CS345
Design and Analysis of Algorithms
Indian Institute of Technology, Kanpur

Tarun Kanodia (190902), Kaustubh Verma (190424)

Assignment 1

Date of Submission: 22th August, 2021

Question 1

You can extend the notion of non-dominated points to 3 dimensions as well. Spend some time to realize that it is not so simple to apply divide and conquer strategy to compute the non-dominated points in 3 dimensions. Some of you may be tempted to solve this problem by reducing an instance of this problem to three instances of 2-dimensional problem (by projecting the points on x-y, y-z, x-z plane). But that will be wrong (think over it to convince yourself). Interestingly, there is a very simple elegant algorithm using a simple data structure that computes non-dominated points in 3 dimensions. The purpose of this exercise is to make you realize this fact.

1. Design an algorithm that receives n points in x-y plane one by one and maintains the non-dominated points in an online fashion. Upon insertion of ith point, the algorithm should take $O(logi)$ time to update the set of non-dominated points. Note: It is perfectly fine if your algorithm only guarantees a bound of $O(i \log i)$ on the total time for insertion of i points. It is not necessary that your algorithm achieves $O(\log i)$ bound on the processing carried out during insertion of ith point.

2. Design an $O(nlogn)$ time algorithm to compute non-dominated points of a set of n points in 3 dimensions. You must use part (a) above carefully

Solution

1. We need a self balancing binary tree (or a **Red-Black Tree**) for efficient insertion and deletion operations. We know that when we maintain the set of non-

dominated points in increasing order of x-coordinates, the y-coordinates are sorted in decreasing order.

Algorithm

We process the points in in an online fashion as we receive them, and maintain a red-black tree sorted by x-coordinates. Thus the y-coordinates would be sorted in decreasing order in the RBT (i.e, inorder traversal of RBT would yield y-coordinates in decreasing order). Therefore, if we have the set of current non-dominated points for $i-1$ points, for inserting the $i-th$ point, we do the following:

(a) Insert the $i-th$ point in the tree using $O(logi)$ time.

(b) Find the inorder successor of the newly inserted point and check if the successor dominates the current point. There are following two cases:

- If the inorder successor dominates the $i-th$ point, remove the current point.
- If the inorder successor does not dominate the $i-th$ (or does not exist), we delete all the inorder predecessors of the current point, that are dominated by the $i-th$ point.

The set now contains the set of non-dominated points for i points.

Proof of Correctness

The algorithm is inductive in nature. So we will prove the correctness using induction. For base cases ($n = 1$), the set only contains the given point and hence it contains the set of all non dominated points. We now prove the following lemma which handles the inductive step of the proof and shows that after the $i-th$ step, the set contains the set of non-dominated points for the first i points.

Lemma 1.1. If the successor of a newly inserted point in the RBT does not dominate it (and hence the current point is successfully inserted in the RBT), then the inserted point is a non-dominated point for the current stream of points.

Proof. Assume that the RBT contains the non-dominated points for $i-1$ points.

We know that RBT contains the points sorted in increasing order of x-coordinates, and hence they will be in decreasing order of y-coordinates. We insert the $i-th$ point in the RBT. Now the predecessors of $i-th$ point can not dominate it, since they have a lesser x-coordinate than the $i-th$ point. We also know that the successors of the $i-th$ point are sorted in decreasing order of y-coordinates. So if the current point is not dominated by it's successor, we are sure that it would not be dominated by further successors of the RBT.

Therefore, $i-th$ point, if successfully inserted (i.e,is not dominated by it's successor), is one of the non-dominated points for the first i points, as it is not dominated by any point from the set of non-dominated points for the previous $i-1$ points. Moreover, we also remove the points that are dominated by the current point, and hence can not be non-dominated for the first i points. Hence, after the $i-th$ point is inserted, the set contains all the non-dominated points among the first i points. □

Time Complexity Analysis

Time complexity for inserting $i-th$ point into RBT = $O(\log i)$
Time complexity for finding the successor/predecessor = $O(\log i)$

We know that deletion operations are supported by RBT with $O(\log i)$ complexity. But we do not know how many deletions take place at a particular step. So we have to do the **amortized time complexity** analysis. We know that each of the i points are inserted once in the RBT, and is deleted atmost once.So the overall number of deletions can not exceed i (with overall complexity $O(i \log i)$), and hence the upper bound on time complexity at the end of inserting $i-th$ point is $O(i \log i)$. (Note: Hence, for inserting all n points, the overall complexity of algorithm in $O(n \log n)$.

2. Algorithm

Algorithm 1 Algorithm to find set of non-dominated points in 3D

```
1: S denotes set of n points.
2: Sort S in decreasing order of z coordinates.
3: N ← φ                          ▷ Contains all the non-dominated points at the end
4: T ← φ                          ▷ RBT containing points sorted by x-coordinates
5: for each point P in S do
6:     P' ← projection of P onto x-y plane in T
7:     Insert P' into T as per the first part
8:     K ← successor of P' in T
9:     if K is not NULL and K dominates P' then
10:        Delete P' from T
11:    else
12:        N ← N ∪ {P}
13:        K ← predecessor of P' in T
14:        while K is not NULL and P' dominates K do
15:            Delete K from T
16:            K ← predecessor of P' in T
17:        end while
18:    end if
19: end for
```

Proof of Correctness

The proof for this algorithm is along the same lines as shown in the first part. We process the point in decreasing order of z-coordinates, and then follow the algorithm in the first part to insert the points in online fashion in the tree. Processing the points in decreasing order of z-coordinates ensures that the current point does not dominate the previously processed points.

Again considering the inductive step, when the $i-th$ point is being processed, we assume that N contains non-dominated projections of the first $i-1$ points onto x-y plane. Now $z_i < z_j \ \forall j < i$. Hence, current point can not dominate the previously processed points. So extending Lemma 1.1 for 3-D points, if the projection of $i-th$ point is inserted into the RBT successfully, then it is a non-dominating point for the first i points (otherwise, if the projection of the $i-th$ point is dominated by some other previously processed point, then it can not be non-dominating, because the point dominating it's projection would have a greater z-coordinate too, and hence would dominate the current point).

Moreover the algorithm deletes projection P' from the RBT (and also does not include P in N) if it is dominated by it's successor. Therefore, at the end of $i-th$ step, N contains the set of all non-dominated points among the first i points.

Time Complexity: Sorting points by decreasing z-coordinate takes $O(n \log n)$ time, and the getting non-dominated set for projections takes $O(n \log n)$ time as shown in the first part. Therefore, overall complexity = $O(n \log n)$.