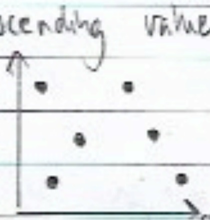
CS3230 challenge 1

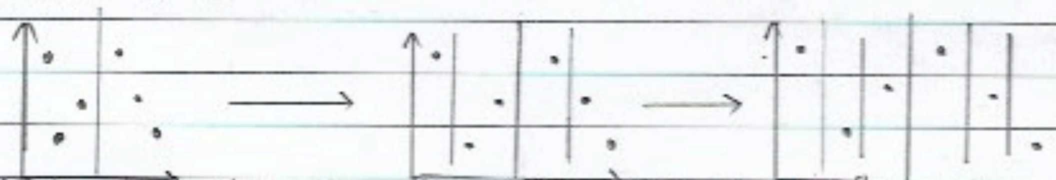
Date No.

Given a set S, an algorithm is required to return a subset of S where it contains all the dominating points. A dominating point (x_1, x_2, x_3) where for any x_i is greater than y_i for other point (y_1, y_2, y_3).

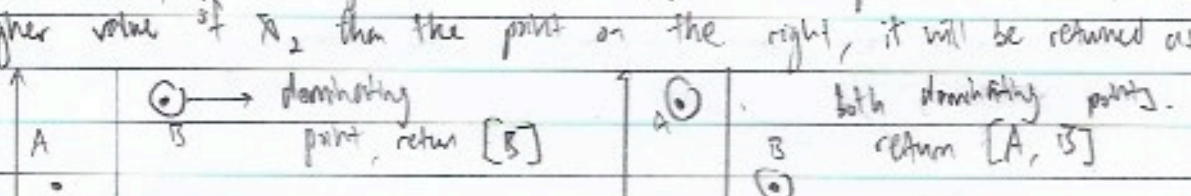
By considering the problem with dimension $d=1$ the dominating point can be found in $O(n)$ time, which is just the largest value on a straight line. When $d=2$, the dominating set can be found in $O(n \lg n)$. First, merge sort the points (x_1, x_2) with ascending values of x_1. When x_1 is equal, tie is broken with ascending values of x_2. We have a graph which looks like this:



Next, we are going to apply the divide and conquer technique. We are going to partition the graph by splitting the points into 2 problems of approximate equal size, until the size is 1.



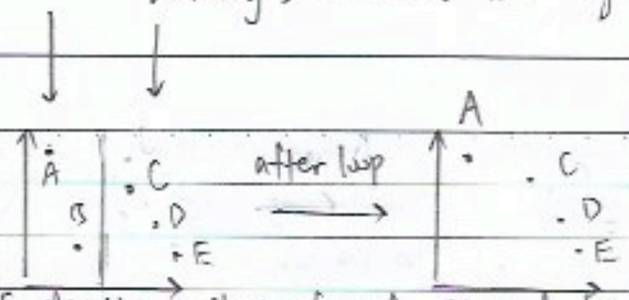
We can observe that when the problem size is 1, the point on the left side of the partition will be returned as part of the dominating set, since it has a larger value of x_1 than the point on the left. If the point on the left has a higher value of x_2 than the point on the right, it will be returned as well.



Now we have our base case to be correct. Assume the correctness holds at the ith iteration. At the i+1 iteration, we have the returned set of dominating points on both the left and right partition. Because the previous iteration first adds A, then adds B, so the set is in sorted order. We loop through the set returned from the left, starting from the first element, then discard them if they have a value of $x_2 <$ the value of x_2 of the first element in the set returned from the right. Because the set returned from the right are already dominant in terms of x_1, so every element after the first element in the returned set from the right is dominant. An example as follows:

dominating set returned from left, {A, B}

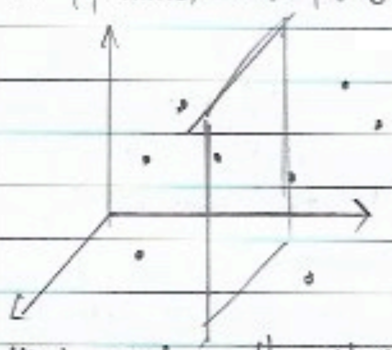
dominating set returned from right {C, D, E}



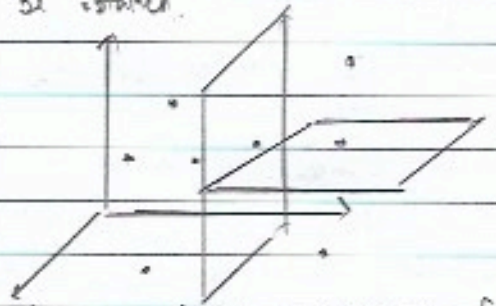
Explanation: Starting from A. Since A has a higher x_2 value than C, it is remained in the dominating set. However, B has a x_2 value < C, so it is discarded. The returned set of dominating points is {A, C, D, E}. The induction step is complete, so the algorithm is correct for 2d space.

Time complexity: $2T(\frac{n}{2}) + f(n)$. The algorithm divides the problem into 2 equal parts, and the looping to return the set takes O(n), hence the time is O(n lg n). Taking into account the merge sort at the beginning, we have a time complexity of n lgn + n lgn = 2 n lgn = O(n lg n).

In a 3d space, we can apply the similar concept. Merge sort the points based on x_1 value, and tie breakers with x_2 and x_3. Divide the points into 2 spaces:



Next, perform the above algorithm, and the dominating set of points can be obtained.



Time complexity: $2T(\frac{n}{2}) + f(n \lg n) = O(n \lg^2 n)$

Here, the f(n lg n) is just the operations we perform on a 2d space to get the returned set of points.