# Assignment 3

Boubacar Ballo - bab9755

April 4, 2024

## Question 1

For the look-ahead to be of 6 instead of 11. After running the algorithm as usual where we sort all the points by x-coordinate and find the median line to divide them into two parts. After recursively finding the minimum distance between two points on each side, we then find the smallest of the two (let us call it $\alpha$). Now, we should consider the strip of width $2\alpha$ that covers some points on the left side and the right side of the the median line. After sorting those points by y-coordinates, we quickly realize that we only need to compare the current point with 6 other points within the sub-list. Geometrically, if we had more than 6 neighbors, then the distance property set by $\alpha$ would be violated. Each of those neighbors should be on the left or right $\alpha * \alpha$ square. Within each square, all points have distance $\geq \alpha$. So excluding our current point, we have 3 points maximum in each square, which is a total of 6. We hence do not need to loop through every single point and the running time for the process can become at most O(n). To implement this component of the algorithm, we check for all pairwise y-distances that are smaller than $\alpha$. Then if the Euclidean distance between the two points is smaller than $\alpha$, then update $\alpha$ as the minimum distance.

## Question 2

An approach to this questions is to divide the cards into two stacks A and B and form card pairs with one card from A and another one from B. In the case the number of cards is odd, we will have one card that will be labeled as unmatched and conserved for later. For each pair, we then check for equivalence. If a pair is equivalent, we retain one of the cards and discard the other. On the contrary, if the pair is not equivalent, we can discard both cards. All the discarded cards and the unmatched ones are kept together for the following rounds of matching. We can now get the candidate and match it against all the cards. We get a majority set whenever the candidate matches with more than half of all the cards. We only preserve the majority status of a class after every step if the majority equivalence class is in the original collection. After each round, the number of cards is given by the series $n/2+$ $n/8 + ...$ ...

which is $\leq 2n$. So the Time complexity is $O(n)$

## Question 3

To solve this question, we can start at the node v and probe it. After getting its value, we can compare it with its children. if $x_v < x_w$, then we return the node v as a local minimum. On the contrary, if $x_v > x_w$, we move to the w node and perform the same comparison operation as mentioned above. If we keep on getting cases were the children are greater than the parent node, we will hit a case where we are dealing with leaf nodes and there would not be any comparisons possible. We would therefore return that leaf node w as the local minimum of the subtree. Considering that this is the worst case scenario, we would have traverse the entire depth of the tree, which is $O(d)$ or otherwise $O(logn)$ where n is the number of nodes that we have