Further clarifications and Hints on Problem Set 1

1. You are to prove a lower bound on the competitive ratio for any deterministic algorithm. What this means is that you need to show the following:

   For any online algorithm $A$, there is sequence of $T$ item requests $\sigma_A$, such that

   \[ A(\sigma_A) \geq \left( 2 - \frac{2}{n+1} \right) \text{OPT}(\sigma_A) - c, \]  

   (0.1)

   where $c \to 0$ as $T \to \infty$. In expression (0.1), $A(\sigma_A)$ represents the total cost incurred by algorithm $A$ on request sequence $\sigma_A$ and $\text{OPT}(\sigma_A)$ represents the total cost incurred by the optimal offline algorithm on request sequence $\sigma_A$.

   Note that $\sigma_A$ will depend on the particular online algorithm $A$ and the problem suggests that $\sigma_A$ should be the following: request the item that is at the end of $A$’s list at each time. With this sequence, it is immediate that on a sequence of $T$ requests, $A(\sigma_A) \geq nT$.

   So it would suffice to argue that for any sequence $\sigma_A$, there is a way to service that sequence at cost at most

   \[ \frac{nT}{2 - \frac{2}{n+1}} + C, \]

   where $C$ does not depend on $T$.

   You may want to use the fact that

   \[ \frac{n}{(n+1)} = \left( 1 - \frac{1}{n+1} \right). \]

2. Here you are asked to prove much larger lower bounds than 2 for some specific algorithms.

   (a) Let’s call the specific online algorithm that after each request moves the requested item 1 position closer to the front of the list (if possible) “SW” (for swap forward). You are asked to show that the competitive ratio of SW is at least $cn$ for some constant $c$. For this purpose, you need to come up with a single request sequence $\sigma$ (that can be made arbitrarily long) such that

   \[ \text{SW}(\sigma) \geq cn \cdot \text{OPT}(\sigma). \]

   (b) Similar question for the algorithm “FC” (frequency count) that always keeps the elements in order sorted by how many times that element has been requested in the past (with most frequent at the front and least frequent at the back).

   There was a typo on the homework – was supposed to say $2n - 2$ requests for $a_3$. 

3. The final paragraph (before “Additional comments”) of
describes the implementation. You are simply asked to fill in all the details for this
paragraph. Thus, as suggested, the deliverables are:

(a) Pseudocode for the auction algorithm using a priority queue with runtime $O(mn\log(n))$.
(b) Proof of the runtime of (a)
(c) Pseudocode/explanation for using a linked list to obtain a runtime of $O(mn)$ (no
need to rewrite the entire algorithm).
(d) Amortized analysis of the runtime of (c). The paragraph in the blog post indicates
exactly where the amortization comes in.

Note that you need to specify what implementation of a priority queue you use, but
you don’t need to detail how to implement a standard priority queue.

4. Suppose that some algorithm assigns item $j$ to buyer $M(j)$, and let $S_i$ denote the set
of items $j$ assigned to buyer $i$, i.e. the set of $j$ such that $M(j) = i$. Then the profit of
the algorithm is

$$\sum_i \min(B_i, \sum_{j \in S_i} v_{ij}).$$

(Notice that without loss of generality, all items can assigned to some buyer, though
they may not increase the profit extracted from that buyer because his budget may be
exceeded.)

Here is an example:
The hint suggests a greedy algorithm, specifically the following: Suppose that at some point you’ve already assigned some subset of the items, say ˜S is the subset that have already been assigned, where ˜S_i is the subset of items that have been assigned to buyer i so far. (Initially ˜S = ˜S_i = ∅ for all i.)

Then some part of each buyer’s budget has been used (initially none). Specifically buyer i’s leftover budget ˜B_i is

\[ ˜B_i = \max(0, B_i - \sum_{j \in ˜S_i} v_{ij}). \]

If we were to then allocate unsold item ℓ to buyer k, say, then our additional profit would be

\[ \min(v_{kℓ}, ˜B_k). \]

The greedy algorithm is to allocate next an item that gives the largest additional profit.

Proving that this gives a factor of two can be done by induction on the number of items allocated, but you will want to strengthen your induction hypothesis. Specifically, you might want to show that the profit of your algorithm when the set of items is I, the set of buyers is B, the values are \{v_{ij}\} and the budgets are B_1, \ldots, B_n is at least half the profit of an optimal algorithm dealing with set of items I, set of buyers B, values \{v_{ij}\} (exactly the same), but with budgets B'_1, \ldots, B'_n, where B'_i \leq B_i for all i.

One little trick here that might be useful (and if you use this you will need to make a quick argument that it is correct): Suppose that your online algorithm assigns item j to bidder 1 and the offline algorithm assigns item j to bidder 2. Then the offline profit can be upper bounded by \min(v_{j1}, B_1) + \min(v_{j2}, B_2) + OPT(L) where L is the leftover problem consisting of all items except for j, and budgets B'_1 := \max(B_1 - v_{j1}, 0), B'_2 := \max(B_2 - v_{j2}, 0) and B'_i := B_i for all i \geq 3.

5. (a) In part (a), you are dealing with the same problem as in problem 4, but it is online, in that the buyers arrive one at a time, and only immediately upon arrival can you allocate an item to that buyer. Moreover, it is the special case where all v_{ij}’s are either 0 or 1, and B_i = 1 for each buyer. But again, the buyers arrive online.

Try to use an online version of the greedy algorithm from the previous problem. Here you can analyze using a simpler direct argument where you imagine the hypothetical offline optimal solution and show that your solution gets at least half of that profit.

(b) Here you need to prove a lower bound on the competitive ratio for any deterministic algorithm. What this means is that you need to show the following:

For any online algorithm A, there is sequence of buyers \sigma_A (each buyer is defined by the set of items he is interested in), such that

\[ A(\sigma_A) \leq \frac{\text{OPT}(\sigma_A)}{2}. \]
6. This problem is asking you to do a probabilistic analysis for the “Random” online algorithm on the particular input described in the problem. For this graph, you need to show that the profit of the Random algorithm is at most \((1 + o(1))n\) (since the profit of the optimal offline algorithm is \(2n\)).

Here is an example when \(n = 3\).

![Diagram of buyers and items]

Hint: You will need to compute the expected number of items among \(n + 1, \ldots, 2n\) that remain after buyers 1, \ldots, \(n\) have arrived. For this purpose you might want to define indicator random variables \(X_i\) that tell you whether buyer \(i\) chooses item \(i\) or not. The value of \(E(X_i)\) depends on how many of the items in \(n + 1, \ldots, 2n\) have been sold when buyer \(i\) arrives. It will suffice for you to prove an upper bound on this. This achieves its maximum value when all of the bidders in \(1, \ldots, i - 1\) bought items in the set \(n + 1, \ldots, 2n\).

Also a reminder that \(\log n = o(n)\).