Homework 3

Vu Tuan Phong Pham

October 2019

1. (a) Let the set $n_i(x_i, y_i)$ to be the set of outward normal vectors of edges of polygon P. We need to find a direction vector $d(d_x, d_y)$ s.t:

$$\forall\ 1 \leq i \leq n, \quad d_x x_i + d_y y_i \leq 0$$

We can solve this linear programming problem using the algorithm discussed in class. We can make up an optimization function $c(1, 1)$, since we only want to report if a solution exists. The expected complexity is O(n).

(b) We can reduce sorting to this problem. We create a map $\phi : a \to z = ax - a^2$ (map each number a to the plane $z = ax - a^2$). If we view the 3-D plane from direction $(0, 1, 0)$, we should get the same kind of 2-D plane like in the argument we did in class for 2-D plane (except the y dimension is now the z dimension). The reason for this is because we map a number to a plane with no y rotation $(y = 0)$. Using the same argument on class, we can say that sorting is reduced to find all optimal solutions to a 3-dimensional linear program with n constraints, so find all optimal solutions has a lower bound of $\Omega(n \log n)$.

2. (a) Since rightmost point of P_1 is to the left of leftmost point of P_2, we can safely assume that the upper tangent line will not be a vertical line. Let the upper tangent line be $y = mx + b$.

For this line to be an upper tangent, if we take 2 arbitrary points q_1, q_2 on this line, $turn(q_1, q_2, p)$ for all $p \in P_1 \bigcup P_2$ has to be a right turn. For easier calculation, we take $q_1(0, b)$ and $q_2(1, m + b)$. Let $p(x, y) \in P_1 \bigcup P_2$, we have:
- $turn(q_1, q_2, p)$ is a right turn
- $\implies cross(p - q_1, p - q_2) \leq 0$
- $\implies (x - 0)(y - (m + b)) - (x - 1)(y - b) \leq 0$
- $\implies -xm - b + y \leq 0$

We can solve this 2-D linear programming to find a solution for (m, b), note that we need a solution such that b is minimize, and such that there are exactly 2 constraints where equality happens (since the tangent line has to pass through 2 points, 1 in P_1 and 1 in P_2). We can use the optimization function $c(1, -1)$ (in order to minimize b), and limit to find only intersection points solution in the feasible region.

(b) Considering each upright rectangle R_i was given using lower left corner (lx_i, ly_i), and upper right corner (rx_i, ry_i), then a vector $v(v_x, v_y)$ satisfy the problem iff:

$$\forall\ 1 \leq i \leq n,$$
$$lx_i - px_i \leq v_x \leq rx_i - px_i$$
$$ly_i - py_i \leq v_y \leq ry_i - py_i$$
(for (px_i, py_i) be the coordinate of point p_i)

We can solve the two 1-D linear programming problem for v_x and v_y. A vector v exist iff a solution for both v_x and v_y exist.

(c) We need to make some observations first. In order for the convex hull of P_1 to intersect P_2, there has to be a point $p_2 \in P_2$ such that p_2 is inside the interior of $conv(P_1)$. This means that if we draw a line l through p_2 (any slope, any y-intercept), then P_1 will not all be in the same half-plane with regard to l. Now we can model the problem to finding a line $l : ax + by + c = 0$ that has these properties:

- $\forall\ p(x, y) \in P_1,\ dot(v(x, y + \frac{c}{b}), n_l(a, b)) \leq 0 \iff ax + by + c \leq 0$ (this is equivalent to given a point $q(0, -\frac{c}{b})$ on l, dot product of vector pq and normal vector n_l of l is ≤ 0, so they stay on the same side of the half-plane).
- $\exists\ p(x, y) \in P_2$ s.t $ax + by + c = 0$.

We use the algorithm discussed in class. We first start by adding only constraints from P_1, and then add constraints from P_2. After we compute the feasible region of constraints in P_1, if there is a point $p(x, y)$ in P_2 s.t $ax + by + c = 0$ is not in the feasible region, then this means there is no line through that point that has P_1 on the same side of the half-plane, and we can conclude that P_1 intersect P_2. Otherwise, we conclude that they don't intersect.

(d) We can use the same observation with 2-c. Let the moving vector be $v(0, 1)$. If at some time t, the car hit the convex hull of P, then there exist a line through $(x_0, y_0 + t)$ that does not have P in 1 same half-plane. We use the same model with 2-c, if we can find a feasible region, then the car will not hit the convex hull at all. Otherwise, it will. Since we are trying to find the smallest t, we use the optimization vector $c(0, -1)$.

3. We first make some observations:

   - If a feasible region exist, then each half plane will contribute at most one edge to the polygon that bounds the feasible region (I am assuming the feasible region is bounded, if not, then we use the artificial bound).
   - Using the algorithm to find feasible region in class, if a half-plane already has an edge contributed to bounding the feasible region, and it gets removed, then that half-plane will never contribute anything to the bounding of feasible region anymore, and hence, we can mark it as redundant.

   From these observations, we can solve the problem using the algorithm discussed in class with a slight modification. When we add a half-plane H to the list, there can be 2 cases:

   - **Case 1:** The half-plane H does not change the feasible region, we can mark this half-plane as redundant
   - **Case 2:** The half-plane H changes the feasible region. We know that this half-plane contribute exactly 1 edge to the bounding of the feasible region. Let's called P, Q to be intersections of this 1 edge to the bounding of the feasible region. I claim that every half-plane H' that contributes to bounding the feasible region that satisfy:
     - edge contributed by H' is on the opposite side of half-plane H
     - edge contributed by H' does not contain either P or Q

     will be redundant. In this case, we can just mark all of H' redundant. Since each half-plane can only be marked redundant once, this extra step does not change the total time complexity of the algorithm, hence, the algorithm will still run in expected time O(n).

4. Let's consider an edge e in the polygon. Denote l_e to be the supporting line of edge e, and H_e to be the half-plane that contains the interior of the polygon in regard of line l_e (note that all the interior does

not have to be on the same half-plane of l_e, instead, we take a point in segment e, draw the normal ray of e from that point. The ray can be in 2 direction, and H_e will be in the direction of the ray first intersect the interior of the polygon).

We note that given a point P, if P is not in H_e, then there will be no ways P can see the edge e from the interior of the polygon.

From that observation, we conclude that if a polygon can be guarded by only 1 guard, the intersection of all H_e has to be non-empty. This is equivalent to finding a solution given constraints as half-plane, so it can be solved using linear programming (the algorithm we discuss in class). We can make up an optimization function (1, 1) (since we just care if a solution exist). Time complexity is O(n).