Brigham Michaelis
January 23, 2018
A00364835

Computational Geometry Assn.1

1. (15 points) Recall that a subset S of the plane is a convex set if for any two points p and q in S, the line segment p-q is also in S. Prove that the intersection of any two convex sets is still convex. This implies that the intersection of any finite number of convex sets is convex as well.

Description

This basic idea of this proof is that we are given two sets that are convex. We then take the intersection of these two sets and show that the shape of the intersection is also convex.
For this proof there two possibilities.

1. The two convex sets intersect such that one convex set is completely consumed or contained within the other set. If this is the case, then as previously stated, the intersection of these two sets is already convex, because we are starting with 2 sets that are already convex.
2. The other possibility is that there is a partial intersection of the two convex sets. This is the case that is of most interest. First we consider some convex hull. Convex hulls have the unique property that for any two points in the convex hull-area the line segment of those two points is also in the convex hull. With this definition we present an observation, because convex hulls are by definition convex, which means their sides are “rounded” or at the very worst the sides are flat. No matter how we intersect convex hulls we can never achieve a condition where the line segment connecting two points in the intersection, is not contained in the intersection of the convex hulls. We show some diagrams to illustrate this idea. Another observation is that: if there were two points in the set made by the intersection of two convex hulls the only way the corresponding line segment would not be in the intersection is if we had “teeth” going into a box wall(see illustration below). In other words there is no way that the intersection of two convex hulls can create an intersection that results in a condition where the line segment between any two points in such an intersection is not already contained by the intersection.

See accompanying handwritten pages for further discussion of the problem.

Algorithm Description
Not Required
Psuedocode
Not Required
Correctness
For this problem the definition of a convex hull is being used to give the proof for the problem. Along with the idea of making cuts with lines to convex hulls and how intersections are and can be created with cuts to a single convex set.
Time Analysis
Not applicable. There is no time analysis involved with this problem.

2. (20 points) Let P be a set of n points in the plane. Let p_0 be the leftmost point of P, i.e., the point with the smallest x-coordinate. Assume all other points of P have been sorted by polar angle in clockwise order around p_0, and the sorted list is given in the input. You may assume the sorted list is p1, p2, . . . , p_{n-1} (e.g., see Fig. 1). With p_0 and the above given sorted list, design an O(n) time algorithm to compute the convex hull of P.

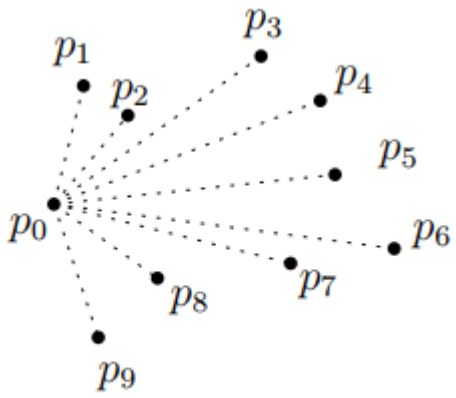

Figure 1: An example for Problem 2

Algorithm Description

We use Graham's algorithm but <u>without</u> the sorting part, put p1 and p2 onto the list L then:

for ($i=P_3$ to P_{n-1})

append P_i at the end of the list L_i (check if the last 3 points make a right turn)

while(|L| >= 3 && the last 3 points do not make a right turn)

we remove the middle point of the last 3 points from the list L

When this algorithm finishes the points in the list L will form the convex hull of P.

Psuedocode

Not Required

Correctness

This works because we are following Graham's Scan, but there is no need to sort the points. In this fashion we can obtain a runtime of O(n). Also we are using a stack for our List.

Time Analysis

This runs in O(n) time because we are using Graham's Scan but <u>without</u> the sorting portion. This way we can achieve a linear runtime.

3. . (15 points) Let P be a set of points in the plane. Prove that the pair of points in P farthest from each other must be two vertices of the convex hull of P.

Description
For this problem we are going to present a number of ways to prove the above problem.

First for the given set of points we can determine which two points have the largest distance between them. This can be done in $O(n^2)$ time.(Just compare every point with every other point and keep track of the max distance). Then run Graham's Algorithm to obtain the list of points that are on the convex hull. Now we just show that the two points with the greatest distance are also in the list produced by Graham's scan. This will work every single time.

Another thought proof for this problem.
Imagine that our convex hull was a set of co-linear points, then the obvious max-distance is found by computing the distance between the first and last points on the line.

2-D thought proof for this problem.
Say you are walking downtown with your friend and you say: "Hey did you know, that for convex hulls the pair of points that are farthest from each other are on the boundary of the convex hull?"
Your friend then replies: "No, I didn't know that, can you prove it?"
You reply: "Yes I can!"

We start with some set of points P. We then apply Graham's algorithm to find the points that constitute the convex hull, and by so doing we are able to obtain the shape. Now that we have a shape we can make a few observations.
One observation is that we can easily find the two points that are the furthest apart from each other in the set. This can be done as mentioned above by computing the distance between every point, this runs in $O(n^2)$ time, by that same action we also have the actual max-distance. With this max distance we can make a circle and use the max distance as the diameter. Now that we have a clearly defined circle we can superimpose this circle on the collection of points, and with a bit of adjusting we can achieve a state where the circle is encapsulating all the points in the set P while touching at a minimum the 2 points that are farthest from each other. Thus showing and proving that the two points that are furthest from each other **must** be on the boundary.

Here we present one more proof. Let's start with the set of points P. We will again use the above mentioned algorithm to compute the max-distance and the two points that make up the max-distance. Once we have these two points we are then going to perform a rotation on the entire data set such that the line segment connecting the two points is parallel with the y-axis. Now we can say that by definition, the highest point in the y-direction and the lowest point in the y-direction must be on the boundary. We could have done the same by rotating the set such that the line segment was parallel with the x-axis. The main idea is that max-distance can be easily computed in $O(n^2)$ time. We can then show that the two points that make up this distance are located on the boundary by subsequently running Graham's algorithm and then verifying that the points that have the largest distance between them are also encountered in the list made by Graham's algorithm.

Algorithm Description
Not Required
Psuedocode
Not Required
Correctness
This proof is given using the max-distance found in a set of points along with the use of Graham's Algorithm.
Time Analysis
$O(n^2)$ Not Required

4. . (20 points) Given k convex polygons, each represented in a standard way (e.g., by a list storing all vertices in the clockwise order), the total number of vertices of all these k convex polygons is n. Design an O(n log k) (not O(n log n)) time algorithm to compute the convex hull of all these k convex polygons.

Note: An O(n log k) time algorithm would be better than an O(n log n) time one when k is sufficiently small. For example, if k = log(n), then n*log(k) = n*log(log(n)), which is absolutely smaller than n*log(n) (i.e., if you know the small-o notation, we have n*log(log(n)) = o(n*log(n))).

Algorithm Description

Let's assume we have some number of convex polygons. We can use merge sort to merge 2 convex polygons in linear time. This happens because we can merge the upper hull and lower hull of one polygon into one sorted list in linear time. Now that we can obtain a sorted list of points for one polygon we can do the same for the rest of the polygons. From here we apply Graham's algorithm, but without the sorting part, to two convex hulls at a time. In this fashion we reduce the number of convex polygons by half. We continue in this fashion every time reducing the number of polygons by half until we have just one convex polygon which is the convex hull of all the convex polygons given in the original problem.
We can merge two sorted data sets in linear time with merge sort. Once this is done, we just apply Graham's Algorithm, but sorting is no longer required. Now we just sort two sets at a time… merge sets a and b
then c and d then e and f etc then merge like you would in merge sort.

Psuedocode
Correctness
We can merge two sorted data sets in linear time with merge sort. Once this is done, we just apply Graham's Algorithm, but sorting is no longer required. Now we just sort two sets at a time… merge sets a and b then c and d then e and f etc then merge like you would in merge sort.
Time Analysis
This algorithm runs in O(n(logk)) time because after every round of merging we have reduced the number of convex polygons by half. Also we have shown how we can achieve a linear merge time for two convex polygons.