Assignment 1
Algorithms Design and analysis

Name: Mahmoud Mohmed Abdelaziz

ID: 19016578

I acknowledge that I'm aware of the academic integrity guidines of this course, and that I worked on this assignment independently without any unauthorized help

محمود محمد عبد العزيز

Scanned with CamScanner

Note:

There are some tests in WorstCaseLinearAlgorithm and in MaxSideLength, it is passing tests included in the lab document and many other tests, but for big tests that I can't trace, sometimes it fails, I will work on solving this issue, unfortunately I'm not able to finish this before deadline, I will leave a github repo containing the code that I will try to solve its issues, and when done I will push it with a commit named "Done", I hope you take a look at it when reviewing the assignment solution, and thanks in advance.

The link: https://github.com/Mahmoud-Moh/Algorithms_Assignment.git

Randomized Select for getting Median:

```
3 usages
private int RandomizedSelect(int A[], int p, int q, int i){
    if(p==q) return A[p];
    int r = partition(A, p, q);
    int k = r - p + 1;
    if(i==k) return A[r];
    if(i<k) return RandomizedSelect(A, p,  q: r-1, i);
    else return RandomizedSelect(A,  p: r+1, q,  i: i-k);
}
```

```
1 usage
static int partition(int A[], int p, int q)
{
    int r = RandomGenerator.getDefault().nextInt(p, q);
    int pivot = A[r];
    swap(A, r, q);

    int i = (p - 1);

    for (int j = p; j <= q - 1; j++) {

        if (A[j] < pivot) {

            i++;
            swap(A, i, j);
        }
    }
    swap(A,  i: i + 1, q);
    return (i + 1);
}

3 usages
static void swap(int[] arr, int i, int j)
{
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

Worst case linear time Median

```
2 usages
public int getMedian() { return Select(arr,  p: 0,  q: arr.length - 1,  i: (arr.length+1) / 2); }

4 usages
private int Select(int A[], int p, int q, int i){
    int n = q - p + 1;
    if(n <= 5)
        return MedianOf5(A);
    int[] medians = new int[n/5];
    int c = 0;
    for(int j=p; j<n/5; j++){
        int[] arr = new int[5];
        System.arraycopy(A, j, arr,  destPos: 0 ,  length: 5);
        //System.out.println(Arrays.toString(arr));
        int median_of_5 = MedianOf5(arr);
        //System.out.println(median_of_5);
        medians[j] = median_of_5;
    }
    //This is going to be changed
    int median_of_medians = Select(medians,  p: 0,  q: medians.length - 1,  i: (medians.length + 1) / 2);
    int r = partition(A, p, q, median_of_medians);
    int k = r - p + 1;
    if(i==k) return A[r];
    if(i<k) return Select(A, p,  q: r-1, i);
    else return Select(A,  p: r+1, q,  i: i-k);
}
```

2 usages

```
int MedianOf5(int arr[]) {
    Arrays.sort(arr);
    return arr[(arr.length + 1) / 2 - 1];
}
```

1 usage

```
static int partition(int A[], int p, int q, int medianOfmedians) {

    int r = findIndex(A, medianOfmedians);
    int pivot = medianOfmedians;
    swap(A, r, q);

    int i = (p - 1);

    for (int j = p; j <= q - 1; j++) {

        if (A[j] < pivot) {

            i++;
            swap(A, i, j);
        }
    }
    swap(A, i + 1, q);
    return (i + 1);
}
```

Max side length:

Used function closestPair from lecture

```
public double calcMaxSideLength(){
    sortPoints();
    double minm_distance = FindClosestPair(0, points.length);
    return max(abs(closestPair[0].x - closestPair[1].x) ,
            abs(closestPair[0].y - closestPair[1].y)  );
}
```

```
3 usages
double FindClosestPair(int start_index, int end_index){
    if ((end_index - start_index) <= 3) {
        return bruteForce(start_index, end_index);
    }
    int mid_index = start_index + (end_index - start_index) / 2;
    double sigma_l = FindClosestPair(start_index, mid_index);
    double sigma_r = FindClosestPair(mid_index, end_index);
    double sigma_min = min(sigma_r, sigma_l);
    ArrayList<Point> points_in_strip = new ArrayList<~>();
    for(int i=start_index; i<end_index; i++){
        if(abs(points[i].x -  points[mid_index].x) <= sigma_min)
            points_in_strip.add(points[i]);
    }
    return minInStrip(points_in_strip, sigma_min);
}
```

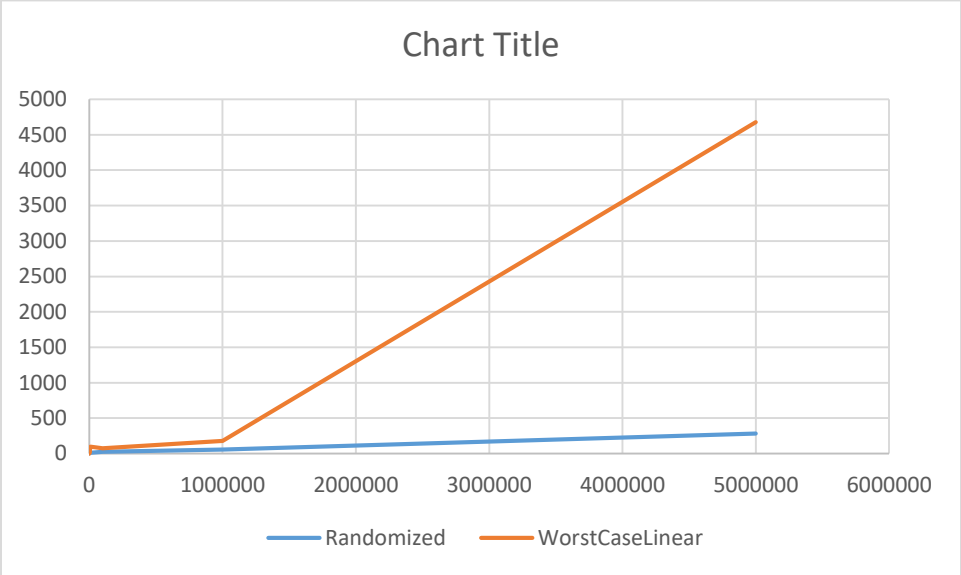
```
1 usage
double bruteForce(int start_index, int end_index){
    double d = Integer.MAX_VALUE;
    for(int i=start_index; i<end_index; i++){
        for(int j=i+1; j<end_index; j++) {
            double x = distance(points[i], points[j]);
            if(x < d) {
                double y;
                if(closestPair[0] == null)
                    y = Integer.MAX_VALUE;
                else
                    y = distance(closestPair[0], closestPair[1]);
                if(x < y) {
                    closestPair[0] = points[i];
                    closestPair[1] = points[j];
                }
                d = distance(points[i], points[j]);
            }
        }
    }
    return d;
}
```

```
double distance(Point p1, Point p2){
    return Math.sqrt((p1.x - p2.x) * (p1.x - p2.x) +
            (p1.y - p2.y) * (p1.y - p2.y)
    );
}

1 usage
double minInStrip(ArrayList<Point> points_in_strip, double prev_min){
    double d = prev_min;
    int n = min(points_in_strip.size(), 7);
    for(int i=0; i<points_in_strip.size(); i++){
        for(int j=1; i+j< n; j++){
            double x = distance(points_in_strip.get(i), points_in_strip.get(i + j));
            if(x < prev_min) {
                double y;
                if(closestPair[0] == null)
                    y = Integer.MAX_VALUE;
                else
                    y = distance(closestPair[0], closestPair[1]);
                if(x < y) {
                    closestPair[0] = points_in_strip.get(i);
                    closestPair[1] = points_in_strip.get(i + j);
                }
                d = distance(points_in_strip.get(i), points_in_strip.get(i + j));
            }
        }
    }
    return d;
```

Analysis: Complexity of WorstCaseLinear is practically close to Naïve method

RandomizedAlgorithm is faster.



Resources:

Partition method was inspired by this resource :

https://www.geeksforgeeks.org/quick-sort/