CS345 Assignment 1

Chinmay Goyal(180206) and Supreeth Baliga(180801)

1 Non Dominated Points

From class lecture we know about an *O(nh)* and an *O(n log n)* algorithm to find the non - dominated point in a set(where h is the number of non dominated points).

Solution:

We first tried to come up with a solution by considering the time complexity. But after many attempts, we realised that this would be a difficult task to carry out by this approach. Instead, we started to first formulate algorithms one-by-one and then prove that the time complexity of the formulated algorithm fits the required frame i.e. *O(n log h)*. After many attempts, we could formulate an algorithm which runs in *O(n log h)* time complexity.

We describe our algorithm as follows:
As assumed in class, even here we are going to assume that no two points have the same x-coordinate or y-coordinate. The question gives us a hint about taking insights from the first algorithm of time complexity of *O(nh)* that we keep eliminating points from the set which are dominated by the current point and then move further. This is what we are using in our algorithm. Let us denote our routine as $findNonDominated(S)$ where S denotes the current set of points we are working on. This routine returns the non-dominated points in S. Below, we are outlining the routine.

Let us say we are currently working on the set of points P. We need to find the x-median (say X_m) of these points in *O(n)* time and divide the set P into two sets P_{left} and P_{right}. So P_{left} contains all the points in the set P with x-coordinates less than or equal to X_m and P_{right} contains all points with x-coordinate greater than X_m. Now, we find the point with maximum y-coordinate (say p) in P_{right}. Clearly, p is a non-dominated point. Now, one thing we can be sure of is that every point in P_{left} with y-coordinate less than that of p is dominated by p. So, we don't need to consider such points anyway as they are not non-dominated. So we delete such points from P_{left}. Furthermore, the points in P_{right} with x-coordinate less than that of p are again dominated by p. Hence, we can delete such points from P_{right}. Now, we recursively call $Q_{left} = findNonDominated(P_{left})$ and $Q_{right} = findNonDominated(P_{right})$. Here, Q_{left} denotes the set of non-dominated points in P_{left} and Q_{right} of non-dominated points in P_{right}. So, the non-dominated points in P is simply $Q_{left} \cup Q_{right}$.

The main difference between this method and the divide-and-conquer algorithm used in class is that, in this algorithm, we are removing all the dominated points before itself and then diving into deeper recursions. So, we can be sure that the non-dominated points that we get from the left half are non-dominated in the whole set, since we already removed the dominated points beforehand. So, this gives us a smaller set of points to process.

1.1 Pseudo-Code

```
$findNonDominated(P)$ {
    if ( |$P$| == 0 ) return $\emptyset$;
    else if( |$P$| == 1 ) return P;
    Compute x-median $X_m$ of points in $P$;
    $P_{left}$ <- points in $P$ with x-coordinate $\leq X_m$;
    $P_{right}$ <- points in $P$ with x-coordinate $> X_m$;
    $p$ <- point in $P_{right}$ with maximum y-coordinate;
    Delete every point in $P_{left}$ with y-coordinate less than that of $p$;
    Delete every point in $P_{right}$ with x-coordinate less than that of $p$;
    $Q_{left}$ = $findNonDominated(P_{left})$;
    $Q_{right}$ = $findNonDominated(P_{right})$;
    return $Q_{left} \cup Q_{right}$;
}
```

1.2 Time Complexity Analysis

T(n,h) = Time taken to find all h non dominated points in a set of n points.

Our claim: $T(n, h) \leq c_1 n + c_2 n log h$

For h = 0, n has to be 0 because if there exists some number of points say n then there is a point with the maximum x-coordinate, and that point will be a non-dominated point. Hence: $T(n, 0) = O(1)$

For h = 1, let us say that the point p is the only non-dominated point. For p to be the only non-dominated point, he following should hold:

$$p_x > q_x \wedge p_y > q_y$$

$$\forall q \in P, q \neq p$$

After finding the point p in the first iteration of the algorithm, all the remaining points are removed, hence: $T(n, 1) = O(n)$

We use induction to prove our claim. We first state our base case:

$$T = \begin{cases} T(n, 0) = O(1) \\ T(n, 1) = O(n) \end{cases}$$

We first find the X_m which takes O(n) time. We then find the point p with maximum y-coordinate in P_{right} and remove the points dominated by p which also takes O(n) time. The recursive calls take $\leq T(\frac{n}{2}, h_{left})$ and $\leq T(\frac{n}{2}, h_{right})$ time because after removing the points dominated by p, the number of points in left and right half are $\leq \frac{n}{2}$. The conquer step only requires merging Q_{left} and Q_{right}, which again takes O(n) time. Hence:

$$T(n, h) \leq T(\frac{n}{2}, h_{left}) + T(\frac{n}{2}, h_{right}) + c_1 n$$

where $h_{left} + h_{right} = h$; $h_{left} \geq 0$ and $h_{right} \geq 1$

Hence we have proved for our base cases. Now we use induction:

$$\begin{aligned}
T(n,h) &\leq c_1 n + c_2\frac{n}{2}log(h_{left}) + c_3\frac{n}{2} + c_2\frac{n}{2}log(h_{right}) + c_3\frac{n}{2} \\
&= c_1 n + c_2\frac{n}{2}log(h_{left}h_{right}) + c_3 n \\
&\leq c_4 n + c_2\frac{n}{2}log(\frac{h^2}{4}) \text{ (using } h_{left} + h_{right} = h; A.M. \geq G.M.) \\
&= c_4 n + c_2 nlog(\frac{h}{2}) \\
&\leq c_5 nlog(2) + c_5 nlog(\frac{h}{2}) \\
&\leq c_5 nlog(h)
\end{aligned}$$

Hence we prove that:

$$T(n,h) = O(nlog(h))$$