least $1/4$ of the average weight of experts.

   Prove that in any contiguous block of steps (e.g., the 51st step through the 77th step), the number of mistakes made by the algorithm is at most $O(m + \log n)$, where m is the number of mistakes made by the best expert in that block, and n is the total number of experts.

   (Recall that the original weighted majority algorithm only gives a guarantee over a block of steps starting at step 1.)

6. (*) Consider a standard experts setting and suppose that at any time step, only a subset of the experts are available to make a prediction. Can we modify the Hedge algorithm from class to give a low regret bound in this case? More precisely, in this setting, at every time step we get to see which experts are "awake" to make a prediction. We choose one of those experts. Then we get to see the cost vector for that step. The total cost of expert i is the sum of its costs over the steps that i was awake in. Naturally, if an expert is awake for very few steps, we cannot hope to prove that our total cost over all the steps is not much larger than the expert's cost over the steps that it was awake in. So we will aim for a slightly different guarantee.

   Let T_i denote the set of steps when expert i was awake, $\text{cost}_i(\text{ALG})$ denote the expected cost of the algorithm over the steps T_i, and $\text{cost}_i(i)$ denote the cost of expert i over the steps T_i. Then, our goal is to say that $\text{cost}_i(\text{ALG}) \leq \frac{1}{1-\epsilon}\text{cost}_i(i) + O(\frac{1}{\epsilon}\log n)$, and this should hold for every expert i.

   Consider the following variant of the Hedge algorithm for this problem:

   (i) Initialize $w_{i,0} = 1$ for all i.

   (ii) At every step t, let $p_{i,t} = w_{i,t}/W_t$ be the probability of picking an awake expert i, where W_t is the sum of the weights of all the experts awake at that step (not the total weight of all the experts).

   (iii) Let $c_{i,t}$ denote the cost to expert i at step t. Update weights for awake experts as follows:

   $$R_{i,t} = \frac{1}{1+\epsilon}\left(\sum_j p_{j,t}c_{j,t}\right) - c_{i,t}$$

   $$w_{i,t+1} = w_{i,t}(1+\epsilon)^{R_{i,t}}$$

   (a) Prove using induction that the total weight of all experts at any step is at most n. (In particular, although individual weights can go up or down, the total weight never goes up).

   (b) Express the weight of expert i at time T in the form of the total expected cost of the algorithm and the total cost of the expert. Use part (a) to prove that $\text{cost}_i(\text{ALG}) \leq (1+\epsilon)\text{cost}_i(i) + O(\frac{1}{\epsilon}\log n)$.