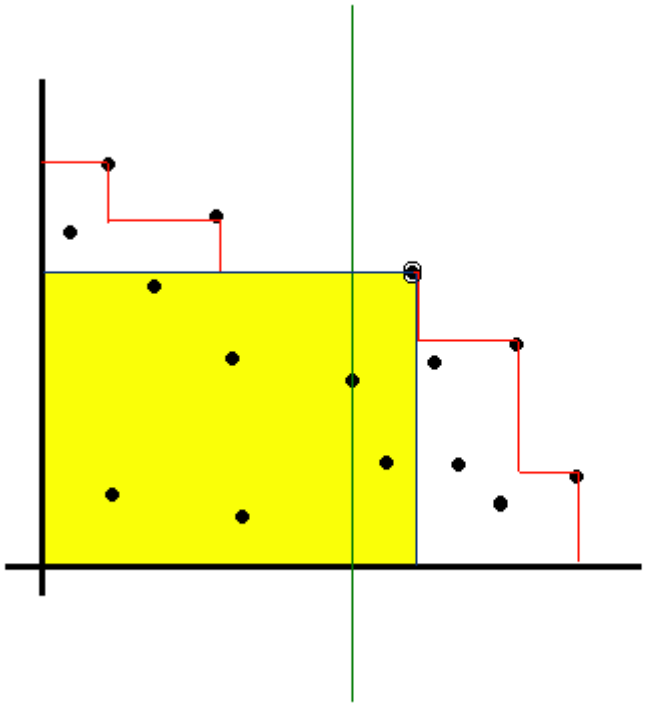
Assignment 1

Aryan Choudhary - 170152
Abhinav Sharma - 180017

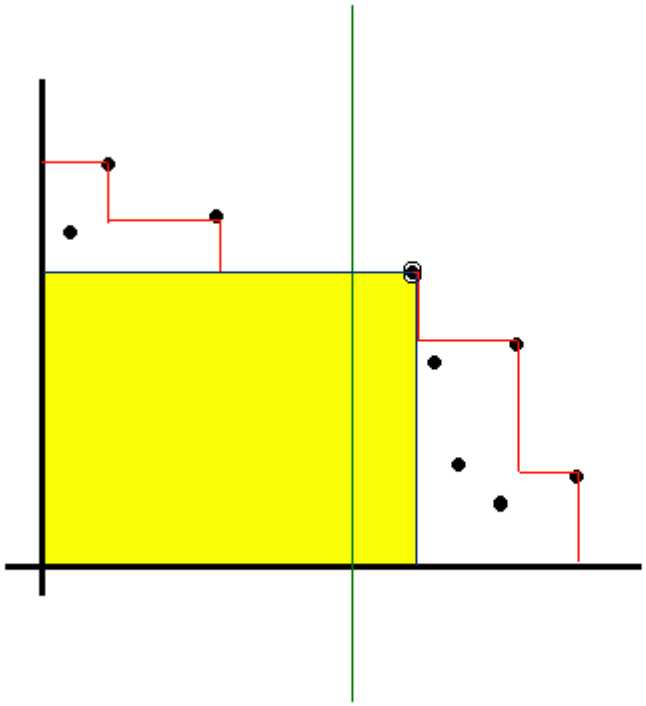
Non-dominated points

Algorithm

The algorithm is based on divide and conquer approach, similar to the one explained in the lecture, with slight modifications. In order to optimise the complexity, on every recursive step we will try to remove the points which we know that cannot be a non-dominated point.
Consider the set of given points in 2-D plane and a line through x-median of the points which divides the set into two set with almost equal number of points.



The encircled point is the non-dominated point in the right half with maximum y-coordinate. We can see that all the points within the yellow region are dominated by the encircled point. So these points can never be non-dominated. Therefore we can simply remove these points.



Further we can recurse over left and right half of the remaining points and combine the two sets of non-dominated points from left and right half to get the complete set of required points.

Pseudo Code

Non-dominated_points(P)

\\P is the set of points under consideration. All points in P have distinct x,y coordinates.

1. if P.size ≤ 1 \\Base Case
2. return P
3. end if
4. $X_{med} \leftarrow$ x median of points in P
5. $P_l \leftarrow$ set of points in P with x-coordinate less than or equal to X_{med}
6. $P_r \leftarrow$ set of points in P with x-coordinate greater than X_{med}
7. $R_{max} \leftarrow$ point with maximum Y-coordinate in P_r
8. Remove all the points in P_l with y-coordinate less than y-coordinate of R_{max}
9. Remove all the points in P_r with x-coordinate less than x-coordinate of R_{max}
10. $S_l \leftarrow$ Non-dominated_points(P_l)
11. $S_r \leftarrow$ Non-dominated_points(P_r)
12. return $S_l \cup S_r$

end

Correctness

The only difference in this algorithm with the one explained in lecture is that at each recursive step we try to reduce the size of the set by removing some points in step 8,9. The removed points are always dominated by the point with maximum y-coordinate in P_r (as shown using figure above), so we never remove any non-dominated point from the set. Also we observe that points in P_l and P_r are mutually exclusive and x-coordinate of all points in P_r is greater than than x-coordinate of points in P_l and y-coordinate of remaining points in p_l is greater than y-coordinate of all points in P_r. Therefore, any point in one set does not dominates any point in other set. Also any point in S_l or S_r is present in overall dominant set. Hence points returned after recursive calls doesn't contain any redundant point.

Complexity Analysis

The above algorithm is based on divide and conquer recursive approach, so we can easily form a recursive equation for time complexity.
Let T(n,h) be the time complexity of the above algorithm where n is size of set of points (i.e. $|P|$) and h is the number of non dominated points in the set. Then,

$$T(n,h) = T(n_1,h_1) + T(n_2,h_2) + O(n)$$

where $n_1 \leq n/2$, $n_2 \leq n/2$ and $h_1 + h_2 = h$.
Each step from 4 to 9 can be done in linear time which contributes to O(n) term in the equation.

Solving the recurrence using substitution method

Steps involved in substitution method are -
1. Prove using strong mathematical induction principle that the relation you have guessed holds.
2. Show that the relation holds for boundary condition instead of base cases in induction. Since for time complexity asymptotic analysis is sufficient.

1. Induction step for recurrence -

Claim - T(n,h) $\leq$ c*n*logh for all 1 <= $n < k$ and 1 <= h <= $n < k$,
where $k_1 \leq n/2$, $k_2 \leq n/2$ and $h_1 + h_2 = h$

$T(k,h) = T(k_1,h_1) + T(k_2,h_2) + O(k)$
$\implies$ T(k,h) = c * $k_1 * \log(h_1) + c * k_2 * \log(h_2) + c_2 * k$
$\implies$ T(k,h) $\leq$ c* k/2 * $\log(h_1) + c * k/2 * \log(h_2) + c_2 * k$
$\implies$ T(k,h) $\leq$ c* k/2 * $\log(h_1 * h_2) + c_2 * k$
$\implies$ T(k,h) $\leq$ c* k/2 * $\log(h_1 * (h - h_1)) + c_2 * k$

$h_1 * (h - h_1)$ is a quadratic equation and maximum value of it occurs at $h_1 = h/2$ and is equal to $h^2/4$
$\implies$ T(k,h) $\leq$ c* k/2 * $\log(h^2/4) + c2 * k$
$\implies$ T(k,h) $\leq$ c* k* $\log(h) + (c_2 - c\log 4) * k$

For large h. $\log h$ dominates $c_2 - c\log 4$. Hence,
$\implies$ T(k,h) $\leq$ c* k* $\log(h)$
$\implies$ T(k,h) = O(k*log(h))

2. For boundary condition we will analyse worst case where $n_1 = n_2 = n/2$ and $h_1 = h_2 = h/2$ and show that this time complexity $T(n,h) = n * \log h$, which will complete the proof.

$T(k,h) = T(k/2,h/2) + T(k/2,h/2) + O(k)$
$\implies$ T(k,h) = 2*T(k/2, h/2) + O(k)
T(k/2,h/2) = 2*T(k/4, h/4) + O(k/2)
$\vdots$
T($\frac{k}{2^{logh}}$,1) = O($\frac{k}{2^{logh}}$)

Proof of $T(n,1) = O(n)$: A point with maximum x coordinate is dominant point. Similarly a point with maximum y coordinate is also a dominant point. Our problem assumes that all x coordinate and all y coordinates are distinct. A set of points can have only one dominant point if that dominant point has both maximum x and maximum y coordinate among all set of points.

In case we have one such point and no of points are more than 1. Our algorithm works as follows - Step 5,6 divides set P into 2 equal half P_l, P_r. Then step 8 removes all points in P_l and step 9 keeps only one point in P_r. Then in base case in next recursive call returns it as the dominated point of entire set. All steps 4 - 9 takes $O(n)$ time and 10-12 take $O(1)$ time. Hence $T(n,1) = O(n)$

On solving the above set of equations, we get-

$T(k,h) = c*k + 2*c*(k/2) + 2^2*c*(k/2^2) + ... + 2^{logh}*c*(k/2^{logh})$
$\implies$ T(k,h) = $\log h * c * k$
$\implies$ T(k,h) = O(k*$\log h$)

Analytically, we can say that the worst case occurs when we cannot remove any point in the removal step in each recurrence and number of dominated points gets divided equally in each set. In, such a case the set of points will simply get divided into half at each level and there exist atleast one non-dominated point in each recurrence. Mathematical proof of time complexity for this case shown above.