CS 6363: Homework 2

Version 1.2

Instructor: Benjamin Raichel

Due Wednesday September 26, at 11:59 pm

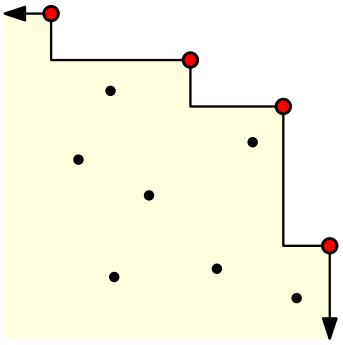
Homework is to be submitted online through eLearning by the above deadline. Typeset solutions (in LaTeX) are strongly preferred, but not required. If you chose to hand wright and scan your solutions, if what is written is illegible you will be marked down accordingly. Please make sure your name is on 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. 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 $O(\log(n + m))$. Note: If your analysis happens to give an $O(\log(n) + \log(m))$ running time, observe that $\log(n) + \log(m) = O(\log(n + m))$.
[Hint: Recall Binary search and/or Quickselect.]

Problem 2. Cascading Waterfalls (25 Points)



A set P of 11 points, where $CW(P)$ has four points shown in red.

Given two points $p = (p.x, p.y)$ and $q = (q.x, q.y)$ in the plane, we say that q *dominates* p if $q.y > p.y$ and $q.x > p.x$. Given a set P of n points in the plane, the *cascading waterfall* of P, denoted $CW(P)$, is the subset of points from P which are *not* dominated by any other point in P. Intuitively, the points in $CW(P)$ look like the boundary of a cascading waterfall. For an example, see the above figure.

Describe and analyze a *divide and conquer* algorithm to compute $CW(P)$ in $O(n \log n)$ time. For simplicity assume any two points in P have distinct x and y coordinate values.

Problem 3. Longest Convex Subsequence (25 points)

Call a sequence $x_1, \ldots, x_m$ of numbers convex if $x_i < \frac{x_{i-1}+x_{i+1}}{2}$ for all $1 < i < m$. Use dynamic programming to give an $O(n^3)$ time algorithm to compute the length of the longest convex subsequence of an arbitrary array $A[1 \ldots n]$ of n integers.

Problem 4. Maximum Subarray Product (25 points)

As input you are given an array $A[1 \ldots n]$ of numbers (which could be zero, negative, positive, and may not be integers). The maximum subarray product question asks you to find the subarray $A[i \ldots j]$ whose elements have the largest product, i.e. the value:

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

Use dynamic programming to find the maximum subarray product in $O(n)$ time.

Side Note: The maximum subarray sum problem from class also can be solved in $O(n)$ time using dynamic programming, but differs in form from the $O(n \log n)$ time divide and conquer algorithm from class. In particular, rather than cutting in the middle, it is easier to consider elements one at a time. As an aside, the maximum subarray sum can also be computed in $O(n)$ with a greedy algorithm.

Hints:

- For each index consider the best solution starting at exactly that index. Then once you have all these values take the best one.
- The maximum subarray product can end at a negative element, so you may want your recursive subproblems to remember slightly more information to handle this case.