Lab 1: Divide and Conquer

1 Introduction

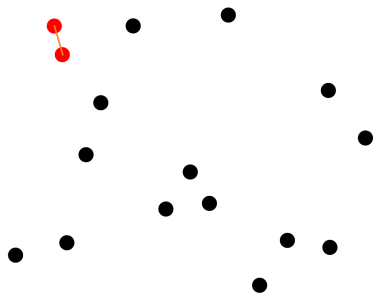
A divide-and-conquer algorithm recursively breaks down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.

The divide-and-conquer technique has a wide range of use cases. The technique is used in several sorting algorithms (for example merge sort), in the fast fourier transform (fft) algorithm, in the Strassen algorithm for matrix multiplication, ...

In this lab session we will solve two problems using the divide-and-conquer principle. The startcode is available on Ufora.

2 Assignment 1: Closest pair problem

The goal is to find the closest distance between two points in a random set of points. The following link, closest pair approach, provides an explanation on how to approach the closest pair problem using the divide-and-conquer technique. Using this approach, the problem will be solved with a time complexity of $O(\lg(n) * n \lg(n)) = O(n(\lg(n))^2)$. This could be further improved to $O(n \lg(n))$ but this is beyond the scope of this assignment.



Listed below are the subtasks required to get the algorithm running. For each of these subtasks, a unit test is written to verify if the code is working properly. The signature for the various functions is defined in the closestpair header file which can be found in the include folder.

1. Write a function to calculate the distance between two points.

2. Write a brute-force algorithm that will return the smallest distance from a set of points. At the end of the assignment, the brute-force technique can be compared to the divide-and-conquer technique.
3. As demonstrated in the explanation, a function is required that can find the closest pair in the strip between the two halves of the plane. Complete the closestPointsStrip function.
4. Finally write the closestPoints function that combines all previous functions to solve the closest pair problem.

3 Assignment 2: Median wage problem

A company is merging two branches. The CEO wants two know what the median wage will be of the employees after the merge. Two lists are given that represent the wages of the employees of the branches.

- First a class Employee has to be constructed that has a float value wage. In order to use this class in the getMedian function, several overloader functions will be required. Tests are provided for these overloaders.
- The getMedian function can be implemented in the medianwage header file.
- There are some constraints for the getMedian function:
  - The function is required to work in place, meaning no new vectors can be created. This also means that the function has a space complexity of O(1).
  - The function needs to work with different types: floats, doubles and the previously created Employee type.
  - The time complexity of the function has to be $O(\min(\lg(m), \lg(n))$ where m and n represent the size of the vectors.