

Project Report

Name: Minh Thang Cao

22 June 2020

1 Introduction

Implementation is an important step of every algorithm which helps us observe the algorithm's efficiency and behavior in practice. This report will briefly explain each part of the algorithm, show the program's implementation along with practical running time analysis and some implementation techniques used. The theoretical information in this report fully refers to the work of A. Maheshwari, W. Mulzer and M. Smid, see [1].

The whole closest pair algorithm consists of three important smaller parts:

1. Computing a separating annulus, denoted $\text{SEPANN}(S, n, d, \mu, c)$
2. The refinement of $\text{SEPANN}(S, n, d, \mu, c)$, denoted $\text{SPARSESEPANN}(S, n, d, t)$
3. The main recursive closest pair algorithm, denoted $\text{CLOSESTPAIR}(S, n, d)$

Throughout the paper, let:

- (P, dist) be a finite metric space in which P is the set of all points, and dist is the function that calculate the distance between any two points
- d be the space's doubling dimension
- S be a non-empty subset of P

2 The First Algorithm: Computing a separating annulus

An important part of the main closest pair algorithm is finding a separating annulus in the subset S . I will briefly describe this algorithm in the next subsection, due to A. Maheshwari, W. Mulzer and M. Smid [1, Section 3.1].

2.1 The $\text{SEPANN}(S, n, d, \mu, c)$ algorithm

In this section, $\mu \geq 1$ is a real constant number, c is calculated based on μ (I would say that $c = 2(4\mu)^d$ [1, Remark 1] since μ is not an integer in this case [1, Section 3.2]).

This algorithm picks a uniformly random point p from the subset S then finds the smallest ball centered at p , denoted $\text{ball}_S(p, R_p)$, that contains at least n/c point. If the outer ball $\text{ball}_S(p, \mu R_p)$ contains at most $n/2$ points, it returns p and R_p . If not, this procedure is repeated until the condition is satisfied. I will rewrite this algorithm's pseudocode below, from [1, Section 3.1]:

Algorithm 1: $\text{SEPANN}(S, n, d, \mu, c)$

```

repeat
  |  $p$  = a uniformly random point in  $S$ 
  |  $R_p = \min\{r > 0 : |\text{ball}_S(p, r)| \geq n/c\}$ 
until  $|\text{ball}_S(p, \mu R_p)| \geq n/c$ 
return  $p$  and  $R_p$ 

```

2.2 Finding the K^{th} smallest element

One step needed to be executed in $\text{SEPANN}(S, n, d, \mu, c)$ is to find the smallest ball which contains at least n/c points. This ball is easy to find using the k^{th} smallest element algorithm. Particularly, in the list of distances between p and all other points in S , we pick the $\lceil n/c \rceil$ -th smallest element, and let it be the radius of the ball we need to find. Thus, all points closer to p are inside this ball.

A very easy approach to find the k^{th} smallest element in a list is to sort it in ascending order, and then simply return the element at the k^{th} place. This sorting algorithm take $O(n \log n)$ time complexity in the worst case. Fortunately, we can improve the time complexity to $O(n)$ using a technique which is similar to Quicksort.

5 Implementation

This implementation of the closest pair doubling algorithm of A. Maheshwari, W. Mulzer and M. Smid [1] is written in C++ since it is a very common and fast programming language with high level supports of object-oriented programming that can help us organize the program efficiently

References

- [1] A. Maheshwari, W. Mulzer and M. Smid. *A Simple Randomized $O(n \log n)$ -Time Closest-Pair Algorithm in Doubling Metrics*, 2020. <https://arxiv.org/abs/2004.05883>