CSC 373: Assignment 2

Worth 10%. Due July 6 at the beginning of tutorial (8pm).

The work you submit must be your own. You may discuss problems with each others; however, you should prepare written solutions alone. Copying assignments is a serious academic offence, and will be dealt with accordingly.

Question 1 [Problem 26 (Chapter 6, page 333 in the text) with explicit mention of Dynamic Programming technique and running time.]

Consider the following inventory problem. You are running a company that sells some large product (let's assume you sell trucks), and predictions tell you the quantity of sales to expect over the next n months. Let d_i denote the number of sales you expect in month i. We'll assume that all sales happen at the beginning of the month, and trucks that are not sold are *stored* until the beginning of the next month. You can store at most S trucks, and it costs C to store a single truck for a month. You receive shipments of trucks by placing orders for them, and there is a fixed ordering fee of K each time you place an order (regardless of the number of trucks you order). You start out with no trucks. The problem is to design an algorithm that decides how to place orders so that you satisfy all the demands $\{d_i\}$, and minimize the costs. In summary:

- There are two parts to the costs: (1) storage – it costs C for every truck on hand that is not needed that month; (2) ordering fees – it costs K for every order placed.
- In each month you need enough trucks to satisfy the demand d_i, but the number left over after satisfying the demand for the month should not exceed the inventory limit S.

Give an algorithm (using the dynamic programming technique) that solves this problem in time that is polynomial in n and S. Present the four steps clearly. What is the running time of your algorithm (in Θ notation) ?.

Question 2 Consider the Bitonic Euclidean Traveling-Salesman Problem (BETSP), which can be stated as follows. Given a collection of cities where no two cities have the same x-coordinate. There are two salemen at the westernmost city (the city with the smallest x-coordinate). The salesmen start from the westernmost city and stop at the easternmost city (the city with the largest x-coordinate). They need to visit *all* cities in between (each city needs to be visited by only one saleman). Furthermore, they can only go eastward, i.e., from one city a saleman can only go to another city with larger x-coordinate. The BETSP is to find the path (i.e. the sequence of cities to be visited) for each saleman so that the total traveling distance of both salemen is smallest.

Consider the example given in Figure 1. There are seven cities, the westernmost is v_1 and the easternmost is v_7.

In Figure 2, the paths given on the left is not optimal (the total distance is approximately 26.74); the optimal solution is given in the middle; and the paths given on the right are not valid, since one saleman (with undashed path) does not always go east.

Give an $\mathcal{O}(n^2)$-time dynamic programming algorithm for solving the BETSP. Indicate the four steps clearly. (Hint: sort the cities in increasing order of x-coordinate.)

Question 3 [Problem 1 page 246 in the text]

You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains n numerical values – so there are $2n$ values in total – and you may assume that no two values are the same. You'd like to determin the median of this set of $2n$ values, which we will define here to be the n-th smallest value.

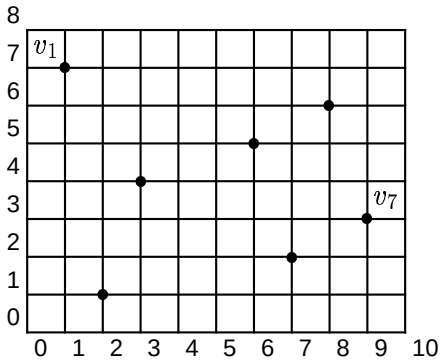

Figure 1: The two salemen start at v_1 and finish at v_7

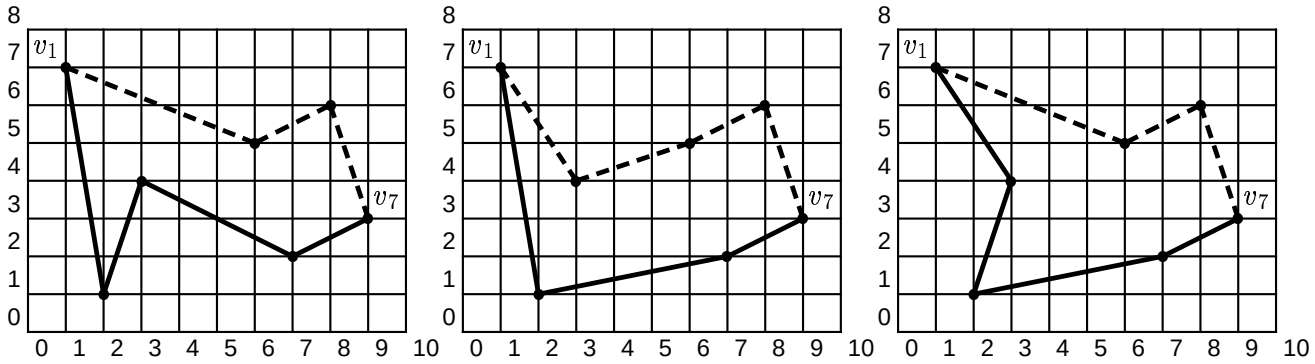

Figure 2: On the left: the total distance is 26.74. In the middle is an optimal solution, the total distance is 25.58. The output on the right picture is not valid, since one saleman does not always go eastward.

However, the only way you can access these values is through *queries* to the databases. In a single query, you can specify a value k to one of the two databases, and the chosen database will return the k-th smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

Give an algorithm that finds the median value using at most $\mathcal{O}(\log(n))$ queries.

Question 4

Give an algorithm that finds the minimum and the maximum elements of n numbers using no more than $3n/2$ comparisons.