Homework 1 (due Friday, January 25, 11:59pm)

**Directions:** Do any five of the following six problems. (If you submit all six, we will only grade 5, and we will pick which five.)

The problems have been carefully chosen for their pedagogical value and hence might be similar or identical to those given out in past offerings of this course at UW, or similar courses at other schools. Using any pre-existing solutions from these sources, or any source not explicitly linked to or discussed on our course web page constitutes a violation of the academic integrity expected of you and is strictly prohibited.

Unless specifically forbidden, you are allowed to collaborate with fellow students taking the class in solving problem sets, however you must always write the solutions up on your own. If you do collaborate on solving the problems, you must acknowledge for each problem the people you discussed that problem with (other than the TA and instructor).

When you present an algorithm, you must (briefly) prove that it achieves the guarantees you claim and prove that it runs in polynomial time (unless you are explicitly asked to prove something more specific.)

Concise, clear and precise solutions are appreciated (and expected).

You will be submitting your homework on [http://gradescope.com](http://gradescope.com) You can find instructions on the course webpage.

One thing to keep in mind is that you need to assign PDF pages to each question when submitting your solutions on Gradescope. Note that one can assign multiple pages to a single question and a single page to multiple questions (if one solves 2 questions on the same page, which we would strongly prefer that you do not do)

1. **Online list update – lower bounds (10 points)**

Suppose you have a linked list with $n$ items. Accessing an item at position $i$ in the list costs $i$, but immediately after the access, that item can be moved to any of the first $i$ positions for free. Any pair of adjacent items can also be swapped at a cost of 1. In class we showed that the Move-To-Front algorithm has competitive ratio 2 for this problem.

Prove that every deterministic algorithm has competitive ratio at least $2 - \frac{2}{n+1}$.

**Hint:** Fix any deterministic algorithm, and consider a "brutal" request sequence $\sigma$ in which every single request is to the currently last element in the list. Show that there is an offline algorithm that on a sequence of length $T$ incurs a cost of at most $\frac{1}{2}T(n + 1) + C$, where $C$ is a constant independent of $T$. (It can depend on $n$.) The simplest way to do this is to use the best static rearrangement of the list (given the request sequence $\sigma$).

2. **Online list update – some bad algorithms (10 points)**

Same setup as previous problem.

(a) [5 Points] Show that the online algorithm that after each request moves the requested item 1 position closer to the front (if possible) has competitive ratio $\Omega(n)$.

(b) [5 Points] Consider an online algorithm that maintains a counter for each item, so if the items stored in the list are named $a_1, \ldots, a_n$, then $c_i(t)$ is the number of requests there have been to $a_i$ up to time $t$. (Of course $c_i(0) = 0$ for all $i$.) Whenever an item is accessed, reorganize the list so that items are kept in order of decreasing value of $c_i(t)$. (i.e., more frequently requested items are closer to the front.) Ties
can be broken arbitrarily. Show that this algorithm has competitive ratio \( \Omega(n) \). **Hint:** Consider a request sequence in which there are first \( 2n \) accesses for item \( a_1 \), then \( 2n - 1 \) accesses for item \( a_2 \), then \( 2n - 3 \) accesses for \( a_3 \) and so on down to \( n + 1 \) accesses for \( a_n \) and then repeat.

3. **Matching (10 points)**

Read this description of the bipartite maximum weight matching algorithm discussed in class.

Fill in the details (pseudocode and proof of running time) for how to implement the algorithm on non-dense, graphs using a priority queue so as to obtain a running time of \( O(mn \log n) \), and with linked lists (and an amortized analysis) to get \( O(mn) \). For the priority queue, please specify your implementation. [This is discussed in the final paragraph before "Additional comments".]

Here the graph is \( n \) by \( n \), and there are \( m \) edges of value 1. All non-edges have value 0.

4. **Generalized matching (10 points)**

Consider the following generalized matching problem. There are two sets: \( B \), a set of buyers, and \( I \), a set of items being sold by a single seller, with a bipartite graph connecting them. Each buyer \( i \) has an associated budget \( B_i \), and a value \( v_{ij} \) for each item \( j \), representing what he will pay for that item. A generalized matching associates each item \( j \) with one buyer. A buyer, on the other hand, may be matched to (i.e., he can purchase) multiple items, as long as the items he is allocated respect his budget. If they go over his budget, he will only pay up to his budget. Thus, if \( S_i \) is the set of items sold to buyer \( i \), then the profit obtained from \( i \) is \( \min(\sum_{j \in S_i} v_{ij}, B_i) \).

A maximum generalized matching assigns the items to the buyers so as to maximize the profit of the seller. Give a polynomial time algorithm for this problem that is guaranteed to obtain at least half of the optimal total profit. **Hint:** Try a greedy algorithm.

5. **Deterministic online matching (10 points)**

Consider the following online and simplified version of the previous problem. There are a set of \( n \) distinct items available for sale. \( n \) buyers arrive one at a time. When buyer \( i \) arrives, you learn which items \( S_i \) he is interested in. (Some of them may already be sold.) At that point you can pick one of the items he is interested in that is still unsold and sell it to him. (Obviously, if all the items he is interested in have already been sold, then you cannot sell anything to him.) This is the only chance you have to sell anything to this buyer. The goal is to come up with an online selling algorithm that maximizes the number of items sold.

(a) [5 Points] Give a 2-competitive deterministic algorithm for this problem. (In a profit problem, like this one, a competitive ratio of 2, means the that optimal offline profit is at most twice that of the online algorithm profit.)

(b) [5 Points] Show that every deterministic algorithm for this problem has competitive ratio at least 2. (It suffices to construct a simple bad example.)

6. **Randomized online matching (10 points)**

Now consider the randomized algorithm that, upon the arrival of a buyer, sells him a uniformly random unsold item among \( S_i \) (so if \( k \) of the items in \( S_i \) are still unsold, a random one of these \( k \) is selected and sold to him). Show that the competitive ratio of this algorithm is \( 2 - o(1) \) (Here the \( o(1) \) refers to the fact that \( n \) is tending to infinity.)

Do this by analyzing the algorithm on the following input. There are \( 2n \) buyers and \( 2n \) items. Buyer \( i \) is equally happy to get any of items \( i, n + 1, n + 2, \ldots, 2n \). (Thus, buyer \( i \) is interested in exactly \( n + 1 \) items.) On the
other hand, buyer $n + i$ is interested only in item $n + i$.

(For your analysis, it may be helpful to use the fact that $\sum_{i=1}^{n} 1/i = \ln n + o(1).$)