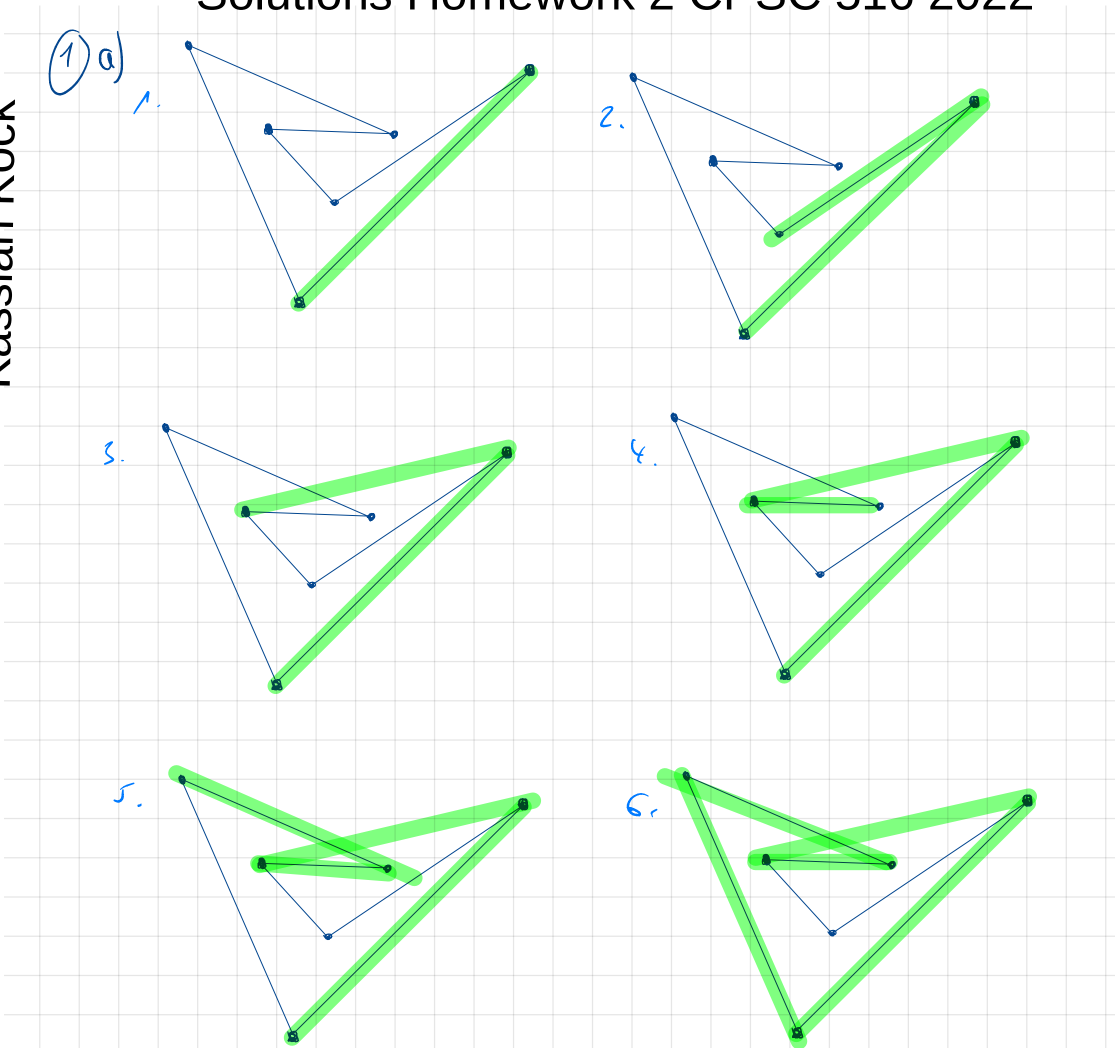
Solutions Homework 2 CPSC 516 2022

Kassian Köck



Yibo Jiao

2. **Statement 1**

   The sum of all points of P to an arbitrary L that has all points of P on one side is equal to the centroid of all points to L multiplied by n

   proof:

   Denote an arbitrary point p_i: $p_i = (x_i, y_i)$, centroid of all points of P is:$c = \left(\frac{\sum x_i}{n}, \frac{\sum y_i}{n}\right)$ With the fact that all points on the same side of L, the distance from c to $L : ax + by + c = 0$:

$$\frac{|\frac{a\sum x_i}{n} + \frac{b\sum y_i}{n} + c|}{\sqrt{a^2+b^2}} = \frac{1}{n}\sum_{i=1}^{n}\frac{|ax_i + by_i + c|}{\sqrt{a^2+b^2}}$$

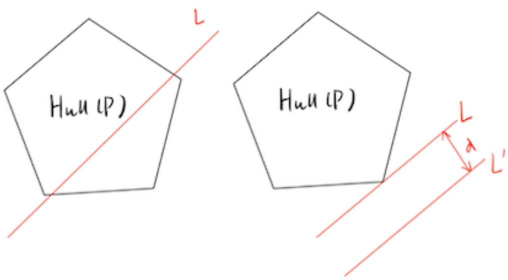
   which is the sum of distances of all points of P to L. QED.

   Statement 2

   Such L must not intersect with interior $Hull(P)$, and must pass through at least one vertex of $Hull(P)$

   proof:

   Firstly, if L intersects with interior of $Hull(P)$, then there must at least two points that are on different sides of L, proved intuitively. Secondly, if L does not intersects with interior of $Hull(P)$ and does not pass through any vertex of $Hull(P)$, the sum of distances is not minimized. Translating L in parallel until L intersects at least 1 vertex decreases the sum of distances. Denote the initial sum of distances as d, and translation distance as t, the decreased distance is $d - nt$

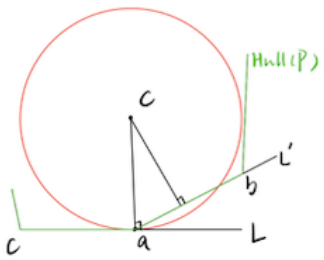


Statement 3

Such L must intersects with at least 2 vertices of $Hull(P)$, i.e., L contains at least one edge of $Hull(P)$.

<u>*proof:*</u>

We prove that any L that intersect only 1 vertex a, has larger sum of distances than at least one of the lines that passes a and another adjacent point (b) on the hull.



The centroid c lies inside $Hull(P)$ because convex hulls of points contains all possible linear combination of all points, c is one of the linear combination of points of P. Since $Hull(P)$ is convex, denote the two adjacent vertices of a as b, c, $\angle bac < 180°$, therefore, the distance from c to L' is smaller than to L because L' passes through a, and L' is not a tangent line of the circle centered at c passes a. QED.

By Statement 1,2,3, such L must not intersects any interior of $Hull(P)$ and passes at least 2 vertices of $Hull(P)$, therefore, the minimum supporting line contains one of the edge of $Hull(P)$.

Pseudo-code:

```
c ← centroid of all points in P
H ← convex hull of P
for each edge h in H do
    d ← dist(c, h)
    if d < output then
        output ← d
    end if
end for
return output
```

Dylan Brown

3. (from Zurich Exercise 4.31) Consider k convex polygons $P_1, \ldots, P_k$, for some constant $k \in \mathbb{N}$, where each polygon is given as a list of its vertices in counterclockwise orientation. Show how to construct the convex hull of $P_1 \cup \ldots \cup P_k$ in $O(n)$ time, where n is the sum of the number of vertices in P_i over all $1 \leq i \leq k$.

   Note that the following algorithm is more or less Graham's Scan, but we can cleverly avoid sorting the points.

   Algorithm. We will use a modified Graham's Scan to connect the points. Normally Graham's Scan takes $O(n \log n)$ time to sort the points, then $O(n)$ time to scan through the sorted points. I will show that I can return the next point to scan in $O(1)$ time without sorting the array, so all we need is the $O(n)$ scan time and our algorithm is $O(n)$. At the first step, we still find the minimum point across all polygons p_1 in $O(n)$ time. Next, we will use binary search to find the most clockwise point of each polygon P_i. This requires $O(\log n)$ work across k polygons, for a total setup time of $O(\log n)$.

   Now, at each stage, the next vertex to visit with be the most clockwise of these k points (assume the point comes from P_i), which we can workout in $O(k) = O(1)$ time. Now we need only workout what the new most clockwise unvisited point is for P_i, and we'll be ready for the next step.

   Since P_i is convex, the vertices which we've already visited / have angle less than some particular angle will be consecutive. Therefore, so long as we track which one's we've already visited, there are at most 2 possible candidates for the new most clockwise unvisited point: the two vertices on either end of this consecutive sequence of visited vertices. We can workout which one it is in $O(1)$ time, which is what we needed.

   So to recap: At each stage, we track the most clockwise unvisited point for each polygon, we grab the most clockwise unvisited point across all polygons, then we find a new unvisited point for the polygon we grabbed from, all in $O(1)$ time. This means that we can simply run the $O(n)$ Graham Scan without pre-sorting by replacing accessing our sorted array with this procedure, and so we have an $O(n)$ algorithm for the convex hull of k convex polygons.

Aryan Tajmir Riahi

Question 4

The problem of deciding evenly spaced arrays has two variants:

(a) Returns `YES` if and only if the array is evenly-spaced even if $x_1 = x_2 = \cdots = x_n$.

(b) Returns `YES` if the array is evenly-spaced but returns `NO` in the case that $x_1 = x_2 = \cdots = x_n$.

Notice that the only difference between the output of these two variants is in one case ($x_1 = x_2 = \cdots = x_n$) which can be checked in $\mathcal{O}(n)$. Hence, these two variants can be reduced two each other in $\mathcal{O}(n)$. So it only suffices to prove the problem for variant **(b)**.

Claim 4.1. Assume $\pi_n \subset \mathbb{R}^n$ is the set of all permutations of $1, 2, \ldots, n$. Also assume W is the set of all points in $\mathbb{R}^n$ the the variant **(b)** returns `YES` for them. Then points in π_n are pairwise disconnected within W.

Proof. Assume $a = (a_1, \ldots, a_n), b = (b_1, \ldots, b_n) \in \pi_n$ are connected within W. It means there exist a continuous function $f : [0,1] \to W$ where $f(0) = a, f(1) = b$. As $a \neq b$ the exist an inversion between them i.e. $\exists 1 \leq i \neq j \leq n$ where $a_i < a_j, b_i > b_j$. Now define a new continuous function $g : W \to \mathbb{R}$ by $g(x_1, \ldots, x_n) = x_i - x_j$. As f, g are both continuous so is $g \circ f$ and we have

$$g \circ f(0) = g(f(0)) = g(a) = a_i - a_j < 0,$$

$$g \circ f(1) = g(f(1)) = g(b) = b_i - b_j > 0.$$

By the *Intermediate value theorem* we conclude that there is some $0 < c < 1$ such that $g \circ f(c) = 0$, i.e. $f(c)_i = f(c)_j$. Now, as $f(c)$ is evenly-spaced (it should be a path in W) we should have $f(c)_1 = f(c)_2 = \cdots = f(c)_n$. Thus, $f(c) \notin W$ by definition. Hence our claim is proved by contradiction.

Now using *Ben-Or* Theorem we conclude that any algebraic decision tree that decides W (or solves the variant **(b)**) should have $\Omega(\log(|\pi_n|) - n) = \Omega(\log(n!) - n) = \Omega(n \log n)$ depth.

Dylan Brown

5. Given n real numbers $S = \langle x_1, x_2, \ldots, x_n \rangle$ and a real number g, we would like to determine if the maximum space between these numbers is g, that is, the maximum difference between the ith and $(i+1)$st smallest in S over all $1 \leq i \leq n-1$ is g. Show that any algorithm in the algebraic decision tree model requires $\Omega(n \log n)$ time to solve this problem. [Hint: Use a reduction.]

Proof. We will show this by reducing from the evenly-spaced problem given in (4). Let $S = \langle x_1, x_2, \ldots, x_n \rangle$ be n real numbers, then we want to determine if they are evenly spaced. In $\Theta(n)$ time we can find the minimum element x' and the maximum element x^*. If S is indeed the permutation of an arithmetic series, then we must have that for some $d > 0, x^* = x' + (n-1)d$, so define $d = (x^* - x')/(n-1)$, and determine if the maximum spacing of S is d.

If it is, then since we have that $x^* = x' + (n-1)d$, and there are only n elements, it must be that each of the difference constraints is tight; that is that it must be that each consecutive element is exactly d greater than the previous, otherwise the largest element could not be $(n-1)d$ greater than the smallest. This then means that our set S is evenly spaced, and we can return true.

It cannot be that the maximum spacing is less than d, because we have that $x^* = x' + (n-1)d$, so for some of the $n-1$ differences to be less than d some of the others must be greater.

If the maximum spacing is greater than d, then similarly because $x^* = x' + (n-1)d$ it must be that there is some spacing which is smaller than d, so the spacings are not all the same, our set is not evenly spaced, and we can return false.

We have given a reduction from evenly-spaced to maximum spacing with $\Theta(n)$ overhead, so since evenly-spaced is $\Omega(n \log n)$ under the algebraic decision tree model, maximum spacing is also $\Omega(n \log n)$ under the algebraic decision tree model, and we are done. $\square$