The University of British Columbia

Department of Computer Science

Computer Science 420—Advanced Algorithm Design and Analysis

Homework Assignment 1
Review and Warmup Questions

Due: 2015 January 15

Please review the class policy on collaboration before starting this assignment. You are free to discuss problems in groups of size at most three. However, your actual homework submission must be prepared on your own. At the top of the first page of every homework submission you must provide a statement about external resources used in the preparation of your submission. This must clearly acknowledge all resources (including books, websites and discussions with fellow students) that you have used. Submissions missing this statement will not be graded.

1) What statement must appear (together with your name and student number) at the top of the first page of every homework submission in this course?

2) Consider the problem of finding *both* the maximum and minimum of a set of n numbers. We observed in class that this combined task can be solved more efficiently than treating the individual sub-problems separately.

   a) Describe, using pseudocode, an algorithm that, given an array $X[1:n]$ of numbers, outputs both the maximum and minimum of X, using at most $\lceil 3n/2 \rceil - 2$ comparisons.

   We want to devise an *adversary strategy* (what we have also called an *anti-algorithm*) to show that $\lceil 3n/2 \rceil - 2$ comparisons are also *necessary*, in the worst case.

   Our adversary strategy determines the outcome of comparisons whose outcome is not already determined by the results of previous comparisons in way that tries to *maximize* the number of comparisons made by the algorithm on some legitimate input. To do so it maintains the numbers (contestants) in four disjoint groups:

   Group A – contestants that have never played (and hence are still candidates for both the maximum and the minimum).

   Group B – contestants that have won one or more contests but have never lost (and hence are still candidates for the maximum).

   Group C – contestants that have lost one or more contests but have never won (and hence are still candidates for the minimum).

   Group D – contestants that have both lost and won at least once (and hence are no longer candidates for either the maximum or the minimum).

   The adversary ensures that group B contestants always win (against other types) and group C contestants always lose (against other types). Otherwise the outcome of contests is chosen arbitrarily (as long as it is consistent with the outcomes of previous comparisons).

   b) Show that, at any stage, an algorithm playing against this adversary is forced to make at least $f(A,B,C) = \lceil 3|A|/2 \rceil + |B| + |C| - 2$ more comparisons before it has determined the maximum and the minimum. (Hint: argue by induction on $f(A,B,C)$, considering all possible cases for the "next" comparison.)

   c) Argue that the lower bound ($\lceil 3n/2 \rceil - 2$) follows.

3) In each of the following subproblems S denotes a given set of n points on the real interval $[0, 1]$. In all cases you should (i) describe the most efficient algorithm that you know of that solves the problem, (ii) give a brief analysis of the complexity of the algorithm, and (iii) say what you can about the *intrinsic cost* of solving the problem (in the worst case). You should assume that S is given as an *unsorted* array $A[0 : n-1]$.

a) Determine a point p_c (not necessarily in S) that minimizes $max_{q \in S} |q - p_c|$.

b) Determine a point p_c (not necessarily in S) that minimizes $\sum_{q \in S} |q - p_c|$.

c) Determine a point $p_c \in [0, 1]$ (not necessarily in S) that maximizes $min_{q \in S} |q - p_c|$.

d) Determine the smallest r such that there exists a point $p_c \in [0, 1]$ (not necessarily in S) satisfying: $|S \cap [p_c - r,\ p_c + r]| \geq 2$.

The following question is for PRACTICE ONLY. It is NOT to be handed in.

4) Suppose that we are given two arrays $A[1 : n]$ and $B[1 : m]$ of real numbers, that encode a collection V_A of n vertical lines $x = A[i]$, $i = 1, \ldots n$, and a collection H_B of m horizontal lines $y = B[j]$, $j = 1, \ldots, m$. Any path P in the plane joining two specified points s and t will cross some subset (possibly empty) of the lines in $V_A \cup H_B$.

a) Describe an algorithm that, given points s and t, finds a path P from s to t that crosses the smallest subset of the lines in $V_A \cup H_B$.

b) Argue that your algorithm is correct. (In particular, you must argue that no other path from s to t exists that crosses fewer lines.

Suppose now that V_A and H_B are line *segments* (rather than full lines), specified by additional arrays $Y^{min}[1 : n]$, $Y^{max}[1 : n]$, $X^{min}[1 : m]$ and $X^{max}[1 : m]$, where the i-th segment in V_A joins points $(A[i], Y^{min}[i])$ and $(A[i], Y^{max}[i])$, and the j-th segment in H_B joins points $(X^{min}[j]), B[j])$ and $(X^{max}[j], B[j])$.

c) Describe an algorithm to determine if there exists a path P from s to t that does not intersect any segment in $V_A \cup H_B$. Argue the correctness of your algorithm.

d) Describe an algorithm that determines a path P from s to t that crosses the minimum number of segments in $V_A \cup H_B$. What is the worst-case cost of your algorithm (as a function of n and m)?