CS 5114
Solutions to Homework Assignment 10

Meghendra Singh

April 28, 2017

[30] 1. Exercise 33.3-2. Consider a model of computation that supports addition, comparison, and multiplication and for which there is a lower bound of $\Omega(n \lg n)$ to sort n numbers. Prove that $\Omega(n \lg n)$ is a lower bound for computing, in order, the vertices of the convex hull of a set of n points in such a model.

Given that the model of computation supports addition (and subtraction, since this is just adding a negative number), comparison and multiplication and there is a lower bound of $\Omega(n \lg n)$ for sorting n numbers. We can prove that the lower bound for computing the vertices of the convex hull of a set n points using this model of computation as follows:

Let there be n distinct numbers $\{x_1, x_2, ..., x_n\}$. We can use these n numbers to create a set Q containing n points such that each point $p_i = (x_i, f(x_i))\ \forall i\ 1 \leq i \leq n$. Here the x-coordinate of a point p_i is the number x_i and the y-coordinate, $f(x)$ is a strictly convex function of x, for example x^2 or e^x. So we get, $Q = \{p_1, p_2, ..., p_n\}$. Since, the y-coordinates of the points are computed using a strictly convex function, all of the n points in Q will be included in the convex hull of Q (i.e. $CH(Q)$). Taking the n numbers to be distinct and $f(x)$ to be strictly convex also avoids the degenerate cases of the points being coincident and collinear.

Any algorithm to find the convex hull of Q (for example GRAHAM-SCAN) would result in an arrangement of all the points in Q in counterclockwise (or clockwise) order of the x-coordinate (and y-coordinate as well because of the strictly convex function). We can extract the x-coordinates of this arrangement of points (let this be A which will be an array of length n) and iterate through them once while only comparing adjacent elements. Assuming the original arrangement was in counterclockwise order, we might encounter two adjacent positions i and $i+1$ in the arrangement, such that $A[i] > A[i+1]$. We can now create a new array B of length n and fill this, first using the sub-array $A[i+1, ..., n]$ followed by the sub-array $A[1, ..., i]$. The elements of B will be the sorted n numbers. There is also a possibility that we might not encounter two adjacent positions i and $i+1$ in the arrangement, such that $A[i] > A[i+1]$, in which case the elements in A are already sorted.

Consider the example when $n = 6$ and the numbers are: $[3, 2, -1, 0, 1, -2]$, let $f(x) = x^2$ then the set Q for these will be $\{(3,9), (2,4), (-1,1), (0,0), (1,1), (-2,4)\}$. One possible convex-hull (say counterclockwise arrangement, returned by a convex hull algorithm) for these points can be $\{(-2,4), (-1,1), (0,0), (1,1), (2,4)(3,9)\}$, the extracted x-coordinates will be $A_1 = [-2, -1, 0, 1, 2, 3]$, which are the original numbers sorted in increasing order. Another possible convex-hull for these points can be $\{(2,4), (3,9), (-2,4), (-1,1), (0,0)(1,1)\}$, the extracted x-coordinates will be $A_2 = [2, 3, -2, -1, 0, 1]$. In this case, while iterating A_2, we encounter $A[2] > A[3]$ (i.e. for $i = 2$ we get $3 > -2$), so we create S with $S[1, ..., n-i] = A[i+1, ..., n]$ and $S[n-i+1, ..., n] = A[1, ..., i]$, we get $S = [-2, -1, 0, 1, 2, 3]$ and once again we have the original numbers sorted in increasing order. We can apply similar reasoning to obtain the sorted numbers, if the arrangement of the points returned by the convex hull algorithm was clockwise.

Hence, we have shown that the arrangement of vertices of the convex hull, returned by any algorithm for finding the convex hull of n points, can be used to obtain the sorted order of n numbers $\{x_1, ..., x_n\}$ in linear time while only using the comparison operation (which is allowed by our model of computation). Since the lower bound of sorting the n numbers is $\Omega(n \lg n)$ in our model of computation, this implies that the lower bound for finding convex hull of n points should also be $\Omega(n \lg n)$.

---

[30] 2. Problem 33-3. ***Ghostbusters and ghosts*** A group of n Ghostbusters is battling n ghosts. Each Ghostbuster carries a proton pack, which shoots a stream at a ghost, eradicating it. A stream goes in a straight line and terminates when it hits the ghost. The Ghostbusters decide upon the following strategy. They will pair off with the ghosts, forming n Ghostbuster-ghost pairs, and then simultaneously each Ghostbuster will shoot a stream at his chosen ghost. As we all know, it is very dangerous to let streams cross, and so the Ghostbusters must choose pairings for which no streams will cross.

Assume that the position of each Ghostbuster and each ghost is a fixed point in the plane and that no three positions are collinear.

a. Argue that there exists a line passing through one Ghostbuster and one ghost such that the number of Ghostbusters on one side of the line equals the number of ghosts on the same side. Describe how to find such a line in $O(n \lg n)$ time.

b. Give an $O(n^2 \lg n)$-time algorithm to pair Ghostbusters with ghosts in such a way that no streams cross.

Given that the position of each Ghostbuster and each ghost is a fixed point in the plane and no three positions (of either Ghostbusters or ghosts) are collinear.

a. Since the Ghostbusters and ghosts are fixed points in a plane, lets assume that the total number of these points is n (there will be $\frac{n}{2}$ ghosts and Ghostbusters). Assuming that the positions (of either Ghostbusters or ghosts) are all distinct, i.e. there are no co-incident points, we can define the set Q of these n points as $Q = \{p_1, p_2, ..., p_n\}$. Here, each point p_i represents either a Ghostbuster or a ghost. If we compute the convex hull of Q (i.e. $CH(Q)$), we can get one of three possible cases, for the set of vertices of $CH(Q)$:

  1. At-least 1 vertex is a ghost and rest are Ghostbusters, or conversely, at-least 1 vertex is a Ghostbuster and the rest are ghosts.

  2. There are more than 1 vertices which are ghosts and more than 1 vertices which are Ghostbusters.

  3. All the vertices are Ghostbusters, or conversely all the vertices are ghosts.

Figure 1 shows an example for each of the three possible cases, for $n = 12$, i.e. 6 ghosts (shown as) and 6 Ghostbusters (shown as).

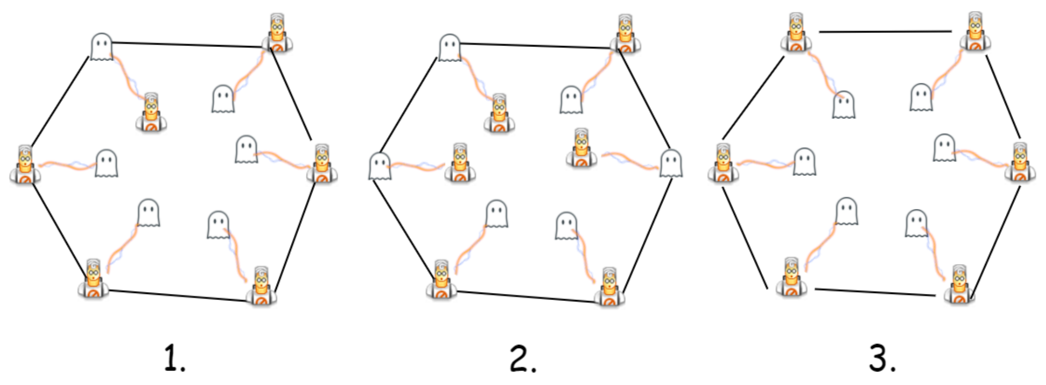

Figure 1: Possible vertices of the convex hull for $n = 12$ points (6 Ghostbuster-ghost pairs).

For cases ***1.*** and ***2.***, we can simply take an edge of $CH(Q)$ which contains a Ghostbuster-ghost pair and extend it on both sides to get a line l. As l lies on an edge of the convex hull of Q, we can be sure that there will be 0 Ghostbuster-ghost pairs on one side of l, and $(\frac{n-2}{2})$ Ghostbuster-ghost pairs on the other side of l. Since, l also passes through one Ghostbuster-ghost pair it is our desired line, for which the number of Ghostbusters on one side equals the number of ghosts on the same side (i.e. there will be 0 ghosts and 0 Ghostbusters on one side of l and $(n-1)$ ghosts and $(n-1)$ Ghostbusters on the other side of l).

For case ***3.***, we have all Ghostbusters (or ghosts) as vertices of $CH(Q)$. If we pick an arbitrary edge e on the convex hull we will have two Ghostbusters at it's ends. Let one of these Ghostbusters be u and the other be v, also let the other adjacent vertex of u on $CH(Q)$ be t. Now if we draw a line l by extending e on both sides, l would pass through two Ghostbusters (i.e. points u and v). If we keep u fixed and move l towards the interior of the convex hull, we will keep on encountering points which will either be a ghost or a Ghostbuster. Since, the total number of Ghostbusters and ghosts are equal (i.e $\frac{n}{2}$), if we keep on moving l towards the interior we will surely reach a point where the number of ghosts encountered becomes equal to the number of Ghostbusters encountered (the number of Ghostbusters encountered is inclusive of u and v). Also, this point will be reached before l aligns with the edge connecting u and t, and this point will surely be a ghost (as we started with 2 Ghostbusters i.e. u and v). At the point where we get an equal number of Ghostbusters and ghosts encountered, we stop moving l and this becomes our desired line, for which the number of Ghostbusters on one side equals the number of ghosts on the same side (if while moving l, we encountered k Ghostbusters and k ghosts, the other side will contain $(n-k)$ Ghostbusters and $(n-k)$ ghosts). The same reasoning can be applied to obtain l for the case when we only have ghosts on $CH(Q)$. Figure 2 shows an example of finding the line l when one vertex of the convex hull is a ghost and others are Ghostbusters (figure on the left) and when all vertices are Ghostbuster (figure on the right) for $n = 12$. The pseudocode of finding l is presented in the algorithm FIND-LINE below:

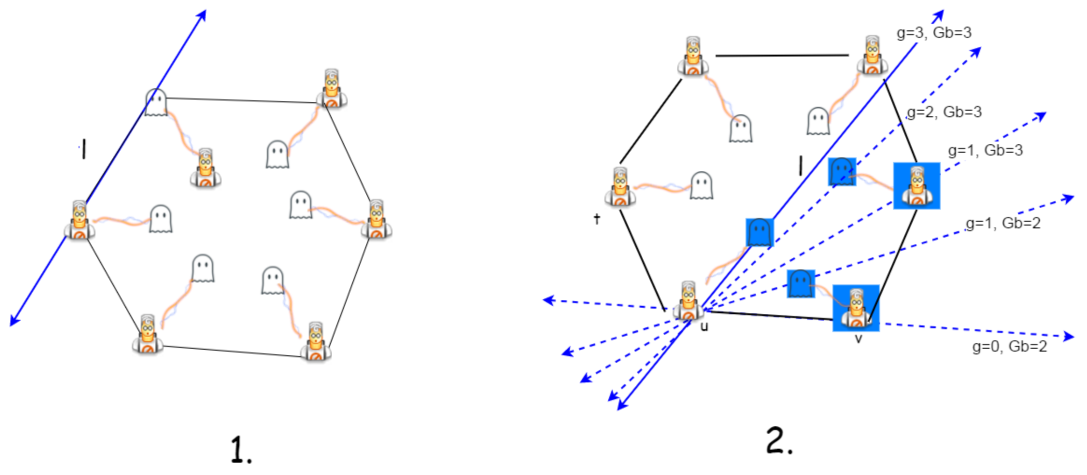

Figure 2: Finding the line l for which number of Ghostbusters on one side of the line equals the number of ghosts on the same side, for $n = 12$ points (6 Ghostbuster-ghost pairs).

```
FIND-LINE($Q$)
$S$ = GRAHAM-SCAN($Q$)
$S' = S$
$top = S'$.TOP()
$i = S'$.POP()
while $S'$.TOP $\neq NULL$
    $j = S'$.POP()
    if $(i, j)$ is a Ghostbuster-ghost pair
        return line $l$ passing though points $(i, j)$
    else
        $i = j$
if $(i, top)$ is a Ghostbuster-ghost pair
    return line $l$ passing though points $(i, top)$
// If no pair found, its case 3. i.e. all vertices are either Ghostbusters or Ghosts
Let $u$ be any arbitrary point $\in S$ and $v$ be
another point $\in S$ that is adjacent to $u$
Let $< p_1, p_2, ..., p_m >$ be the points in $Q - \{u, v\}$,
sorted by polar angle in counterclockwise order around $u$
if $u$ is a Ghostbuster
    $g = 0$
    $Gb = 2$
else
    $g = 2$
    $Gb = 0$
for $i = 1$ to $m$
    if $p_i$ is a ghost
        $g = g + 1$
    else
        $Gb = Gb + 1$
    if $g == Gb$
        return line $l$ passing through points $(u, p_i)$
```

Clearly in Find-Line, since we need to compute the convex hull for finding the said line l, the running time would be equal to the running time of the algorithm for finding the convex hull, i.e. Graham-Scan, which is $O(n \lg n)$. For case ***3.***, we need to do more processing after computing the convex hull to find l, in the worst case we will have to check for a total of $n - 3$ points to obtain l (there will always be one Ghostbuster-ghost pair that will never be checked, i.e. the other adjacent vertex of u i.e. vertex t on the edge of $CH(Q)$ and its paired ghost). This would require sorting the $n - 3$ points by polar angle in counterclockwise (or equivalently clockwise) order around u as done in **line 15** above, which will also take $O((n-3)\lg(n-3)) \in O(n \lg n)$. Hence, the worst case running time for finding the line l will be $T(n) = 2O(n \lg n)$.

b. We can use the Find-Line algorithm discussed above to compute the line l which divides the set Q of Ghostbusters and ghosts into two parts Q' and Q'' such that the number of Ghostbusters in Q' equals the number of ghosts in Q', similarly the number of Ghostbusters in Q'' equals the number of ghosts in Q''. Since, the number of Ghostbusters and ghosts are equal in each part, we can be sure that pairing the Ghostbuster and ghost present on l will not lead to streams crossing in either of the two parts Q' and Q''. We can now recursively apply Find-Line on Q' and Q'' to obtain lines that further divide these parts such that the streams will not cross between their sub-parts and pair the Ghostbusters and ghosts present on the dividing lines. This will continue until both the parts are empty and all the Ghostbusters and ghosts in Q have been paired such that no streams would cross. The pseudocode for the algorithm Find-Pairs is presented below:

```
Find-Pairs($Q$)
1  if $Q \neq \{\}$
2      $l$ = Find-Line($Q$)
3      Let $(u, v)$ be the points on $l$
4      Pair($u, v$)
5      Let $Q'$ and $Q''$ be the sets of points on the two sides of $l$
6      Find-Pairs($Q'$)
7      Find-Pairs($Q''$)
```

In Find-Pairs, we use Pair(u, v) to pair together the points u and v (**line 4**). We know that the line l will not always divide the set Q in a similar way, for example l might divide Q into equal sized sets, where each of the sets have $\frac{n-2}{2}$ points. On the other hand if a Ghostbuster-ghost pair is found on an edge of the convex hull of Q, l will divide Q into $Q' = \{\}$ and $Q'' = Q - \{u, v\}$. Since, in each recursive call to Find-Pairs we either call Find-Line and pair two points or simply return (i.e. in the case that Q was empty) we will always have a total of $\frac{n}{2}$ calls to Find-Pairs (one call for each Ghostbuster-ghost pair), in which we will be calling Find-Line and pairing two points. As each call to Find-Line takes $O(n \lg n)$ time and we make $\frac{n}{2}$ such calls, we will get a running time of $O(n^2 \lg n)$ for Find-Pairs.

We can improve upon the Find-Pairs algorithm by first making a call to Graham-Scan and pairing all the Ghostbusters and ghosts that are adjacent on the edges of the convex hull of Q, removing all of these paired points from Q, and applying the algorithm recursively on the set of remaining unpaired points in Q. However, if during any recursive step, we end up getting a convex hull that didn't have any Ghostbuster-ghost pairs, we continue to **line 1** of the Find-Pairs algorithm above. This way we can pair together multiple Ghostbusters and ghosts during a single recursive call (and for the particular case, when all the pairs are found on the convex hull in the first call, it would only take $n \lg n$ time). This will however, not change the worst case running time of the algorithm, because we can still successively encounter the case when Q gets split into an empty set and a set containing $|Q| - 2$ points which would result in a $O(n^2 \lg n)$ running time.