CS 3330: Homework 6

Fall 2017

Given two points $p = (p.x, p.y)$ and $q = (q.x, q.y)$ in the plane, we say that p is *northeast* of q if $p.x \geq q.x$, $p.y \geq q.y$, and at least one of these two inequalities is strict. For example, $(4, 3)$ is northeast of $(3, 2)$, $(3, 3)$ is northeast of $(3, 2)$, and $(5, 2)$ is northeast of $(3, 2)$. Whereas $(1, 0)$ is not northeast of $(3, 2)$, $(4, 1)$ is not northeast of $(3, 2)$, $(1, 4)$ is not northeast of $(3, 2)$, and $(3, 2)$ is not northeast of $(3, 2)$.

Given a set P of points in the plane, we say that a point $q \in P$ is *extreme* if no point in P is to the northeast of q. For example, with

$$P = \{(1, 0), (3, 2), (1, 4), (4, 1), (5, 2)\},$$

the only extreme points are $(1, 4)$ and $(5, 2)$.

In this assignment, we consider the problem of computing the set of extreme points in a given set P of points. The straightforward algorithm for this works as follows. We consider each point q in P, and determine if it is extreme by going through every other point r in P, and checking if r is to the northeast of q; if no point is to the northeast of q, we add q to the set of extreme points.

In this assignment, your task is to develop an algorithm that is substantially faster than the straighforward algorithm on input point sets that are "nicely distributed". To generate a nicely distributed point set with n points, we sample n points as follows: to generate a point $p = (p.x, p.y)$, we let $p.x$ to be a random integer in the range $[0, n]$, and we let $p.y$ to be an independent random integer in the range $[0, n - p.x]$. This process gives us a set of n points in the triangle formed by the points $(0, n)$, $(0, 0)$, and $(n, 0)$.

The algorithm you develop should correctly return the set of extreme points in the input point set. It should be noticably faster than the straightforward algorithm on nicely distributed point sets. Furthermore, the ratio of the actual running time of the developed algorithm to the actual running time of the straightforward algorithm should tend to decrease as we increase n.

What you should submit. Your Python/Java program, when run, should prompt the user to input a value of n. Upon receiving the value of n, it should generate a nicely distributed point set with n points, run the straightforward algorithm on it, output the time taken, then run your algorithm on it, and output the time taken.

You need to submit your source code in the ICON dropbox for Homework 6. Please include instructions on how the user may run the program and specify n. In addition, please submit some documentation that includes a high level description of your algorithmic idea, and some typical running times of the two algorithms for the following values of n: 1000, 2000, 5000, 10000, 50000, 100000.

1. Given the way we generate points, we may have multiple copies of the same point. Thus, the input to the two algorithms will actually be a multiset rather than a set.
2. Developing a good algorithm for arbitrary inputs (as opposed to nicely distributed ones) is probably harder, and is not required for this assignment.
3. While your algorithm should exploit features of the input point distribution, don't make it too dependent on the particular method for generating data. In particular, your algorithm should not explictly use the fact that the point coordinates are in the range $[0, n]$.