CS 331 Assignment #3

Greg Plaxton

February 17, 2014

NOTE ADDED 2/17/14. In the original version of the handout, a U'' appeared in the definition of $mcm(G, U')$. This has been corrected to U'. END OF NOTE.

There are three parts to this assignment. The first part, described in Section 1.1 below, consists of two paper-and-pencil exercises. You are encouraged to attend the discussion sections and office hours to get hints on how to solve these exercises. You are also permitted to ask for clarifications on Piazza. You are not allowed to work with other students on these exercises. This part is due *at the start of class* on Wednesday, February 26. Please note that no extension will be given on this part of the assignment!

The second part, described in Section 1.2 below, consists of a programming task that is due by 8pm on Monday, March 3. As in preceding assignments, you should follow the instructions in the document entitled "Guidelines for Programming Tasks", which may be found in the *Assignments* section of the class website. For this programming assignment, the secondary deadline is 8pm on Friday, March 7.

The third part, described in Section 2 below, consists of recommended exercises. There is nothing to be turned in for this part of the assignment, as these exercises will not be graded. You are encouraged to work on the recommended exercises in order to prepare for the tests.

1 Programming & Problem Solving

In Assignment 2, you implemented an incremental algorithm for the auction problem of Assignment 1. In this assignment, you will design and implement a more efficient incremental algorithm for the same computational problem. Once the n items have been read in, your algorithm will incur a one-time $O(n \log n)$ cost to sort the items by quality. Thereafter, each successive bid insertion will be processed in $O(n)$ time.

1.1 Paper-and-Pencil Exercises

We begin with a few useful definitions.

For any set of bids U, we define $\mathit{linear}(U)$ as the set of all linear bids in U, and we define $\mathit{single}(U)$ as the set of all single-item bids in U.

For any set of items V, we define $sort(V)$ as the sequence consisting of the items in V listed in nondecreasing order of quality, with ties broken by ID (lower ID comes first, say).

For any set of linear bids U, we define $sort(U)$ as the sequence consisting of the bids in U listed in nondecreasing order of slope, with ties broken by ID (lower ID comes first).

For any configuration $G = (U, V, E)$ and any set of single-item bids U' in U with distinct target items, we define $sort(G, U')$ as the sequence consisting of the bids in U' listed in ascending order of the positions in $sort(V)$ of their respective target items.

For any configuration $G = (U, V, E)$, we define $nice(G)$ as the set of all subsets U' of U such that there exists an MCM of G for which the set of matched bids is equal to U'.

For any configuration $G = (U, V, E)$ and any set of bids U' in $nice(G)$, we define $mcm(G, U')$ as the MCM of G constructed by: (1) matching each single-item bid in U' with its target item; (2) matching each linear bid in $sort(linear(U'))$ with the item in the corresponding position of $sort(V')$, where V' denote the set of items that were not matched in the previous step.

We are now ready to describe the new auction algorithm, which is based on the incremental framework of Assignment 2. In what follows, let $G = (U, V, E)$ denote the current configuration, and let M denote a (previously computed) MWMCM of G. Let G' denote the new configuration obtained from G by adding a new bid u^*. We wish to calculate an MWMCM M' of G'.

If the new bid u^* is a single-item bid on an item v that is matched in M to another single-item bid u (which might be the dummy single-item bid corresponding to the reserve price of the item), then it is easy to compute a new MWMCM M' in $O(1)$ time: We merely compare the offers of u^* and u, and retain the bid with the higher offer (if u is a dummy single-item bid, we interpret the reserve price of v as the offer of u), breaking ties arbitrarily. Because it is easy to insert a bid u^* meeting the conditions stated in the first sentence of this paragraph, we refer to such a bid insertion as *trivial*. All other bid insertions are said to be *nontrivial*. In what follows, we assume that the bid insertion associated with the new bid u^* is nontrivial.

Let U' denote the set of bids that are matched in M, and let U'' denote $U' + u^*$. Observe that for all u in U'', the set of bids $U'' - u$ belongs to $nice(G')$. Let u be a bid in U'' maximizing $w(mcm(G', U'' - u))$. From our work on Assignment 2, we know that $mcm(G', U'' - u)$ is an MWMCM of G'. In the present assignment, our goal is to compute such an MWMCM in $O(n)$ time, thereby improving upon the $O(n^2)$ bound of Assignment 2.

To facilitate this improvement, it is convenient to maintain two auxiliary arrays A and B along with the current configuration G and MWMCM M of G. Specifically, array A contains $sort(linear(U'))$ and array B contains $sort(G, single(U'))$.

We begin to process the insertion of bid u^* by inserting u^* into array A (if u^* is a linear bid) or B (if u^* is a single-item bid). So, for example, if u^* is a linear bid, we update array A to contain $sort(linear(U') + u^*)$ instead of $sort(linear(U'))$; this update is easily accomplished in $O(n)$ time.

Having inserted the new bid u^* into array A or B as appropriate, we make the following definitions. For each integer i such that $0 \leq i < |A|$, let M_i denote $mcm(G', U'' - A[i])$. For

each integer i such that $0 \leq i < |B|$, let M'_i denote $mcm(G', U'' - B[i])$.

By solving the two exercises below, you will learn how to compute an MWMCM M' of G' in $O(n)$ time. Remark: In the course of determining a particular MWMCM M' of G', you will identify the bid u in U'' that is unmatched in M'. In the program that you are asked to write in Section 1.2, you will delete u from array A (if u is a linear bid) or array B (if u is a single-item bid) in order to establish the desired relationship between G', M', A, and B; this is easy to do in $O(n)$ time.

Exercise 1. Show how to compute $w(M_i)$ for all i such that $0 \leq i < |A|$ in a total of $O(n)$ time.

Exercise 2. Show how to compute $w(M'_i)$ for all i such that $0 \leq i < |B|$ in a total of $O(n)$ time.

1.2 Programming Task

Your programming task is to solve the same computational problem as in Assignments 1 and 2. The total time used by your program to process an instance with m (non-dummy) bids and n items is required to be $O(mn + n \log n)$ time. In this bound, the $O(n \log n)$ term reflects the one-time cost of sorting the items by quality. The $O(mn)$ term reflects the total cost of processing the m bid insertions, each of which takes $O(n)$ time. It is strongly recommended that you implement the algorithm outlined in Section 1.1.

Within a few days we will provide sample input and output files that you can use to test your program. See the *Assignments* section of the class website to obtain these files. Any updates or clarifications to the assignment will also be posted there.

2 Recommended Exercises

1. Problem 5.2, page 246.
2. Problem 5.6, page 248.
3. Problem 6.5, page 316.
4. Problem 6.8(b), page 319. Clarification: The only robots eligible to be destroyed in second i are the ones that arrive in second i. Thus at most x_i robots can be destroyed in second i.
5. Problem 6.14, page 324.