CS 466/666, Spring 2015
Timothy Chan

Assignment 1 (due May 27 Wednesday 5pm)

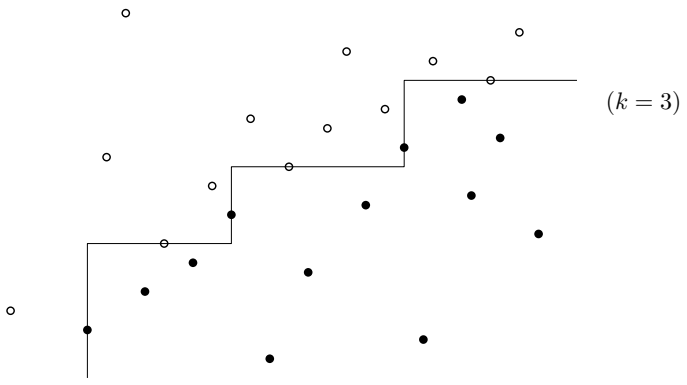
Please read `http://www.student.cs.uwaterloo.ca/~cs466/policies.html` first for general instructions.

1. [*18 marks*] Given an undirected graph $G = (V, E)$ with n vertices and m edges ($m \geq n$), consider the problem of deciding whether G contains a clique of size 4, i.e., whether there exists four vertices $v_1, \ldots, v_4 \in V$ with $v_i v_j \in E$ for all $i, j \in \{1, \ldots, 4\}$ ($i \neq j$). Naive brute-force search requires $O(n^4)$ time, but in this question, you will explore faster algorithms.

   (a) [*6 marks*] Give an $O(n^{3.373})$-time algorithm for this problem. (You may use the $O(n^{2.373})$-time triangle-finding algorithm from class as a subroutine.)

   (b) [*12 marks*] Give an $O(m^{1.69})$-time (or better) algorithm for the problem. [Hint: adapt the "high-low" trick from class. For the "low" case with a given parameter d, aim for a runtime of $\sum_{v \in V: \deg(v) \leq d} \deg(v)^{2.373}$, which would be at most $O(md^{1.373})$ (why?). You may use the triangle-finding algorithm from class as a subroutine. For the "high" case, use part (a)...]

2. [*25 marks*] We are given a set B of n black points and a set W of n white points in 2D. (The points are *not* given in sorted order.) The problem is to find a *staircase* with the smallest number of steps that separates the black and the white points, with all black points on or below the chain and all white points on or above the chain. Here, a staircase with k steps is a polygonal chain with vertices $(x_1, -\infty), (x_1, y_1), (x_2, y_1), (x_2, y_2), (x_3, y_2), \ldots, (x_k, y_k), (\infty, y_k)$ for some $x_1 < x_2 < \cdots < x_k$ and $y_1 < y_2 < \cdots < y_k$. The output is the sequence $\langle x_1, y_1, x_2, y_2, \ldots, x_k, y_k \rangle$ of an optimal staircase. (The optimal solution may not be unique. You may assume that a solution exists and that no two x- or y-coordinates are identical.)



   The following greedy approach is known to find an optimal solution:

1. $x_0 = y_0 = -\infty$
2. for $i = 0, 1, \ldots$ do {
3. $\quad x_{i+1}$ = the smallest x-coordinate among all black points $(x, y) \in B$ with $y > y_i$
4. $\quad y_{i+1}$ = the smallest y-coordinate among all white points $(x, y) \in W$ with $x > x_i$
5. $\quad$ if $x_{i+1} = \infty$ then return $\langle x_1, y_1, x_2, y_2, \ldots, x_i, y_i \rangle$
   }

(You do not need to prove correctness of this greedy algorithm.)

(a) [*2 marks*] What is the running time of the naive implementation of the above greedy algorithm as a function of n and k (the smallest number of steps)?

(b) [*6 marks*] In preparation for part (c), describe how to preprocess any set S of k points in 2D in $O(k \log k)$ time, so that given any value b, we can find the smallest x-coordinate among all points $(x, y) \in S$ with $y > b$ in $O(\log k)$ time. [Hint: for the preprocessing, you may want to first sort the points in decreasing y-order...]

(c) [*12 marks*] Now describe how to speed up the above greedy algorithm to obtain an $O(n \log k)$-time algorithm for the original problem. [Hint: borrow ideas from the $O(n \log h)$ convex hull algorithm from class; use part (b) as a subroutine.[1]]

(d) [*5 marks*] Prove that as a function of n (but not k), the problem requires a lower bound of $\Omega(n \log n)$ worst-case time in a comparison-based model of computation. [Hint: you may use the standard $\Omega(n \log n)$ lower bound for sorting n integers in a comparison-based model of computation.]

3. [*17 marks*] The *median* of a set S of n elements (real numbers) is an element of rank $\lfloor 0.5n \rfloor$. (The *rank* of x refers to the number of elements smaller than x.) It is known (e.g., see [CLRS, Ch9] or any algorithms textbook) that the median can be computed in $O(n)$ time.

   In this question, we study a dynamic version of the problem: how to maintain an approximate median under insertions of new elements. For our purposes, an *approximate median* is defined as an element of rank between $0.49n$ and $0.51n$.

   Consider the following simple "lazy" strategy: don't update the median for the next $0.01n$ elements; after we have seen $0.01n$ elements, recompute the median from scratch (by the $O(n)$ algorithm mentioned above). The precise pseudocode is given below. Initially, $n_0 = n = \text{count} = y = 0$.

   insert(x):
   1. $n = n + 1$, $A[n] = x$, count = count + 1
   2. if count $\geq 0.01 n_0$ then {
   3. $\quad y$ = the (exact) median of $A[1], \ldots, A[n]$
   4. $\quad n_0 = n$, count = 0
      }
   5. return y

[1] In multi-part questions such as this, even if you are unable to do (b) correctly, you may still use the result stated in (b) to do (c).

(a) [*2 marks*] Show that insert() always returns an approximate median of $A[1], \ldots, A[n]$.

(b) [*1 marks*] What is the worst-case running time of insert()?

(c) [*6 marks*] Determine the amortized running time of insert(). Use a direct argument that avoids potentials.

(d) [*8 marks*] Redo part (c), this time using the potential method.