Luis Enrique Franco Marín

410921353

**Week 4: Maximum Subarray and Variation**

A component or section of an array is frequently referred to as a subarray. An array is a collection of variables defined collectively by a programmer. Rather than declaring numerous variables, the programmer might create a single array with multiple labels.

Many of the same operations can be performed on a subarray as they can on the complete array. Allowing operations on a subarray increases the flexibility of these tools when it comes to storing many variables in a single set.

Given this last fact, that we can perform the same operations on a subarray as we can do on the whole array, it is then possible to divide an array in subarrays perform a required operation and then compare results. This approach is called Divide and Conquer.

The Divide and Conquer approach on problem solving is based on the principle of dividing the problem into smaller problems that can be solved individually and then continue until the whole problem is solved. For this a base case is necessary and then the problem is solved, with the results of the smaller problems as inputs, recursively.

**Problem:**

Given a set of points in a two-dimensional space, you will have to find the distance between the closed two points.

The input file contains several sets of input. Each set of input starts with an integer N (0 \le N \le 10000)N(0≤N≤10000), which denotes the number of points in this set. The next NN line contains the coordinates of NN two-dimensional points. The first of the two numbers denotes the XX-coordinate and the latter denotes the YY-coordinate. The input is terminated by a set whose N = 0N=0. This set should not be processed. The value of the coordinates will be less than 40000 and non-negative.

For each set of input produce a single line of output containing a floating point number (with four digits after the decimal point) which denotes the distance between the closest two points. If there is no such two points in the input whose distance is less than 10000, print the line 'INFINITY'.

**Code:**

Text

Description automatically generated

**Discussion:**

The problem to be solved is to find the two closest points in a 2-dimensional space. For this the problem can be solved broken into parts with the base case having just 2 points. The distance between two points can be easily computed and this gives an upper bound for finding the closest distance between other two points.

The problem is solved recursively by breaking the subarray in halves, then finding the closest distance between points on both of the halves. The one with the shortest distance becomes the upper bound, then an extra step is required, it is to find the points that are close to the middle partition by the minimum distance found, this means that only points which are closer than the actual minimum distance to the middle partition can be closer to some other point that’s on the other side of the middle partition.

Once the closest distance on the left part, the right part, and the middle part are found the minimum distance is chosen. This minimum distance becomes the minimum distance of some right or left part for a larger partition of the array and so on and so forth.

The array is partitioned in subarrays which can’t exceed the total size of the array nor are they stored separately, what is stored separately is the string of points that can potentially be at a closest distance than the current closest distance and this can be potentially all the points in the current subarray. This gives the algorithm a space complexity of O(N).

A bottleneck is then encountered at the processing of the string of potential distances that cross the middle partition. The algorithm then has a time complexity of O(NlgN).

**Result:**

A picture containing graphical user interface

Description automatically generated