CS31005 Algorithms – II

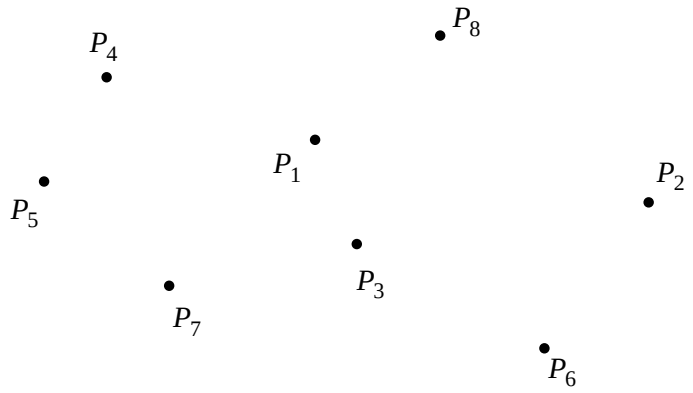
Class Test 2

Date: 02-Nov-2020 Maximum marks: 50

Instructions

- Answer all the questions. Be brief and precise.
- If you use any algorithm/result/formula covered in the lectures or the tutorials, just mention it, do not elaborate.

1. Explain the working of Graham's scan for computing the upper hull of the following set of eight points. **(10)**

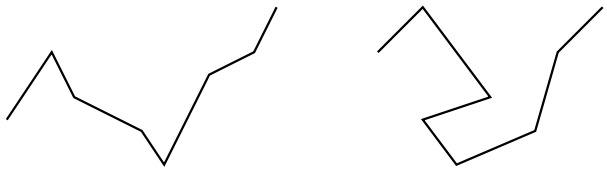


Solution The points in sorted order of x-coordinates are $P_5, P_4, P_7, P_1, P_3, P_8, P_6, P_2$. The following operations are performed on a stack.

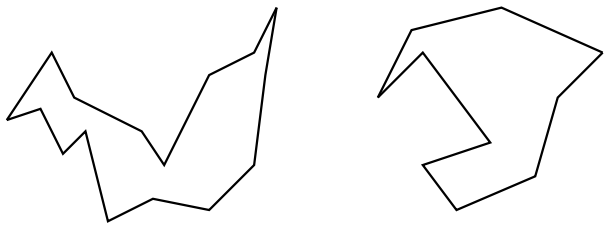
P_5 :	Push P_5	[Init]
P_4 :	Push P_4	[Stack has only one element]
P_7 :	Push P_7	[Right turn]
P_1 :	Pop P_7	[Wrong turn]
	Push P_1	[Right turn]
P_3 :	Push P_3	[Right turn]
P_8 :	Pop P_3	[Wrong turn]
	Pop P_1	[Wrong turn]
	Push P_8	[Right turn]
P_6 :	Push P_6	[Right turn]
P_2 :	Pop P_6	[Wrong turn]
	Push P_2	[Right turn]

The final content of the stack from bottom to top is P_5, P_4, P_8, P_2.

2. A chain of non-intersecting line segments is called progressive if the x-coordinate always increases during the traversal of the chain from the leftmost point to the rightmost point. A simple (but not necessarily convex) polygon is called progressive if it consists of two progressive chains. The following figure demonstrates this concept.



Progressive chain Non−progressive chain

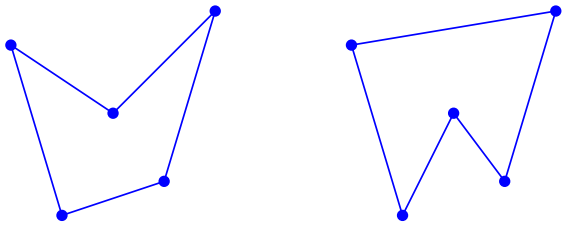


Progressive polygon Non−progressive polygon

You are given a set S of $n \geqslant 3$ points. Your task is to construct a progressive polygon P having its only vertices at *all* of the given points in S. You may assume that the points in S are in general position.

(a) Prove/Disprove: For every S, the polygon P can be constructed uniquely. **(4)**

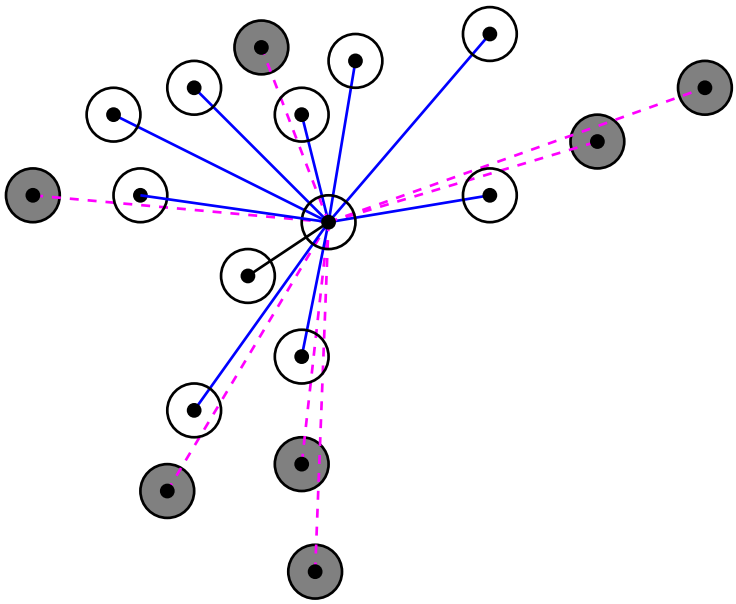
Solution False. See the following counterexample.



(b) Propose an $O(n \log n)$-time algorithm to construct P from S. **(6)**

Solution Locate the leftmost point P and the rightmost point Q (these are unique by the assumption of general position). Let U be those of the given points that lie above PQ, and V those that lie below PQ. Sort U and V by an $O(n \log n)$-time algorithm. Start at P, follow the points in the sorted U, go to Q, follow the points in the sorted V in the reverse order, and finally come back to P.

3. There are n cell-phone towers in the plane. The towers are located at the points (x_i, y_i) for $i = 1, 2, 3, \ldots, n$. Around each tower, there is an interference zone of a fixed radius r (the same for all the towers). Assume that the interference zones do not overlap with each other. Two towers can communicate without interference if the line segment joining them does not intersect with the interference zone of any other tower. Your task is to determine all the towers with which the first tower (located at (x_1, y_1)) can communicate without interference. The following figure gives an example. Propose an $O(n \log n)$-time algorithm to solve this problem. Clearly mention the data structures that your algorithm uses. Also justify that your algorithm actually achieves the given running time. **(10)**



Solution (Sketch) Use a ray-sweep algorithm. The ray emanates from the position (x_1, y_1) of the first tower.

Active towers: Those for which the interference zones intersect with the ray. These are kept sorted with respect to their distances from (x_1, y_1).

Events:

- **Enter circle:** Insert the tower in the list of active towers.
- **Leave circle:** Delete the tower from the list of active towers.
- **Center of circle:** Report the tower as supporting interference-free communication if and only if it is the closest active tower from (x_1, y_1).

The list of active towers should support arbitrary insertion and deletion, and should therefore be implemented as a height-balanced tree.

The event queue may be implemented as a heap. There are $3(n-1)$ events. All can be initially put in the heap, and a makeheap operation is performed on the heap. The heap ordering is with respect to the angles with the horizontal.

4. You are given two arrays $A = (a_1, a_2, a_3, \ldots, a_m)$ and $B = (b_1, b_2, b_3, \ldots, b_n)$ of *positive* integers. Your task is to find out whether there exist a *non-empty* subset I of $\{1, 2, 3, \ldots, m\}$ and a *non-empty* subset J of $\{1, 2, 3, \ldots, n\}$ such that $\sum_{i \in I} a_i = \sum_{j \in J} b_j$. Prove that this problem is NP-Complete. **(10)**

Solution Let us call the given problem EQUAL-SUM. A specification of I and J is a succinct certificate for EQUAL-SUM, implying that the problem is in NP. In order to prove its NP-Hardness, we make a reduction from the SUBSET-SUM problem. Let $S = (s_1, s_2, \ldots, s_k)$ and the target sum t constitute an instance for SUBSET-SUM. Here, all s_i and t are positive integers. Take $A = S$ and $B = \{t\}$.

5. You are given an undirected graph $G = (V, E)$ and a positive integer k. You want to determine whether there exist k or fewer edges in E such that the removal of these edges from E makes G bipartite. Prove that this problem is NP-Complete. (**Hint:** Use the max-cut problem for the reduction.) **(10)**

Solution The problem is clearly in NP, because the list of edges to remove from E in order to make it bipartite is a succinct certificate for the problem. It is easy to verify whether the list contains $\leqslant k$ edges, and their removal from G leaves a bipartite graph.

For NP-Hardness, we make a reduction from MAX-CUT. Let (G, r) be an instance for MAX-CUT. If m is the number of edges in G, generate the instance (G, k) of the given problem, where $k = m - r$. G has a cut of size $s \geqslant r$ if and only if the removal of the remaining $m - s$ edges makes G bipartite. The number of edges removed is $m - s \leqslant m - r = k$.