

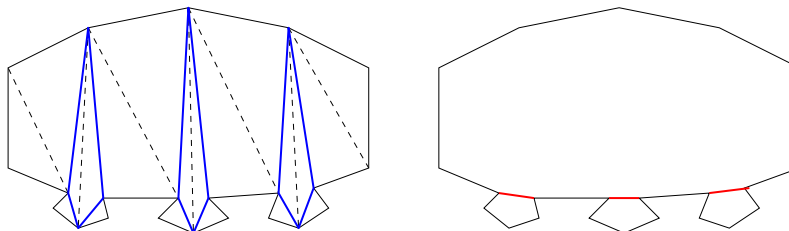
## COMPUTATIONAL GEOMETRY

### Homework Set # 4 – Solution Notes

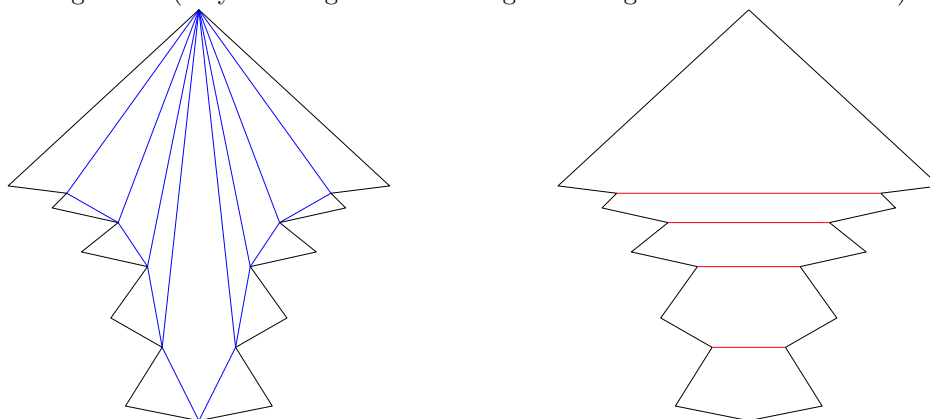
(1). [25 points] *The Hertel-Mehlhorn algorithm was shown to be within factor 4 of optimal. (See section 2.4 of O’Rourke.)*

(a). *Find a generic family of examples<sup>1</sup> of simple polygons having  $r$  reflex vertices, and a triangulation of each, such that there exists a triangulation and an order of inessential diagonal removal such that the Hertel-Mehlhorn algorithm produces a partitioning into  $2r + 1$  convex polygons, while an optimal partitioning results in only  $\lceil r/2 \rceil + 1$  convex polygons.*

There are several possible generic examples of polygons for which the Hertel-Mehlhorn algorithm yields  $2r + 1$  pieces (2 essential diagonals per reflex vertex), while an optimal convex decomposition requires only  $\lceil r/2 \rceil + 1$  pieces. One type of families of examples is shown below, where a “big” convex polygon has  $k$  convex “tabs” attached to it, creating  $r = 2k$  reflex vertices. The figure on the left shows a triangulation, with the blue solid diagonals highlighting the diagonals that survive the H-M algorithm (they are always essential, no matter what order we choose to delete inessential diagonals). (Any other triangulation that includes these blue diagonals will also result in the blue diagonals being the survivors in the H-M algorithm.) The figure on the right shows an optimal decomposition into convex pieces, using solid red diagonals.



Another class of examples is shown below. On the left is shown (in blue) the decomposition given by the Hertel-Mehlhorn algorithm (only one diagonal of the original triangulation was inessential).



(b). *Argue that if one starts with the “right” triangulation of  $P$  and deletes inessential diagonals in the “right” order, then the algorithm will in fact give an optimal decomposition into (the exact minimum number of) convex pieces.*

We start with an optimal decomposition (it must exist) and add diagonals,  $d_1, d_2, \dots, d_k$ , to complete it into a full triangulation of the polygon. Now, if we were to run Hertel-Mehlhorn on the resulting triangulated polygon, each of the diagonals  $d_i$  we added could be removed (they are inessential, since their removal leaves convex faces in the decomposition); in fact, Hertel-Mehlhorn has the option to delete them in the order  $d_k, d_{k-1}, \dots, d_1$ , returning us to the original optimal decomposition.

<sup>1</sup>i.e., a class of examples so that the size of the polygon (number of vertices) can be made to be arbitrarily large

(c). [Optional challenge (extra credit)]: Consider part (a) again. Can we find a family of examples for which any triangulation has an ordering of deletion in the Hertel-Mehlhorn algorithm that leads to a “bad” result (factor 4 away from optimal)? (One possibility would be to find a family of examples having a unique triangulation.) The point of this question is to try to separate the two places where the algorithm makes choices: the triangulation, and the deletion order of diagonals.

I am still looking for the definitive answer to this.

(2). [25 points] *Problem 4.10, page 93, BKOS.*

Here is the basic idea: A redundant half-plane  $h$  (with bounding line  $\ell$ ) fully contains the feasible region. Consider moving  $\ell$  parallel to itself, in the direction into the region  $h$ , until it contacts a vertex  $v$  of the feasible region  $\cap H$ ; let  $\ell'$  be the corresponding line through  $v$ . The vertex  $v$  has two incident edges, each corresponding to a non-redundant half-plane (say half-planes  $h'$  and  $h''$ ). The cone  $h' \cap h''$  is contained in the half-plane determined by  $\ell'$  (the side of the half-plane being the same as that of  $h$  relative to  $\ell$ ), and thus is contained in the half-plane  $h$ , which is a superset.

Now, to determine all redundant half-planes, we simply construct the feasible set  $h_1 \cap h_2 \cap \dots \cap h_n$ , which is done in  $O(n \log n)$  time by divide-and-conquer. (This was discussed briefly in class and is detailed in Section 4.2.) The non-redundant half-planes are exactly those that do *not* contribute an edge to the feasible set. (Note that a redundant half-plane may pass through a vertex  $v$  of the feasible set without defining one of the two edges of the feasible set incident on  $v$ .)

(3). [25 points] *Problem 4.15, page 93, BKOS*

My solution is below; note that the time bound is actually worst-case  $O(n)$ , not just expected  $O(n)$ , as we get using the randomized incremental algorithm of the text.

A simple polygon  $P$  is star-shaped if there exists a point  $q = (q_x, q_y) \in P$  such that the line segment  $qp$  is contained in  $P$ , for every  $p \in P$ . Equivalently,  $P$  is star-shaped if there exists a point  $q$  such that  $qp \subset P$  for every point  $p \in \partial P$ . (Can you prove this? One direction is trivial; for the other direction, note that we can extend a segment  $qp \subset P$ , for  $p \in \text{int}(P)$ , to a supersegment  $qp'$ , with  $p' \in \partial P$ , and  $qp' \subset P$ .) What does it mean for  $q$  to be able to see all of the points  $p$  on some edge,  $e$ , of  $P$ ? It means that  $q$  must lie in the halfplane,  $H_e$ , to the left of the oriented line containing  $e$  (oriented counterclockwise about  $P$ , so that the interior of  $P$  lies to the left). Clearly, this is necessary; otherwise,  $qp$  lies exterior to  $P$  for points sufficiently close to  $p \in e$ . In order to see that it is sufficient that  $q \in H_e$  for all  $e$ , consider to the contrary some point  $p \in e$ , and assume that  $q \in \cap_e H_e$  does *not* see  $p$ . Let  $p'$  be the first point of the boundary of  $P$  that is hit when going from  $p$  towards  $q$ . Then  $p'$  lies on an edge whose left side contains  $pp'$ , not  $p'q$ , contradicting the fact that  $q$  is in the intersection of all halfplanes  $H_e$ .

Thus, we must determine if there exists a point  $q \in P$  such that  $q \in \cap_e H_e$ . For this, we can use linear programming with two variables,  $q_x$  and  $q_y$ :  $\min q_y$  subject to  $q \in H_e$  for each  $e$ . (The objective function does not matter; all we are checking is to see if the feasible set is nonempty, and, if so, to return a feasible point  $q$ .)

Note: This linear-time method allows one to test if the “kernel” ( $\cap H_e$ , set of all possible locations of a single guard point) is nonempty, and, if so, to return a witness point. In fact, by advancing  $e$  around the boundary of the polygon  $P$  and being careful how to update the intersection of the halfplanes, one can *construct* the kernel in  $O(n)$  time.

(4). [25 points] *Problem 4.16, page 93, BKOS*

The problem easily maps to that of computing the intersection of half-planes, in time  $O(n \log n)$ .

(5). [25 points] *In class we sketched a dynamic programming algorithm for computing a minimum-weight (total edge length) triangulation of a simple polygon  $P$ . Suppose now that our goal is to find a triangulation of  $P$  that makes the areas of the triangles “most balanced” by maximizing the area of the smallest-area triangle. Give a DP algorithm to solve this problem. What is the running time (in big-Oh notation)?*

xxx To be filled in.