Computatinal Geometry (Fall 2012)
Project 1: Convex Hull Computation

Peyman Afshani

September 24, 2012

Parts (A), (B), and (C) are mandatory. The rest of the project is optional (the optional parts are also marked with a +). However, you can choose to do all the optional parts instead of part (C).

In this project, you will implement a few convex hull algorithms and will compare their performances. After implementing each algorithm and making sure that it runs correctly, you must evaluate its performance on some "test cases". Each test case is essentially a (large) input set of points in the plane. For this project, create at least five input sets (test cases). Include one input point set generated uniformly randomly inside a square, another generated uniformly randomly inside a circle, and one input point set whose points lie on the curve $Y = X^2$. You are free to pick the other two input sets however you wish.

Run your algorithms on all of your test cases and measure their running times. Separate the running times into two parts: how much time your algorithm spends reading the input and how long it takes to compute the convex hull. Set the number of points in your input sets large enough that your algorithm takes up about 70% of the RAM of the computer. In your report, include all these times as well as the specifications of the machine you used, number of input points and the number of the convex hull points.

Your implementations should accept an input file of the following format. The i-th line of the input file will be empty if it is the last line of file. Otherwise, the i-th line will describe the point p_i and it will contain two floating point numbers that represent the x- and y-coordinate of p_i, respectively. The numbers are separated by a single space. We also say p_i has label i (for example, the first line of the input file describes the point p_1 which has label 1).

The output of the algorithm should be in the following format. The first line of the file will contain an integer that is the label of the convex hull point with the smallest x-coordinate. If there are two convex hull points with the smallest x-coordinate, then the one with the larger y-coordinate should be first. The following lines should contain the labels of the vertices of the convex hull in the clockwise order (i.e., first the upper hull vertices should appear in left to right order, then the lower hull vertices from right to left). Note that all the numbers in the output file are integers.

Sample input file (see Figure 1(a)):

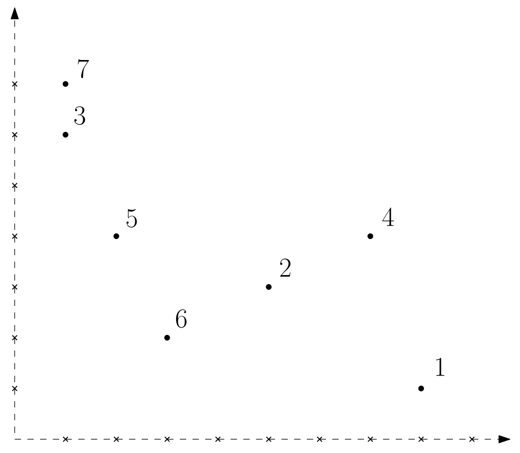
```
8 1
5 3
1 6
7 4
2 4
3 2
1 7
```

Sample output file (see Figure 1(b)):

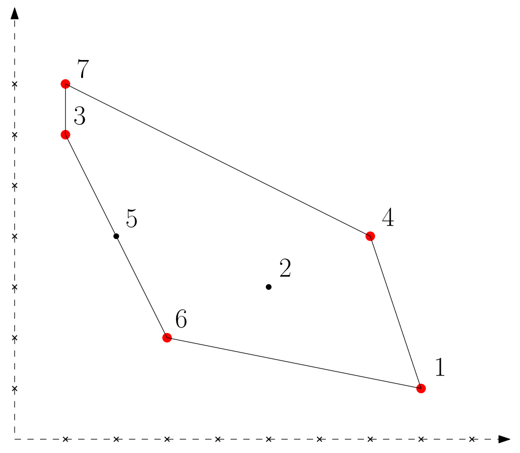
```
7
4
1
6
3
```

Part A. Implement and test the incremental convex hull algorithm discussed in the class (the one discussed in pages 6 and 7 of BKOS). Let's call this algorithm INC_CH.

Part B. Implement and test a divide-and-conquer convex hull algorithm. You can implement the algorithm discussed in the class or you can come up with your own divide-and-conquer algorithm. Follow the previous



(a) The points in the sample input with their labels.



(b) The convex hull points are colored in red and are marked with larger circles.

input and output conventions. Let's call this algorithm DC_CH.

Hint. In your implementation, you do not need to implement the exact median finding algorithm and instead you can use the following heuristic: pick five random points and then pick the median among the five random points.

Part C. Implement the marriage-before-conquest convex hull algorithm. Follow the following pseudocode for the upper hull construction (use a similar code for the lower hull). Let's call this algorithm MbC_CH.

1. Find the point with median x coordinate $p_m = (x_m, y)$ and partition the input into two sets P_ℓ and P_r where P_ℓ contains all the points with x-coordinate smaller than x_m and P_r contains the rest of the points.
2. Find the "bridge" over the vertical line $X = x_m$ (i.e., the upper hull edge that intersects line $X = x_m$). You need to implement linear programming for this step. Let (x_i, y_i) and (x_j, y_j) be the left and right end points of the bridge.
3. Prune the points that lie under the line segment $(x_i, y_i), (x_j, y_j)$ (these will be the points whose x-coordinates lie between x_i and x_j.
4. Recursively compute the upper hull of P_ℓ and P_r.

Next, add one more pruning step to the above algorithm and call it MbC2_CH. This extra pruning step is the step 2 in the algorithm below.

1. Find the point with median x coordinate $p_m = (x_m, y)$ and partition the input into two sets P_ℓ and P_r where P_ℓ contains all the points with x-coordinate smaller than x_m and P_r contains the rest of the points.
2. Find the point p_ℓ with the smallest x-coordinate (if there are more than one, take the one with the largest y-coordinate) and the point p_r with the largest x-coordinate (if there are more than one, take the one with the smallest y-coordinate). Note that these can be done at the same time as step 1. Prune all the points that lie under the line segment $p_\ell p_r$.
3. Find the "bridge" over the vertical line $X = x_m$ (i.e., the upper hull edge that intersects line $X = x_m$). Let (x_i, y_i) and (x_j, y_j) be the left and right end points of the bridge.

4. Prune the points that lie under the line segment $(x_i, y_i), (x_j, y_j)$ (these will be the points whose x-coordinate lie between x_i and x_j.
5. Recursively compute the upper hull of P_ℓ and P_r.

Part D+. Prove that MbC_CH algorithm runs in $O(n \log h)$ time where h is the number of points on the convex hull (hint: look at the recursion tree. Observe that each time you recurse, the number of points halves).

Part E+. Let $x_1 < \cdots < x_h$ be the x-coordinates of the points on the upper hull and let n_i be the number of input points $p = (x, y)$ such that $x_i \le x < x_{i+1}$, $1 \le i < h$. Show that the upper hull computation in MbC_CH runs in time

$$O\left(\sum_{i=1}^{h-1} n_i \log\left(\frac{n}{n_i}\right)\right).$$

Part G$^+$. Implement and measure the performance of Chan, Snoeyick and Yap's convex hull algorithm. Let's call this CSY_CH. For reference, the rough pseudocode of the upper hull construction in this algorithm is the following.

UpperHull(P, p_ℓ, p_r): Here, P is the input point set, p_ℓ is the point with the smallest x-coordinate in P, P_r is the point with the largest x-coordinate in P and the procedure computes the upper hull of P.

1. Prune points below the line segment $p_\ell p_r$.
2. Pair points randomly into $\lfloor n/2 \rfloor$ pairs, (s_i, t_i), $1 \le i \le \lfloor n/2 \rfloor$, in which s_i has smaller x-coordinate than t_i.
3. Find the pair (s_m, t_m) with the median slope among the pairs (you can use the median finding heuristic mentioned in the previous hint).
4. Find the maximal point $p_m = (x_m, y_m)$ in the direction of (s_m, t_m).
5. Partition P into two sets: P_ℓ and P_r in which P_r contains all the points of P with x-coordinate larger than x_m and P_ℓ contains the rest.
6. Prune P_r: if P_r contains a pair (s_i, t_i) with slope larger than the median slope, then prune s_i.
7. Prune P_ℓ: if P_r contains a pair (s_i, t_i) with slope smaller than the median slope, then prune t_i.
8. Recursively compute the upper hull of P_r and P_ℓ.

Part H$^+$. Consider a partition, Δ, of the point set P into k disjoint subsets $P_1, \cdots, P_k$ and let $n_i = |P_i|$, $1 \le i \le k$. We say that Δ is "good" if for every i, $1 \le i \le k$, either $n_i = 1$ or P_i can be placed inside a triangle t_i in such a way that t_i completely lies below the upper hull of P. Define the "entropy" of Δ as

$$H_\Delta = \sum_{i=1}^{k} \frac{n_i \log(\frac{n}{n_i})}{n}.$$

Prove that for any good partition Δ, the upper hull computation in CSY_CH algorithm or MbC2_CH algorithm runs in $O(nH_\Delta)$ time. Prove that this is not the case for MbC_CH algorithm, that is, there exists a point set P and a good partition Δ such that nH_Δ is asymptotically smaller than the time it takes for MbC_CH algorithm to compute the upper hull of P.