CS31005 Algorithms – II, Autumn 2021–2022

Second Test

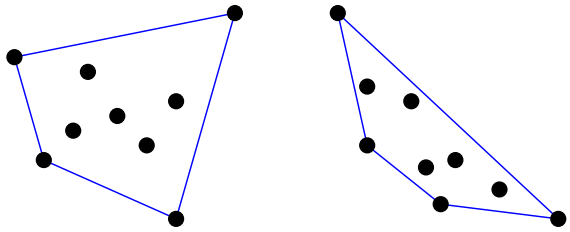
Oct 05, 2021, 2:15pm Maximum marks: 50

Duration: 75 minutes (60 minutes for answering questions + 15 minutes for download/submission)

All your answers MUST BE HANDWRITTEN on paper. Scan all papers with your answers in a SINGLE pdf, and upload it as the answer to the Quiz in Moodle. The size of the final pdf must be less than 10 MB. You must upload the pdf strictly by 3:30 pm Moodle server time, the quiz submission will close after that.

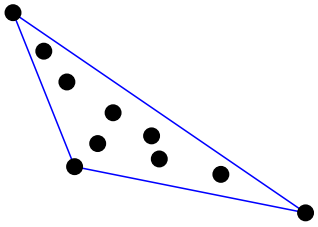
1. You are given the four corners of a convex quadrilateral along with n other points in the interior of the quadrilateral. Your task is to identify the four corners of the quadrilateral from the given set S of $n+4$ points, and output them in clockwise order. Assume that the points of S are in general position. To solve the problem, you compute the leftmost point L, the rightmost point R, the topmost point T, and the bottommost point B in S, and output the sequence L,T,R,B. Prove/Disprove the correctness of the algorithm. **(6)**

Solution Incorrect algorithm. It is not guaranteed that the points L,T,R,B are distinct. The following examples illustrates two such situations.

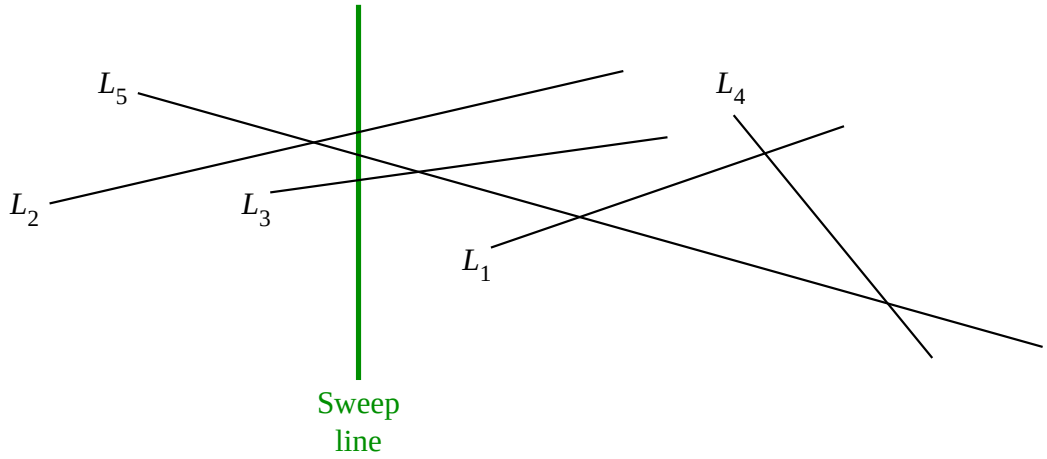


1. You are given the three corners of a triangle along with n other points in the interior of the triangle. Your task is to identify the three corners of the triangle from the given set S of $n+3$ points, and output them in clockwise order. Assume that the points of S are in general position. To solve the problem, you compute the leftmost point L, the rightmost point R, the topmost point T, and the bottommost point B in S. By the pigeon-hole principle, there must be a repetition among L,T,R,B. Remove the repeated point, and output the remaining three points in clockwise order. Prove/Disprove the correctness of the algorithm. **(6)**

Solution Incorrect algorithm. There may be two repetitions among L,T,R,B. The following examples illustrates this situation.



2. Consider the line-sweep algorithm covered in the class for computing the intersections of line segments. The algorithm is applied to the five lines in the following figure. Suppose that the sweep line is vertical, moves from left to right, and is currently at the position as shown in the figure. Denote the enter-segment events as $ES(L_i)$, the leave-segment events as $LS(L_i)$, and the intersection events as $\cap(L_i,L_j)$.



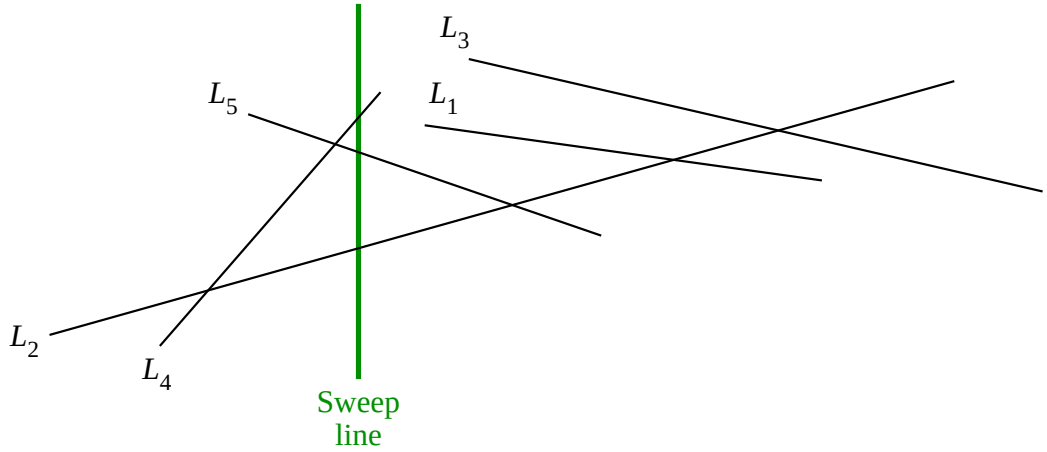
(a) List, in the order of occurrence, all the events that the sweep line encounters as it completes its sweep from the current position. **(4)**

Solution $\cap(L_3,L_5)$, $ES(L_1)$, $\cap(L_1,L_5)$, $LS(L_2)$, $LS(L_3)$, $ES(L_4)$, $\cap(L_4,L_1)$, $LS(L_1)$, $\cap(L_4,L_5)$, $LS(L_4)$, $LS(L_5)$.

(b) Which of these events are stored in the event queue at the current position of the sweep line? Assume that the event queue is initialized by all enter-segment and leave-segment events. **(3)**

Solution $\cap(L_3,L_5)$, $ES(L_1)$, $LS(L_2)$, $LS(L_3)$, $ES(L_4)$, $LS(L_1)$, $LS(L_4)$, $LS(L_5)$.

2. Consider the line-sweep algorithm covered in the class for computing the intersections of line segments. The algorithm is applied to the five lines in the following figure. Suppose that the sweep line is vertical, moves from left to right, and is currently at the position as shown in the figure. Denote the enter-segment events as $ES(L_i)$, the leave-segment events as $LS(L_i)$, and the intersection events as $\cap(L_i,L_j)$.



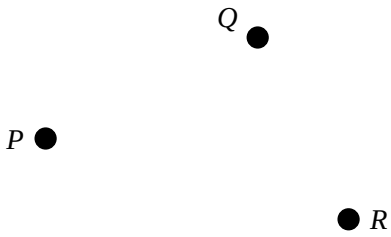
(a) List, in the order of occurrence, all the events that the sweep line encounters as it completes its sweep from the current position. **(4)**

Solution $LS(L_4)$, $ES(L_1)$, $ES(L_3)$, $\cap(L_2,L_5)$, $LS(L_5)$, $\cap(L_1,L_2)$, $\cap(L_2,L_3)$, $LS(L_1)$, $LS(L_2)$, $LS(L_3)$.

(b) Which of these events are stored in the event queue at the current position of the sweep line? Assume that the event queue is initialized by all enter-segment and leave-segment events. **(3)**

Solution $LS(L_4)$, $ES(L_1)$, $ES(L_3)$, $\cap(L_2,L_5)$, $LS(L_5)$, $LS(L_1)$, $LS(L_2)$, $LS(L_3)$.

3. We run Fortune's line-sweep algorithm to compute the Voronoi diagram of the three points P,Q,R as given in the following figure. Assume that the sweep line is horizontal, and moves from top to bottom.



(a) In what order do the events (site and circle) occur in the execution of the algorithm? **(2)**

Solution Site event for Q, site event for P, site event for R, circle event for P,Q,R.

(b) Describe the beach line just before and just after the circle event. Do not draw the beach line. Specify only the sequence of sites contributing parabolic arcs to the beach line from left to right. If some site(s) contribute(s) multiple arcs on the beach line, list the site(s) the required number(s) of times. **(4)**

Solution Before the circle event: Q,P,Q,R,Q.
After the circle event: Q,P,R,Q.

(c) Is the circle event a fork-in event or a fork-out event? **(1)**

Solution Fork-in

3. We run Fortune's line-sweep algorithm to compute the Voronoi diagram of the three points P,Q,R as given in the following figure. Assume that the sweep line is horizontal, and moves from top to bottom.

(a) In what order do the events (site and circle) occur in the execution of the algorithm? **(2)**

Solution Site event for P, site event for R, site event for Q, circle event for P,Q,R.

(b) Describe the beach line just before and just after the circle event. Do not draw the beach line. Specify only the sequence of sites contributing parabolic arcs to the beach line from left to right. If some site(s) contribute(s) multiple arcs on the beach line, list the site(s) the required number(s) of times. **(4)**

Solution Before the circle event: P,Q,P,R,P.
After the circle event: P,Q,R,P.

(c) Is the circle event a fork-in event or a fork-out event? **(1)**

Solution Fork-out

4. Let S be a set of n points (in general position) in the two-dimensional plane. We want to reorder ***all*** the points of S to a list $P_1,P_2,\ldots,P_n$ in such a way that

(1) no segment P_iP_{i+1} intersects with any other segment P_jP_{j+1} except perhaps at the endpoints, and
(2) $P_iP_{i+1}P_{i+2}$ is a right turn for all $i = 1,2,\ldots,n-2$.

Propose an efficient algorithm to solve this problem. What is the time complexity of your algorithm? **(8 + 2)**

Solution Assume that our convex-hull algorithm outputs the corners of the output hull in a clockwise fashion starting from any vertex.

First algorithm: Let $L_1,L_2,\ldots,L_k$ be the onion layers of S. We need to list the points of the layers with some caution. We can start the listing from any point of L_1, and stop just before completing the polygon. Let Q be the

last point listed. From Q, compute the appropriate tangent to L_2 (the tangent having all points of L_2 on or to the right of it is appropriate). Then, list the points of L_2 in clockwise fashion starting from the point of tangency, and stop just before coming back to that point. Again, draw the appropriate tangent from the last point listed to the next inner layer L_3. Start listing points of L_3 starting from the point of tangency. Repeat until all the layers are considered.

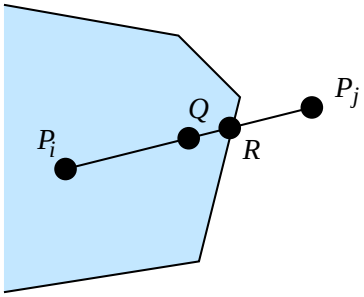
Second algorithm: Compute $L = \mathrm{CH}(S) = (Q_1, Q_2, \ldots, Q_h)$. List $Q_1, Q_2, \ldots, Q_h$. Remove $Q_1, Q_2, \ldots, Q_{h-1}$ from S. Unlike the previous algorithm, we keep Q_h in S. We then again compute $\mathrm{CH}(S)$. It is easy to see that Q_h must belong to this second hull. List the points of the second hull starting immediately after Q_h and stopping immediately before Q_h. Then, remove from S the point Q_h and all the points on the second hull except the one listed last. This process is repeated until S contains only one point.

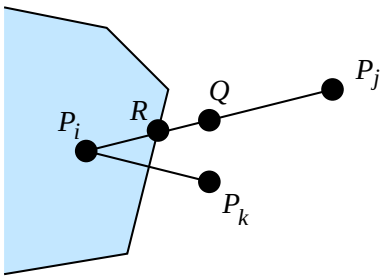
The running time of both these algorithms is $\mathrm{O}(n^2)$ if we use Jarvis march for each convex-hull computation.

5. A forest is inhabited by n prides of lions. The i-th pride lives in a den at location $P_i = (x_i, y_i)$, and requires a circular area of a fixed radius r around P_i for hunting. A pride is called safe if its hunting area does not intersect with the hunting area of any other pride. A survey gives the forest ranger the den locations x_i, y_i, and the hunting radius r (same for all prides). The ranger wants to figure out which prides are safe.

Let S be the set of the n den locations (assumed to be in general position). Propose an algorithm that, given $\mathrm{Vor}(S)$, solves the ranger's problem in $\mathrm{O}(n)$ time. Assume that each segment (finite or semi-infinite) of $\mathrm{Vor}(S)$ stores the indices i, j such that the segment belongs to the perpendicular bisector of P_iP_j. Justify the correctness of your algorithm, and that its running time is $\mathrm{O}(n)$. **(10)**

Solution The i-th pride is safe if and only if the den (other than P_i) closest to P_i is at a distance $\geqslant 2r$ from P_i. In view of this observation, it suffices to compute for each i, the den P_j ($j \neq i$) closest to P_i. We first claim that in this case $\mathrm{VCell}(P_i)$ and $\mathrm{VCell}(P_j)$ share an edge. More specifically, we show that the midpoint Q of the segment P_iP_j belongs to a Voronoi edge. Suppose not. Since P_j cannot stay inside $\mathrm{VCell}(P_i)$, the segment P_iP_j must intersect some edge of $\mathrm{VCell}(P_i)$, say, at a point R (see the figure below). We start with the assumption that $Q \neq R$.





If R is on the open segment QP_j, then the open segment QR is closer to P_j than to P_i, and so cannot belong to $\mathrm{VCell}(P_i)$ (see the left part of the figure). So consider the case that R is on the open segment P_iQ (see the right part of the figure). Let the edge of $\mathrm{VCell}(P_i)$ that intersects with P_iP_j belong to the perpendicular bisector of P_iP_k. But then $|P_iP_k| \leqslant 2|P_iR| < 2|P_iQ| = |P_iP_j|$, a contradiction to the fact that P_j is the closest neighbor of P_i.

This assertion also follows directly from a result proved in the tutorials. Consider the circle C with center at Q and passing through P_i and P_j. Since P_j is the neighbor closest to P_i, and the (unique) point on or inside C and most distant from P_i is its diametrically opposite point P_j, the circle does not contain any other den on or inside it. Therefore $\mathrm{VCell}(P_i)$ and $\mathrm{VCell}(P_j)$ share an edge.

Given $\mathrm{Vor}(S)$, one can locate the closest neighbor of each P_i. Since $\mathrm{Vor}(S)$ is of size $\Theta(n)$, all these closest neighbors can be found in $\mathrm{O}(n)$ time.

6. Let $\Phi(x_1, x_2, \ldots, x_n)$ be a Boolean formula in the conjunctive normal form (CNF). We say that Φ is all-but-one satisfiable if there is a truth assignment of the variables for which all except exactly one of the clauses of Φ evaluate to true. By AB1SAT, we denote the problem of deciding whether the given CNF formula Φ is all-but-one satisfiable.

(a) Prove that AB1SAT is in NP. **(4)**

Solution An appropriate truth assignment of the variables is a succinct certificate for an instance in Accept(AB1SAT). The CNF formula can be evaluated in time linear in the size of Φ plus n. It is an easy matter to check how many clauses are satisfied for the given truth assignment.

(b) Prove that AB1SAT is NP-complete. (**Hint:** Use reduction from CNFSAT.) **(6)**

Solution We reduce CNFSAT to AB1SAT. Let $\Phi(x_1, x_2, \ldots, x_n)$ be an instance for CNFSAT. We introduce a new variable y, and generate the CNF formula $\hat{\Phi}(x_1, x_2, \ldots, x_n, y) = \Phi(x_1, x_2, \ldots, x_n) \wedge y \wedge y'$. We show that Φ is satisfiable if and only if $\hat{\Phi}$ is all-but-one satisfiable. If Φ is satisfiable by some truth assignment of $x_1, x_2, \ldots, x_n$, then the same truth assignment along with any truth assignment for y all-but-one satisfies $\hat{\Phi}$. On the other hand, if Φ is not satisfiable, then for each truth assignment of $x_1, x_2, \ldots, x_n$, at least one clause of Φ evaluates to false. Moreover, one of the clauses y and y' must be false too. So $\hat{\Phi}$ cannot be all-but-one satisfied.

This reduction shows that AB1SAT is NP-hard. By Part (a), it is NP-complete too.