

## COMPUTATIONAL GEOMETRY: Practice Midterm

(1). [15 points] Without appealing to Fisk's proof of the Art Gallery Theorem, give a simple argument to show that a simple polygon  $P$  having  $n$  vertices can always be guarded using at most  $n - 2$  strictly interior point guards.

(2). [20 points] In the predicate `InCone` below, I have left some blanks. Fill them in to make the predicate correct. (Reminder: In C, "!" means "not", "&&" means "and", "||" means "or", "!=" means "not equal to".)

```

/*-----
Returns TRUE iff the diagonal (a,b) is strictly internal to the
polygon in the neighborhood of the a endpoint.
-----*/
bool InCone( tVertex a, tVertex b )
{
    tVertex a0,a1;      /* a0,a,a1 are consecutive vertices. */

    a1 = a->next;
    a0 = a->prev;

    /* If a is a convex vertex ... */
    if( LeftOn( a->v, a1->v, a0->v ) )
        return Left( a->v, b->v, a0->v )
            && Left(_____);

    /* Else a is reflex: */
    return !( LeftOn(_____ )
            && LeftOn( b->v, a->v, a0->v ) );
}

```

(3). [20 points] Let  $P$  be a simple polygon having  $n$  vertices. For each of the computations below indicate how efficiently one can perform the calculation, in terms of  $O(\dots)$  notation (e.g.,  $O(n)$ ,  $O(\log n)$ ,  $O(n^2)$ ,  $O(n \log n)$ ). Try to give the best (lowest) upper bound possible.

- (a). Compute the number of reflex vertices of  $P$ .
- (b). Compute the convex hull of  $P$  (giving as output the vertices of the hull in clockwise order around the hull).
- (c). Determine one ear of  $P$  (any ear is OK).
- (d). Triangulate  $P$ , giving a valid set of diagonals in any order.
- (e). Triangulate (all of) the pockets of  $P$ , giving the set of diagonals in any order.

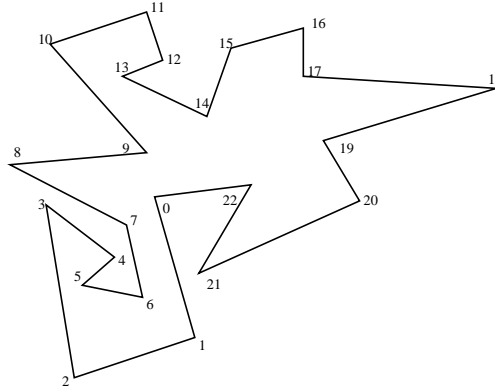
(4). [20 points] The Art Gallery Theorem tells us in general the number of (stationary) point guards that are sufficient and sometimes necessary to guard the interior of a simple polygon  $P$  having  $n$  vertices. If  $P$  has special structure, though, there is some hope that the guard number may be less. For each class of simple polygon below, state the best bounds you can on the worst-case guard number,  $G(n)$ , among  $n$ -gons of the class. In each case, give a very brief justification for your answer.

- (a).  $P$  is a simple polygon (not of any special class)

- (b).  $P$  has exactly one reflex vertex
- (c).  $P$  is convex
- (d).  $P$  is  $y$ -monotone
- (e).  $P$  has a single pocket

(5). [25 points] For the simple polygon  $P$  below, do the following:

- (a). [3 points] Show the horizontal trapezoidalization of  $P$ .
- (b). [3 points] Show the decomposition into monotone polygons given by the algorithm of Section 2.2.
- (c). [3 points] Show a triangulation of  $P$  that can be obtained from triangulating the monotone polygons.
- (d). [8 points] By inspection, obtain the *point* guard number for  $P$ , allowing guards to be placed at *any* point (interior or boundary) of the polygon. Justify your answer! (Give an argument that fewer guards cannot suffice.)
- (e). [8 points] Assume that we execute Melkman's convex hull algorithm on the vertices of  $P$  in the order  $v_0, v_1, v_2, v_3$ , etc. (In the figure, I label  $v_i$  with " $i$ ".) Show the deque, indicating the "top"  $d_t$  and "bottom"  $d_b$  at the instant just after having computed the hull of the first 8 vertices ( $v_0-v_7$ ) and also just after the first 9 vertices ( $v_0-v_8$ ).

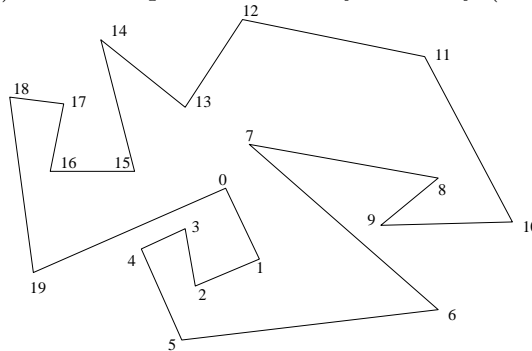


AMS 345/CSE 355

Joe Mitchell

## COMPUTATIONAL GEOMETRY: Another Practice Midterm

(1). [10 points] By inspection, obtain the *point* guard number for  $P$ , allowing guards to be placed at *any* point (interior or boundary) of the polygon. Justify your answer! (Give an argument that fewer guards cannot suffice.) Here we speak of ordinary visibility (not clear visibility).



(2). [20 points] In the predicate `IntersectProp` below, I have left some blanks. Fill them in to make the predicate correct. (Reminder: In C, “!” means “not”, “&&” means “and”, “||” means “or”, “!=” means “not equal to”. `Xor(A, B)` is the exclusive “or” of  $A$  and  $B$ : it is true iff exactly one of  $A$  or  $B$  is true.)

```

/*-----
Returns true iff ab properly intersects cd: they share
a point interior to both segments. The properness of the
intersection is ensured by using strict leftness.
-----*/
bool    IntersectProp( tPointi a, tPointi b, tPointi c, tPointi d )
{
    /* Eliminate improper cases. */
    if (
        Collinear(a,b,c) ||
        Collinear(a,b,d) ||

        Collinear(_____) ||

        Collinear(_____)
    )
        return FALSE;

    return
        Xor( Left(a,b,c), _____ )

        && Xor( _____, Left(c,d,b) );
}

```

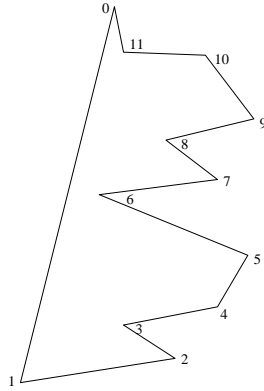
(3). [10 points] Let  $P$  be a simple polygon in the plane. Prove or disprove: If a guard is placed at every *convex* vertex of  $P$ , this set of guards sees all of  $P$ . (Here we speak of ordinary visibility, not clear visibility.)

(4). [10 points] Let  $P$  be a simple polygon with  $n$  vertices. Suppose that I give you a piece of code, `RayShoot(v,w)` that is able to answer a “ray shooting query” in  $P$  in time  $O(\log n)$ : Specifically, `RayShoot(v,w)` takes a vertex  $v$  of  $P$  and any other point  $w \neq v$  and determines the first point (closest to  $v$ ) of the boundary of  $P$  where the ray  $vw$  (with endpoint  $v$ , passing through  $w$ ) intersects the boundary. In other words, the code computes the point of the boundary of  $P$  where a bullet would hit if it is shot from  $v$  towards  $w$ . (In fact, one can do ray-shooting in  $O(\log n)$  time, although the algorithms are nontrivial.)

Explain briefly how you could use this code as part of an algorithm to triangulate  $P$  in worst-case time  $O(n \log n)$ .

(5). [7 points] Give an example of a generic class (i.e., a family of examples that generalizes to arbitrarily large values of  $n$ ) of a simple  $n$ -gon  $P$  for which  $P$  has only one possible triangulation.

(6). [8 points] For the monotone mountain shown below, show the (unique) triangulation that is given by the algorithm we presented in class for triangulating monotone mountains.



Which diagonal is the FIFTH to be added during the algorithm?

(7). [20 points] Let  $S$  be a set of  $n$  points in the plane, given in arbitrary order. For each of the computations below indicate how efficiently one can perform the calculation, in terms of  $O(\dots)$  notation (e.g.,  $O(n)$ ,  $O(\log n)$ ,  $O(n^2)$ ,  $O(n \log n)$ ). Try to give the best (lowest) upper bound possible in terms of  $n$  and  $h$  (the number of convex hull vertices). Give a brief justification of your answer! (1-2 sentences suffice)

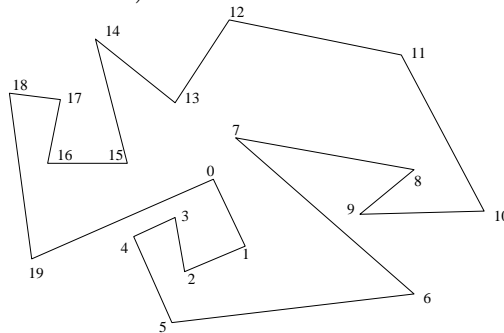
(a). Compute the convex hull,  $CH(S)$ , of  $S$  (giving as output the vertices of the hull in clockwise order around the hull).

(b). Determine if the convex hull,  $CH(S)$ , is a hexagon (6-sided polygon). (The output is simply a “yes” or “no” answer.)

(c). Determine one edge of  $CH(S)$ .

(d). Given as input the set  $S$ , compute the area of  $CH(S)$ .

(8). [8 points] For the simple polygon  $P$  below, list (in order) the first *six* diagonals that are output when `Triangulate` is executed on it (exactly as is done in Table 1.2 for the polygon shown in Figure 1.27, and as you did on HW2).



(9). [7 points] Assume that we execute Melkman’s convex hull algorithm on the vertices of the polygonal chain below, in the order  $v_0, v_1, v_2, v_3$ , etc. (In the figure, I label  $v_i$  with “ $i$ ”.) Show the deque, indicating the “top”  $d_t$  and “bottom”  $d_b$  at the instant just after having computed the hull of the first 6 vertices ( $v_0-v_5$ ), the first 7 vertices, and the first 8 vertices.

