CS 331 Assignment #1

Greg Plaxton

January 13, 2014

There are two parts to this assignment. The first part, described in Section 1 below, consists of a programming task that is due by 8pm on Monday, January 27.

Please be sure to read the document entitled "Guidelines for Programming Tasks", which may be found in the Assignments section of the class website. This document includes a description of the lateness policy for programming tasks. For this assignment, the secondary deadline is 8pm on Wednesday, January 29.

The second 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

Your friend Pat has created a website to auction off last-minute bed-and-breakfast accommodations in Austin. Every day from 10am to 11am, any Austin B&B that has registered with the site can add one or more rooms (for a one-night stay, arriving on that same day) to the auction. For each room that is added, a reserve price is specified; the reserve price represents the minimum amount that the B&B is willing to accept for that room. Pat has written a "quality estimation" program that uses customer feedback data along with data from other third-party services (e.g., TripAdvisor) to compute an overall numerical "quality" (higher is better) for each room.

At 11am, all of the rooms that have been submitted to the auction are posted on the website, and the bidding period begins. The reserve prices are kept private, but other relevant info concerning each room (e.g., name of B&B, room description/photos, room quality score as computed by Pat's program) is made available during the bidding period, which lasts until 11:30am. During the bidding period, two types of bids can be submitted. The first type of bid, which we refer to as a *single-item bid*, offers a specific amount of money for a specific room in the auction. The second type of bid, which we refer to as a *linear bid* specifies two monetary amounts: an *intercept* a and a *slope* b. The interpretation of a linear bid with intercept a and slope b is that the bidder is willing to pay $a + bq$ for any room with quality q.

In the current implementation of Pat's auction system, no information is published about the bids during the bidding period, so it is effectively a sealed-bid auction.

Once the auction closes at 11:30am, an "auction resolution" program is executed to determine who gets which room, and at what price. In more detail, the auction resolution program runs in two phases, as follows. The first phase computes something called an *assignment* of rooms to bids. For each room, an assignment specifies whether it is to be sold (if not, the room is retained by the seller), and if so, to which bid. In addition, an assignment is required to respect the following two constraints. First, a bid can win at most one room, i.e., an assignment cannot map multiple rooms to the same bid. Second, a single-item bid can win only the specific room named in that bid.

We define the *weight* of an assignment as the sum of the following monetary amounts, one for each room: a room with a reserve price of a that is retained by the seller contributes a to the weight; a room assigned to a single-item bid with an associated offer of a contributes a to the weight; a room with quality q that is assigned to a linear bid with intercept a and slope b contributes $a + bq$ to the weight. The goal of the first phase is to compute a maximum-weight assignment. Pat's current implementation of the first phase is based on a heuristic approach that is not guaranteed to produce a maximum-weight assignment. As discussed in greater detail below, your task is to create a program for the first phase that is guaranteed to produce a maximum-weight assignment.

The second phase of the auction resolution program determines the price paid by each winning bidder. Currently, Pat uses the following simple "pay-as-bid" rule to determine the prices: if a room is assigned to a single-item bid with an associated offer of a, then the price of the room is a; if a room with quality q is assigned to a linear bid with intercept a and slope b, then the price of the room is $a + bq$. Later in the course we will consider a different pricing scheme.

1.1 Your Task

The first line of the input will contain a single nonnegative integer n representing the number of items in the instance. The next n lines will each contains two integers (separated by a single blank) that specify the quality and reserve price (in cents) of the n items. We assign IDs to the items from 0 to $n - 1$ in the order that they appear in the input.

Every subsequent input line will either contain a single integer, or three blank-separated integers. The first integer on each line will belong to the set $\{1, 2, 3\}$.

If the first integer on a line is a 1, then the purpose of the line is to add a single-item bid to the auction, and there will be two more integers on the line that specify the offered amount (in cents) and the ID of the target item, respectively. Remark: The ID of the target item is guaranteed to be equal to the ID of an existing item.

If the first integer on a line is a 2, then the purpose of the line is to add a linear bid to the auction, and there will be exactly two more integers on the line that specify the intercept (in cents) and slope (in cents) of the bid, respectively.

We assign IDs to the bids by numbering them from 0 in the order that they appear in the input.

If the first integer on a line is a 3, then there will not be any other integers on the line, and your program should produce a single line of output containing $n + 1$ integers, where n is the number of items. The first number on the output line should be the weight of a maximum-weight assignment. The remaining n numbers on the output line should encode a maximum-weight assignment (if there is more than one maximum-weight assignment, your program will choose one of them arbitrarily), as follows: Indexing the remaining n numbers on the output line from 0 to $n - 1$, then the index-i number should be equal to the ID of the bid to which the item with ID i is assigned, or to -1 if the item with ID i is retained by the seller.

Upon reaching the end of the input file, your program should terminate.

For this assignment, it is recommended that you implement a "brute force" algorithm to compute a maximum-weight assignment. For example, you may wish to simply enumerate all possible assignments, calculating the weight of each, in order to identify a maximum-weight assignment. By proceeding in this manner it should be easy to ensure that the input-output behavior of your program is correct. You do not need to worry too much about the scalability of your algorithm, as we will only run it on extremely small instances.

1.2 Remarks

The inputs on which your program will be tested will be properly formatted in every respect, so your program does not need to check for formatting errors. For example, the ID of the target item for a single-item bid is guaranteed to be an integer in $\{0, \ldots, n - 1\}$.

Your program is required to work correctly even in the presence of negative reserve prices, qualities, single-item offers, intercepts, or slopes; i.e., you should not assume that these quantities are nonnegative.

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 1.4, page 23.
2. Problem 1.8, page 27.
3. Problem 2.4, page 67.
4. Problem 2.8, page 69.