Indian Institute of Technology Kharagpur
Computer Science and Engineering

CS 60064 | Computational Geometry | Spring 2021

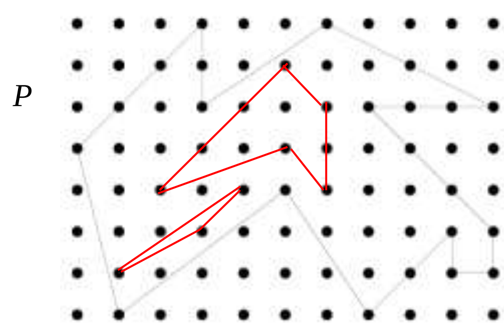
Date: 12.03.2021 **Total marks** = 100
Online Test-02 Credit: 20% **Time:** 11:00 am -12:30 PM

Instructions (Read carefully)

- **A.** This is an OPEN-BOOK/OPEN-NOTES, online test. Answer all questions. This question paper has *three* pages. While describing algorithms, sketch the steps only, **no pseudo-code is needed**. In solving a problem, unless it is particularly asked for, if you have to use a known algorithm, just refer to it without giving further details. For example, if you need to use Graham's scan in your solution, just write "Call Graham's scan", leading to time complexity $O(n\log n)$.
- **B.** **Submission of answers:** Please create a pdf file including **your name, roll-number** and submit it to the CSE Moodle Page by 12:45 PM, Friday, 12 March 2021.

---

1. (10 points)
Consider a uniform square grid with unit spacing between two consecutive horizontal or vertical pairs. A polygon P with two holes (shown in red) is drawn all of whose vertices coincide with grid points as in the figure below. Calculate the area of P **excluding** that of the holes.

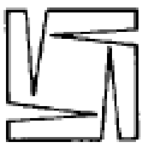


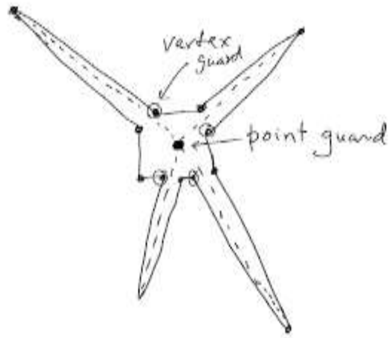
Solution:
By Pick's Theorem, Area of a lattice polygon = # interior points + (#boundary points/2) - 1;
Thus, area of the surrounding polygon P = 30 + (22/2) -1 = 30 + 11 – 1 = 40 square unit;
Area of big red hole = 2 + 4 - 1 = 5 square unit; Area of red triangle = ½ square unit;
Hence, area of P excluding holes = 40 – 5.5 = 34.5 square units.

2. (10 points)
(a) A vertex v of a polygon is said to be reflex if the internal angle at v is > π. Let P be a simple polygon with n vertices and assume that P have $r \geq 4$ reflex vertices. Show that r vertex guards are sometimes necessary and always sufficient to see the interior of P.

Solution:
Sufficiency – By draw bisectors through each reflex angle, P can be partitioned into $r + 1$ convex pieces. A vertex put on a reflex vertex can see the two adjacent convex pieces. Hence, r vertex guards are sufficient.

Necessity – The following example below (on the left) shows that four vertex guards are necessary in a polygon that has four reflex vertices:

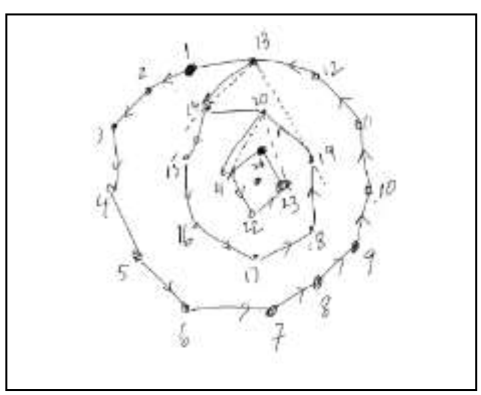




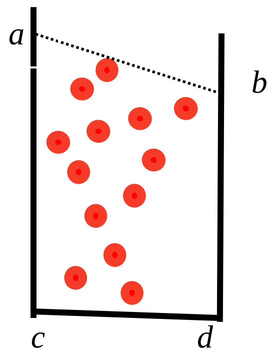
(b) A guard is said to be a *point guard* if it can be placed anywhere in a polygon (i.e., not necessarily on a vertex or on an edge). Construct a simple polygon that requires at least four vertex guards for boundary visibility but a single point guard for area visibility. (5 + 5)
Solution (b)**:** See an example above (figure on the right).

3. (15 points)
(a) You are given n points, $n \geq 3$, being randomly placed in the plane. Assume that no three of them are collinear. Show that it is possible to connect all these points using a single non-crossing polygonal chain C and label them, such that every three consecutive points on C experience only a left-turn. Sketch an algorithm for constructing such a chain and discuss its time complexity.

Solution: Construct convex layers C1, C2, … , Ck (layers of convex hull) using Chazelle's $O(n \log n)$-time algorithm, or using repeated Jarvis March in $O(n^2)$-time. Assume that C1 denotes the outermost hull and Ck is the innermost hull. Traverse a chain (path) along the boundary of C1 in counterclockwise direction to visit all its vertices, and from the last vertex, draw two tangents to the next inner hull (takes $O(\log n)$ time for drawing tangents). Take the tangent that ensures continuation of the chain with left-turn in the next inner hull. Repeat until the innermost hull is covered. See the figure below for illustration. Overall, it can be finished in $O(n\log n)$-time or in $O(n^2)$-time depending on which algorithm is being used for constructing the layers.



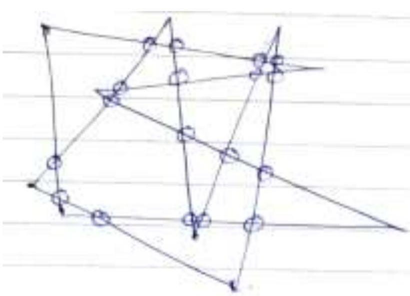
4. (15 points)
A U-shaped box whose top-side is open, contains n red dots. The positions of the dots (x- and y-coordinates) are known. There is a straight lid $\{a, b\}$ which may be moved up and down or rotated around the axis perpendicular to the paper. Design an efficient algorithm to determine the position of the lid $\{a, b\}$ such that all dots are contained within the trapezoid $X = \{a, b, d, c\}$, and $Area(X)$ is minimized.



Solution: Find the red dot r that has max. y-coordinate, i.e., the topmost one. The lid $\{a, b\}$ must touch that. If r appears at the mid-point of $\{a, b\}$, then, this will define the trapezoid X with minimum area, regardless of its tilt *w.r.t.* the horizontal. Else, if the portion on the left (right) is longer, rotate the lid CCW (CW) with pivot at r, till it touches the next red dot, and repeat this to decrease area if similar cases arise before we reach the median. In both cases, X must touch an edge of the upper chain of the convex hull bounded between the two vertical walls of X. Hence, this would need $O(n \log n)$-time for the hull, and $O(n)$ for finding areas for X with respect to those edges of the hull, so a total of $O(n \log n)$-time.

5. (15 points)
(a) Let P_1 and P_2 be two simple polygons comprising five vertices each. Construct P_1 and P_2 in such a way that the number of intersections between them is maximized.
(b) Given two convex polygons P_1 and P_2 with m and n vertices, respectively, sketch an $O(m + n)$-time algorithm to construct the convex hull of $P_1 \cup P_2$. (7 + 8)

Solution (a): Maximum #intersections = 18
See below

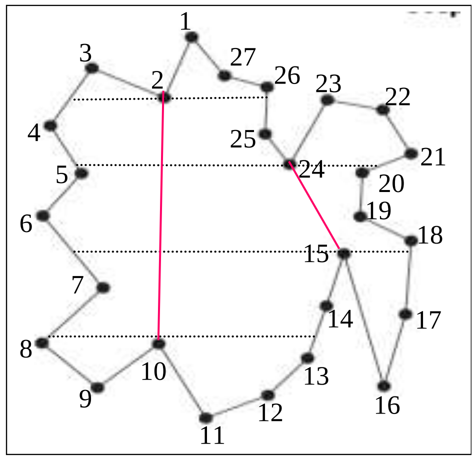


Solution (b): We are given two convex polygons P_1 and P_2 with m and n vertices. Since polygons are given as ordered sequence of vertices, we can extract upper and lower hulls of two polygons and merge the lists to form a sorted sequence in $O(m + n)$ time. Then, by plane sweep, we can construct their intersection points in $O(m + n)$ time because #intersections between two convex polygon is also $O(m + n)$. Next, by traversing along the boundary and switching contours at every intersection point, we can construct their union-polygon $P_1 \cup P_2$ in $O(m + n)$ time. Now, by running Melkman's algorithm, we can construct the convex hull in $O(m + n)$ time. (Or, on $P_1 \cup P_2$, Graham's scan can be run *without sorting*, since the sequence of vertices around the boundary of $P_1 \cup P_2$ is known at the time of its construction.)
*If there is no intersection point, check whether they are disjoint or one includes the other. This can be checked in linear time.
Two cases now:
1. Disjoint case: Draw two outer tangents - we are done within linear time;
2. Inclusive case: Nothing should be done, output the larger one.

6. (10 points)

(a) In the polygon *P* shown below, consider a horizontal sweep-line moving from top-to-bottom. Identify the merge and split vertices in *P*, and for each of them show the helper of the corresponding left bounding edge.

(b) Partition *P* into minimum number of *y*-monotone polygons and justify why your solution is indeed *minimum* (show the solution only, no need to show algorithmic steps). (5 + 5)



Solution (a): Merge vertices: 2, 24; Split vertices: 10, 15

Left-bounding edge(2) = (3,4); Helper: 2

Left-bounding edge(24) = (4,5); Helper: 24

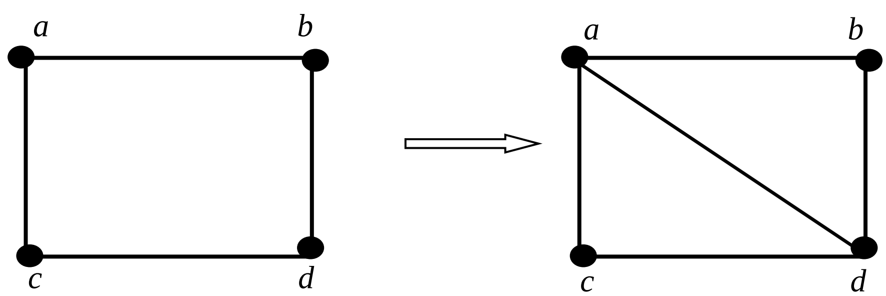
Left-bounding edge(10) = (7,8); Helper: 14

Left-bounding edge(15) = (6,7); Helper: 18

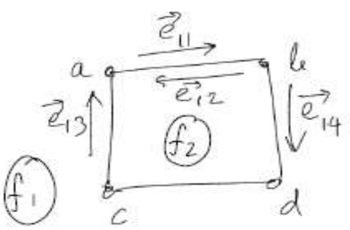
Solution (b): We can add two diagonals (2, 10) and (24, 15) as shown in red to partition the polygon into three y-monotone pieces. This is indeed *minimum* because we have three top (start) vertices (1, 3, 23) and three bottom (end) vertices (9, 11, 16), since every *y*-monotone polygon must have *exactly* one top and one bottom vertex.

7. (10 points)

Consider the planar sub-division shown below on the left, and in the next step, it will be modified as shown on the right. For both initial and final configurations, write the DCEL descriptions for one vertex (say, *a*), one edge (say, *ab*), and one inner face only {e.g. (*abcd*) for the initial configuration and (*acd*) for the final configuration}.



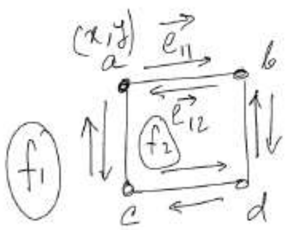
Solution



DCEL representation for a half-edge $\vec{e}_{11}$ (connecting a & b) :

half-edge	origin	twin	incident face	next	prev.
$\vec{e}_{11}$	a	$\vec{e}_{12}$	f_2	$\vec{e}_{14}$	$\vec{e}_{13}$

(note: suffix of a half-edge, i.e. $\vec{e}_{11}$ is arbitrary) .

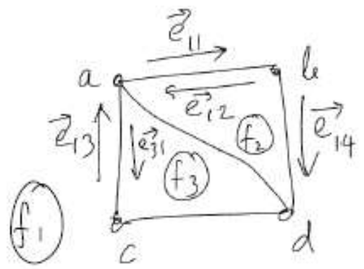


DCEL representation for vertex $v(x,y)$

v	(x,y)	$\vec{e}_{11}$
label	co-ordinates	incident edge (outgoing)

Face: abcd

f_2	$\vec{e}_{12}$	ϕ
label	outer component	inner component



DCEL representation for a half-edge $\vec{e}_{11}$ (connecting a & b):

In the second configuration, DCEL representation for vertex a and half-edge $\vec{e}_{11}$ remain the same as before. For face "acd", it would be:

f_3 $\quad \vec{e}_{31}$ (outer component) $\quad \Phi$ (Inner component)

8. (15 points)

(a) Let C denote a family of n unit-circles on the 2D plane as shown in the figure above. A convex hull of C is a convex polygon that tightly surrounds C. Some of these circles may be intersecting. Sketch an $O(n \log n)$-time algorithm for constructing the convex hull of C.

(b) In Chan's convex hull algorithm, n given points are first partitioned into r groups, each of size m. How is the correct value of m chosen so that the time complexity of constructing the hull becomes $O(n \log h)$, where h is the number of hull vertices? When does the algorithm terminate? (10 + 5)

Solution (a): The convex hull will comprise some circular arcs and straight segments, which are tangents. Alternatively, we can bound it with straight segments only by extending the tangents as shown in the figure below. In both cases, since, the circles are of equal size, the circles that will lie on the boundary of the hull if and only if their centers lie on the convex hull of all centers. Note that in $O(n \log n)$ time, we can construct the hull for the centers. Next, we can construct tangents between two circles that are consecutive around the hull in overall $O(n)$ time, because a tangent between two circles can be constructed in $O(1)$ time. Hence, overall the convex hull of C can be constructed in $O(n \log n)$ time.

Solution (b): The algorithm terminates when $m \geq h$. The smallest value of can be chosen using a very fast iterative technique as shown below.

Hull(P) :

(1) For $t = 1, 2, \ldots$ do:
  (a) Let $m = \min(2^{2^t}, n)$.
  (b) Invoke PartialHull(P, m), returning the result in L.
  (c) If $L \neq$ "try again" then return L.

PartialHull(P, m) :

(1) Let $r = \lceil n/m \rceil$. Partition P into disjoint subsets $P_1, P_2, \ldots, P_r$, each of size at most m.
(2) For $i = 1$ to r do:
  (a) Compute Hull(P_i) using Graham's scan and store the vertices in an ordered array.
(3) Let $p_0 = (-\infty, 0)$ and let p_1 be the bottommost point of P.
(4) For $k = 1$ to m do:
  (a) For $i = 1$ to r do:
    • Compute point $q_i \in P_i$ that maximizes the angle $\angle p_{k-1} p_k q_i$.
  (b) Let p_{k+1} be the point $q \in \{q_1, \ldots, q_r\}$ that maximizes the angle $\angle p_{k-1} p_k q$.
  (c) If $p_{k+1} = p_1$ then return $\langle p_1, \ldots, p_k \rangle$.
(5) Return "m was too small, try again."