CS 6363: Homework 2

Version 1.0

Instructor: Benjamin Raichel

Due Monday September 26, at 11:59 pm

Homework is to be submitted online through eLearning by the above deadline, and on paper in the following class. Typeset solutions (in LaTeX) are strongly preferred, but not required. If you chose to hand wright and scan your solutions, if what is written is illegible you will be marked down accordingly. Please make sure your name is on each page of the submission. You do not need to restate the problem, just the problem number.

Explanations are to be given for each problem, unless the problem specifically states it is not necessary. Explanations should be clear and concise. Rambling and/or imprecise explanations will be marked down accordingly.

Problem 1. Inversions (25 points)

An *inversion* in an array $A[1 \ldots n]$ is a pair of indices $\{i, j\}$ such that $i < j$ and $A[i] > A[j]$. The number of inversions in an n-element array is between 0 (if the array is sorted) and $\binom{n}{2}$ (if the array is sorted backward). Describe and analyze an algorithm to count the number of inversions in an n-element array in $O(n \log n)$ time. *[Hint: Modify mergesort.]*

Problem 2. Selection in sorted arrays (25 points)

As input you are given two sorted arrays $A[1 \ldots n]$ and $B[1 \ldots m]$ of integers. For simplicity assume that all $n + m$ values in the arrays are distinct. Describe and analyze an algorithm to find the kth ranked value in the union of the two arrays (where the 1st ranked value is the smallest element). The running time of your algorithm should be $O(\log(n + m))$.

Problem 3. Maximum subarray sum (25 points)

As input you are given an array $A[1 \ldots n]$ of integers. As you saw in class, in the maximum subarray sum problem, you are asked to find the value

$$\max \left\{ 0, \max_{i \leq j} \sum_{k=i}^{j} A[k] \right\}$$

Using dynamic programming, describe an $O(n)$ time algorithm to find the maximum subarray sum.

Problem 4. Starting a jewelery collection (25 points)

You are a thief who is planning to steal a collection valuable jewels on display at the museum. There are n different jewels in total, each has a different size and black market value. The sizes are given in an array $S[1 \ldots n]$ and the values are given in an array $V[1 \ldots n]$. Unfortunately you cannot steal all the jewels, and instead can only take what will fit in your robber's bag which has size b (and a big dollar sign on the front of course). Specifically, the sum of the sizes of the items that you choose should be less than or equal to b. You can assume b as well as the entries in S are all positive integers, and the values in V are positive real numbers.

Let $[n] = \{1, \ldots, n\}$ denote the set of jewels. Naturally, you wish to steal a subset $J \subseteq [n]$ of jewels which maximizes the total value subject to fitting in you bag. Fortunately, you've taken CS 6363 and so using dynamic programming you know which jewels to steal.

Describe and analyze an $O(nb)$ time algorithm that outputs the maximum total value of the jewels that you can steal. Formally, let $W = \{J \subseteq [n] \mid \sum_{i \in J} S[i] \leq b\}$, then your algorithm should output $\max_{J \in W} \sum_{i \in J} V[i]$.