Design and Analysis of Algorithms

Lab - 4

Divide and Conquer

A divide and conquer algorithm is a strategy of solving a large problem by breaking the problem into smaller sub-problems, solving the sub-problems, and combining them to get the desired output.

A. Write a C/C++ program for the multiplication of integers using Karatsuba's algorithm.
B. Write a C/C++ program to find the closest pair of points using divide and conquer.

Do the run time analysis and time complexity analysis with the different values of input size. Maintain the tabular data (n, execution time) and plot it graphically using data plotting tools.

Suggestion:

The Karatsuba algorithm is used by the system to perform fast multiplication on two n-digit numbers, i.e. the compiler takes less time to compute the product than the time-taken by a normal multiplication.

To compute X*Y where X and Y are n digit numbers. (n is the max of the number of digits among two numbers). Divide the numbers into two parts, say first half and second half as:

$$X = A * 10^{n/2} + B$$
$$Y = C * 10^{n/2} + D$$

$$X * Y = (A * C) * 10^{n} + (A * D + B * C) * 10^{n/2} + B * D$$

$$X * Y = (A * C) * 10^{n} + [(A + B) * (C + D) - A * C - B * D] * 10^{n/2} + B * D$$

So here on dividing into two halves, instead of calculating we need to compute 3 multiplications.

Call the procedure recursively till we get multiplication of a single digit. That is, call in recursion for A*C, B*D and (A+B)*(C+D) and compute X*Y

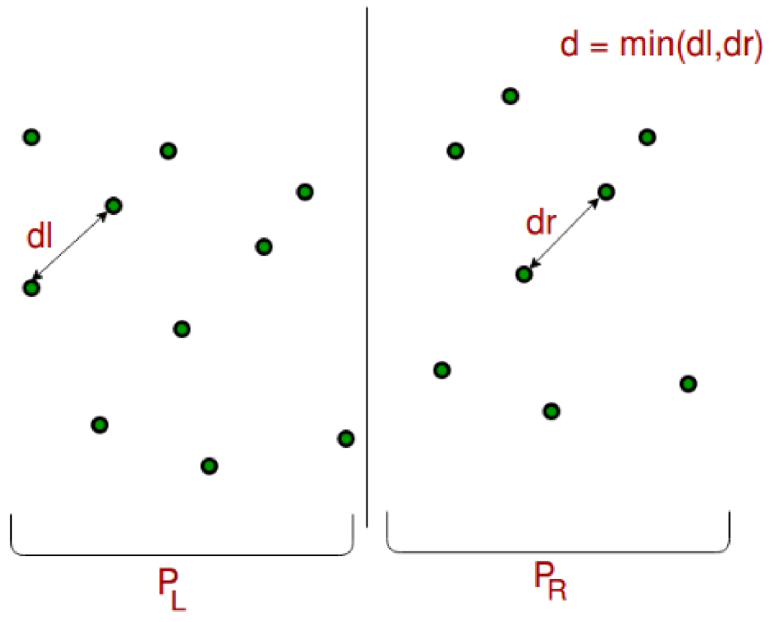
The time complexity of the Karatsuba algorithm for fast multiplication is O($n^{\log 3}$).

To find *Closest pair of points* means those two points having minimum distance among the all given points in the XY plane.

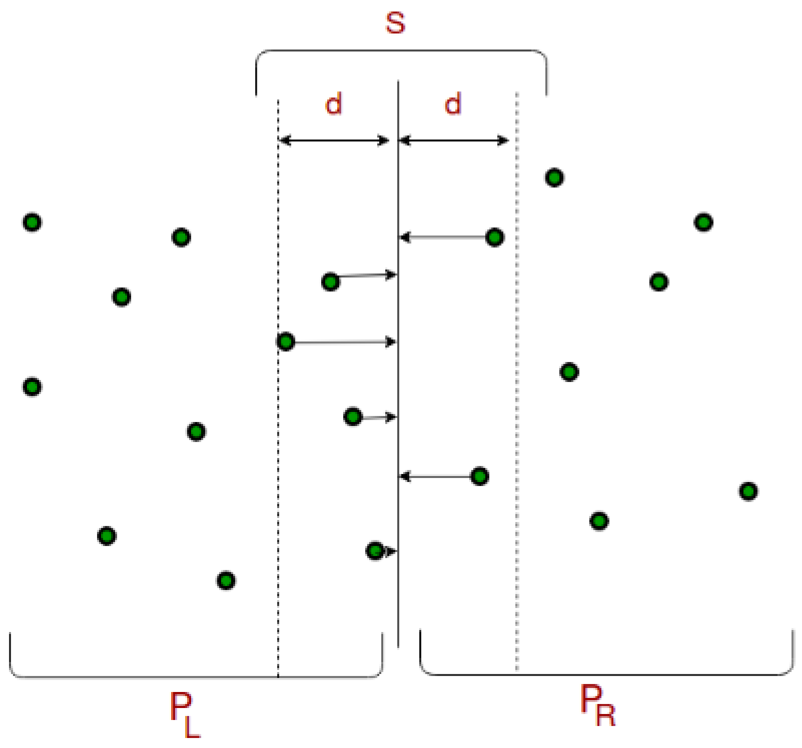
Distance between any two points P1(x1,y1) and P2(x2,y2) is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

To compute distance between every point with every other point take $O(n^2)$ time.
Idea is to compute it using the divide and conquer technique.
Sort the points as values on x-axis and divide the plane through the midpoint. Compute the minimum distance in the left half and right half. Divide recursively so that in each half there are two points and distance can be calculated.

If the minimum distance pair can be in the left half or right half, then minimum distance,d, is minimum among both sides.



But there can be the case that the minimum distance pair lies in such a way that a point of the pair lies in the left side and another in the right side, then it can be a maximum 2d distance apart.



So those points sort them on Y-axis, compute the distance between two consecutive points lies on opposite sides, it take $O(n)$ time if there are n points in that strip.
The time complexity of this method is finding closest pair is $O(nlogn)$.