

**Tugas Kecil 2 IF2211 Strategi Algoritma  
Semester II Tahun 2022/2023**

**Penerapan Algoritma Divide and Conquer pada Closest  
Pair of Points Problem**

Disusun oleh:

Ammar Rasyad Chaeroel 13521136



**PROGRAM STUDI  
TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO  
DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG 2022**

## 1. ALGORITMA

1. Algoritma *Brute Force*
  - a. Jarak terdekat ditetapkan sebagai jarak antara point pertama dan point kedua sebagai patokan pertama
  - b. Kedua point terdekat disimpan pada variable masing-masing
  - c. Iterasikan tiap pasangan point
  - d. Jika memiliki jarak yang lebih dekat, maka jarak antara kedua point tersebut disimpan pada variable jarak terdekat bersama dengan pasangan point tersebut
  - e. Terdapat pasangan point terdekat
2. Algoritma *Divide and Conquer*
  - a. Semua point pada list point diurutkan berdasarkan koordinat pertama membesar
  - b. Point di tengah list yang sudah diurutkan dijadikan midpoint sebagai patokan
  - c. List point akan dibagi menjadi dua dengan size yang sama dan masing-masing dicari pasangan point terdekat
  - d. Terdapat 2 pasangan point terdekat yang kemudian akan dipilih jarak yang terdekat di antara kedua pasangan point tersebut
  - e. Membuat strip dimana jika jarak koordinat pertama dari point dengan midpoint lebih kecil dari jarak terdekat, maka akan dimasukkan ke list strip, membentuk *hyperplane* pada bidang ruang Euclidean  $R^n$
  - f. Point pada strip diurutkan berdasarkan koordinat kedua
  - g. Iterasikan semua point pada strip tersebut dan menghitung jarak terdekat dari point pada strip
  - h. Pasangan point dengan jarak terdekat telah terpilih
  - i. Langkah b-h akan dilakukan secara rekursif sampai dengan LEAF\_SIZE, dimana LEAF\_SIZE merupakan batas bawah algoritma *divide and conquer* untuk melakukan brute force dan memasuki tahap *conquer*

### CATATAN:

- Algoritma *Brute Force* dan *Divide and Conquer* dapat memberi hasil yang “terbalik” ( $\langle P1, P2 \rangle$  seharusnya ekuivalen dengan  $\langle P3, P4 \rangle$  namun bisa jadi seperti  $\langle P1, P2 \rangle$  dengan  $\langle P4, P3 \rangle$  dimana P4 dan P3 tertukar), hal ini merupakan *by design* dan tidak ditangani oleh program.
- 3D plotting hanya akan bisa dijalankan jika GNUPlot terinstal pada instalasi Windows (sesuai dengan requirement pada GitHub).
- Jika masukan plotting (y/yes or n/no) salah, maka program akan asumsi pilihan “no”. Ini merupakan *design choice*.
- Masukan n (jumlah point) tidak terbatas, namun minimal 2.
- Masukan dimensi dibataskan menjadi 1 sampai 10 (inklusif).

## 2. SOURCE CODE (C++)

main.cpp

```
#include <iostream>
```

```

#include <vector>
#include <chrono>
#include "point.h"
#include "rand.h"
#include "sort.h"
#include "plot.h"
#include "getcpu.h"

using namespace std;

const int LEAF_SIZE = 64; // leaf size for divide and conquer,
increase for more operations but slower, most optimal in my testing is
64

inline bool compareX(const Point& p1, const Point& p2, bool equals) {
    return equals ? p1.getCoordinate(0) <= p2.getCoordinate(0) :
p1.getCoordinate(0) < p2.getCoordinate(0);
}

inline bool compareY(const Point& p1, const Point& p2, bool equals) {
    return equals ? p1.getCoordinate(1) <= p2.getCoordinate(1) :
p1.getCoordinate(1) < p2.getCoordinate(1);
}

/**
 * @brief Brute force algorithm to find the closest pair of points
 * @param points
 * @return Closest pair of points
 */
Tuple<Point, Point> bruteForce(const vector<Point>& points) {
    double minDist = dist(points[0], points[1]);
    Point p1 = points[0];
    Point p2 = points[1];
    for (int i = 0; i < points.size(); i++) {
        for (int j = i + 1; j < points.size(); j++) {
            double distIJ = dist(points[i], points[j]);
            if (distIJ < minDist) {
                minDist = distIJ;
                p1 = points[i];
                p2 = points[j];
            }
        }
    }
    return Tuple<Point, Point>(p1, p2);
}

```

```

        }

    }

    }
    return {p1, p2};
}

/**
 * @brief Helper function for DnC
 * @param points
 * @return Closest pair of points
 */
Tuple<Point, Point> divAndConHelper(vector<Point>& points) {
    if (points.size() <= LEAF_SIZE) {
        return bruteForce(points);
    }
    const int dim = points[0].getDimension();
    Point midPoint = points[points.size() / 2];
    vector<Point> pointsXL(points.begin(), points.begin() +
points.size() / 2);
    vector<Point> pointsXR(points.begin() + points.size() / 2,
points.end());

    auto [p1, p2] = divAndConHelper(pointsXL);
    auto [p3, p4] = divAndConHelper(pointsXR);

    double d1 = dist(p1, p2);
    double d2 = dist(p3, p4);
    double minDist;
    Point p5(dim), p6(dim);
    if (d1 < d2) {
        p5 = p1;
        p6 = p2;
        minDist = d1;
    } else {
        p5 = p3;
        p6 = p4;
        minDist = d2;
    }

    vector<Point> strip;
    for (const auto & point : points) {

```

```

        if (abs(point.getCoordinate(0) - midPoint.getCoordinate(0)) <
minDist) {
            strip.push_back(point);
        }
    }

    //    timsort(strip, compareY);
    //    sort(strip.begin(), strip.end(), compareY);
    quickSort(strip, 0, strip.size() - 1, compareY);
    for (int i = 0; i < strip.size(); i++) {
        for (int j = i + 1; j < strip.size() &&
(strip[j].getCoordinate(1) - strip[i].getCoordinate(1)) < minDist;
j++) {
            double distIJ = dist(strip[i], strip[j]);
            if (distIJ < minDist) {
                minDist = distIJ;
                p5 = strip[i];
                p6 = strip[j];
            }
        }
    }
    return {p5, p6};
}

/**
 * @brief Divide and conquer algorithm to find the closest pair of
points
 * @param points
 * @return Closest pair of points
 */
Tuple<Point, Point> divideAndConquer(vector<Point>& points) {
    quickSort(points, 0, points.size() - 1, compareX);
    //    timsort(points, compareX);
    //    sort(points.begin(), points.end(), compareX);
    return divAndConHelper(points);
}

/**
 * @brief Driver function
 */

```

```

int main() {
    int n, dim;
    string input;
    cout << "Enter number of points (must be greater than or equal to
2): ";
    while (true) {
        getline(cin >> ws, input);
        if (strtol(input.c_str(), nullptr, 10) < 2) {
            cout << "Invalid input. Must be greater than or equal to
2: ";
        } else {
            break;
        }
    }
    n = stoi(input);

    cout << "Enter the dimension: ";
    while (true) {
        getline(cin >> ws, input);
        auto in = strtol(input.c_str(), nullptr, 10);
        if (in < 1 || in > 10) {
            cout << "Invalid input. Must be from 1 to 10 (inclusive):
";
        } else {
            break;
        }
    }
    dim = stoi(input);

    vector<Point> points(n, Point(dim));
    for (auto& p : points) {
        for (auto& c : p) {
            c = getRand();
        }
    }

    cout << "Dimension: " << points[0].getDimension() << endl;
    cout << "Number of points: " << points.size() << endl;

    cout << endl;
    cout << "Processor: " << getProcessorInfo() << endl;
}

```

```

cout << endl;

cout << "-----" << endl;
cout << "Brute force algorithm" << endl;
cout << "-----" << endl;

auto t1 = chrono::high_resolution_clock::now();
auto [p1, p2] = bruteForce(points);
auto t2 = chrono::high_resolution_clock::now();

cout << "Number of operations: " << operations << endl;
cout << "Closest pair of points: " << p1 << " and " << p2 << endl;
cout << "Distance: " << dist(p1, p2) << endl << endl;

auto duration = chrono::duration<double, milli>(t2 - t1).count();
cout << duration << " ms" << endl;

cout << "-----" << endl;
cout << "Divide and conquer algorithm" << endl;
cout << "-----" << endl;

operations = 0;

t1 = chrono::high_resolution_clock::now();
auto [p3, p4] = divideAndConquer(points);
t2 = chrono::high_resolution_clock::now();

cout << "Number of operations: " << operations << endl;
cout << "Closest pair of points: " << p3 << " and " << p4 << endl;
cout << "Distance: " << dist(p3, p4) << endl << endl;

duration = chrono::duration<double, milli>(t2 - t1).count();
cout << duration << " ms" << endl;

// Ask to save to file and plot
if (dim == 3){
    cout << "Do you want to save the points to a file and plot
them? (y/n): ";
    getline(cin >> ws, input);

```

```

        transform(input.begin(), input.end(),
input.begin(), ::tolower);
        if (input == "y" || input == "yes") {
            saveToFile("points.txt", points, p1, p2);
            startPlot("points.txt");
        } else if (input == "n" || input == "no") {
            cout << "Exiting..." << endl;
        } else {
            cout << "Invalid input. Exiting..." << endl;
        }
    }
    return 0;
}

```

#### plot.cpp

```

//
// Created by ammar on 27/02/2023.
//

#include <fstream>
#include "plot.h"

/**
 * Saves the given points to a file
 * @param filename
 * @param points
 * @param p1
 * @param p2
 */
void saveToFile(const string& filename, const vector<Point>& points,
const Point& p1, const Point& p2) {
    ofstream file(filename);
    for (const Point& point : points) {
        for (auto& coord : point) {
            file << coord << " ";
        }
        if (point == p1 || point == p2) {
            file << 1;
        } else {

```



```

        file << 0;
    }
    file << endl;
}
file.close();
}

/**
 * Starts plotting the points in the given file
 * @param file
 */
void startPlot(const string& file = "points.txt") {
    cout << "Plotting..." << endl;
    FILE* pipe = popen("gnuplot -persist", "w");
    if (!pipe) {
        throw runtime_error("Could not open pipe to GNUPlot");
    }
    fprintf(pipe, "set title \"Nearest Points\\n\\n");
    fprintf(pipe, "splot \"%s\" using 1:2:3:($4==1 ? $3 : 1/0) with points pointtype 7, \"%s\" using 1:2:3:($4==0 ? $3 : 1/0) with points pointtype 5 \\n", file.c_str(), file.c_str());
    pclose(pipe);
    cout << "Plotting done!" << endl;
}

```

plot.h

```

//
// Created by ammar on 27/02/2023.
//

#ifndef TUCIL2_13521136_PLOT_H
#define TUCIL2_13521136_PLOT_H

#include <string>
#include <vector>
#include "point.h"

void saveToFile(const string& filename, const vector<Point>& points,
const Point& p1, const Point& p2);

```

```

void startPlot(const string& file);

#endif

```

point.h

```

#ifndef TUCIL2_13521136_POINT_H
#define TUCIL2_13521136_POINT_H

#include <iostream>
#include <string>
#include <cmath>

using namespace std;

static long long operations;

// n-dimensional point
class Point {
public:
    explicit Point(int n) : dimension(n), coordinates(n, 0) {}
    Point(int n, const vector<double>& coordinates) : dimension(n),
coordinates(coordinates) {}

    [[nodiscard]] int getDimension() const { return dimension; }
    [[nodiscard]] double getCoordinate(int i) const { return
coordinates[i]; }
    [[nodiscard]] const vector<double>& getCoordinates() const
{ return coordinates; }

    bool operator==(const Point& p) const {
        for (int i = 0; i < dimension; i++) {
            if (coordinates[i] != p.coordinates[i]) {
                return false;
            }
        }
        return true;
    }

    Point operator+(const Point& p) const {

```

```

        vector<double> newCoordinates(dimension);
        for (int i = 0; i < dimension; i++) {
            newCoordinates[i] = coordinates[i] + p.coordinates[i];
        }
        return {dimension, newCoordinates};
    }

    Point operator-(const Point& p) const {
        vector<double> newCoordinates(dimension);
        for (int i = 0; i < dimension; i++) {
            newCoordinates[i] = coordinates[i] - p.coordinates[i];
        }
        return {dimension, newCoordinates};
    }

    explicit operator string() const {
        string result = "(";
        for (int i = 0; i < dimension; i++) {
            result += std::to_string(coordinates[i]);
            if (i < dimension - 1) {
                result += ", ";
            }
        }
        result += ")";
        return result;
    }

    friend ostream& operator<<(ostream& os, const Point& p) {
        os << (string) p;
        return os;
    }

    vector<double>::iterator begin() { return coordinates.begin(); }
    vector<double>::iterator end() { return coordinates.end(); }
    [[nodiscard]] vector<double>::const_iterator begin() const
    { return coordinates.begin(); }
    [[nodiscard]] vector<double>::const_iterator end() const { return
coordinates.end(); }
private:
    int dimension;

```

```

        vector<double> coordinates;
    };

    // I want to use std::tuple, but I'm scared of breaking the rules :")
    template <typename T1, typename T2>
    class Tuple {
    public:
        T1 first;
        T2 second;
    };

    inline double dist(const Point& p1, const Point& p2) {
        double result = 0;
        for (int i = 0; i < p1.getDimension(); i++) {
            result += pow(p1.getCoordinate(i) - p2.getCoordinate(i), 2);
        }
        operations++;
        return sqrt(result);
    }

#endif

```

sort.h

```

//
// Created by ammar on 2/26/23.
//

#ifndef TUCIL2_13521136_SORT_H
#define TUCIL2_13521136_SORT_H

#include <vector>
#include "point.h"

#define RUN 32

using namespace std;

inline double min(double a, double b) {
    return a < b ? a : b;
}

```

```

}

void swap(Point &a, Point &b) {
    Point temp = a;
    a = b;
    b = temp;
}

void insertionSort(vector<Point>& points, int left, int right, bool
(*compare)(const Point&, const Point&)) {
    for (int i = left + 1; i <= right; i++) {
        const Point& temp = points[i];
        int j = i - 1;
        while (j >= left && compare(temp, points[j])) {
            points[j + 1] = points[j--];
        }
        points[j + 1] = temp;
    }
}

void merge(vector<Point>& points, int l, int m, int r, bool
(*compare)(const Point&, const Point&)) {
    int left = m - l + 1;
    int right = r - m;
    vector<Point> L(left, Point(points[0].getDimension())), R(right,
Point(points[0].getDimension()));
    for (int i = 0; i < left; i++) {
        L[i] = points.at(l + i);
    }
    for (int i = 0; i < right; i++) {
        R[i] = points.at(m + 1 + i);
    }
    int i = 0, j = 0, k = l;

    while (i < left && j < right) {
        if (L[i].getCoordinate(0) <= R[j].getCoordinate(0)) {
            points[k++] = L[i++];
        } else {
            points[k++] = R[j++];
        }
    }
}

```

```

    }
    while (i < left) {
        points[k++] = L[i++];
    }
    while (j < right) {
        points[k++] = R[j++];
    }
}

// WHY DOES THIS NOT WORK IT WORKED ON MY PREVIOUS PROGRAMS
void timsort(vector<Point>& points, bool (*compare)(const Point&,
const Point&)) {
    int n = points.size();
    for (int i = 0; i < n; i += RUN) {
        insertionSort(points, i, min((i + RUN - 1), (n - 1)),
compare);
    }
    for (int size = RUN; size < n; size *= 2) {
        for (int left = 0; left < n; left += 2 * size) {
            int mid = left + size - 1;
            int right = min((left + 2 * size - 1), (n - 1));
            if (mid < right) merge(points, left, mid, right,
compare);
        }
    }
}

int partition(vector<Point>& points, int start, int end, bool
(*compare)(const Point&, const Point&, bool)) {
    Point pivot = points[start];
    int count = 0;
    for (int i = start + 1; i <= end; i++) {
        if (compare(points[i], pivot, true)) {
            count++;
        }
    }

    int index = start + count;
    swap(points[start], points[index]);
    int i = start, j = end;

```

```

        while (i < index && j > index) {
            while (compare(points[i], pivot, true)) i++;
            while (compare(pivot, points[j], false)) j--;
            if (i < index && j > index) {
                swap(points[i++], points[j--]);
            }
        }
        return index;
    }
}

/**
 * Quick sort algorithm
 * @param points
 * @param start
 * @param end
 * @param compare
 */
void quickSort(vector<Point>& points, int start, int end, bool
(*compare)(const Point&, const Point&, bool)) {
    if (start < end) {
        int p = partition(points, start, end, compare);
        quickSort(points, start, p - 1, compare);
        quickSort(points, p + 1, end, compare);
    }
}

#endif

```

rand.h

```

//
// Created by ammar on 2/26/23.
//

#ifndef TUCIL2_13521136_RAND_H
#define TUCIL2_13521136_RAND_H

/**
 * Generates a random number using the RDRAND instruction
 * @param bound

```

```

    * @return Random number
    */
double getRand(long double bound = 1e+18 /* Literally just a random
number so it's not huge */) {
    unsigned long long result;
    __builtin_ia32_rdrand64_step(&result);
    return (double) (result / bound);
}

#endif

```

## getcpu.h

```

//
// Created by ammar on 01/03/2023.
//

#ifndef TUCIL2_13521136_GETCPU_H
#define TUCIL2_13521136_GETCPU_H

#ifdef _WIN32
#include <intrin.h>
#endif

#ifdef linux
#include <cpuid.h>
#endif

#include <string>

using namespace std;

string getProcessorInfo() {
    string result;
#ifdef _WIN32
    int CPUInfo[4] = {-1};
    unsigned nExIds, i = 0;
    char CPUBrandString[0x40];
    __cpuid(CPUInfo, 0x80000000);
    nExIds = CPUInfo[0];

```



```

    for (i = 0x80000000; i <= nExIds; ++i) {
        __cpuid(CPUInfo, i);
        if (i == 0x80000002)
            memcpy(CPUBrandString, CPUInfo, sizeof(CPUInfo));
        else if (i == 0x80000003)
            memcpy(CPUBrandString + 16, CPUInfo, sizeof(CPUInfo));
        else if (i == 0x80000004)
            memcpy(CPUBrandString + 32, CPUInfo, sizeof(CPUInfo));
    }
    result = CPUBrandString;
#endif
#ifdef linux
    unsigned int eax, ebx, ecx, edx;
    eax = 0x80000000;
    __get_cpuid(eax, &eax, &ebx, &ecx, &edx);
    unsigned int nExIds = eax;
    char CPUBrandString[0x40];
    for (i = 0x80000000; i <= nExIds; ++i) {
        eax = i;
        __get_cpuid(eax, &eax, &ebx, &ecx, &edx);
        if (i == 0x80000002)
            memcpy(CPUBrandString, &eax, sizeof(eax));
        else if (i == 0x80000003)
            memcpy(CPUBrandString + 16, &eax, sizeof(eax));
        else if (i == 0x80000004)
            memcpy(CPUBrandString + 32, &eax, sizeof(eax));
    }
    result = CPUBrandString;
#endif
    return result;
}

#endif

```

### 3. CONTOH MASUKAN DAN LUARAN

Semua contoh dijalankan pada komputer dengan processor Intel Core i7 10750H @ 2.6GHz (5GHz single core).

### Dimensi 3

Test case (ukuran N)	Screenshot
16	<pre> Enter number of points (must be greater than or equal to 2): 16 Enter the dimension: 3 Dimension: 3 Number of points: 16  Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz  ----- Brute force algorithm ----- Number of operations: 121 Closest pair of points: (3.498118, 15.387722, 6.951052) and (1.940007, 17.433009, 6.055403) Distance: 2.7227 0.0262 ms  ----- Divide and conquer algorithm ----- Number of operations: 121 Closest pair of points: (1.940007, 17.433009, 6.055403) and (3.498118, 15.387722, 6.951052) Distance: 2.7227 0.0269 ms Do you want to save the points to a file and plot them? (y/n): n Exiting... </pre>
64	<pre> Enter number of points (must be greater than or equal to 2): 64 Enter the dimension: 3 Dimension: 3 Number of points: 64  Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz  ----- Brute force algorithm ----- Number of operations: 2017 Closest pair of points: (14.981242, 14.575138, 8.227813) and (15.195963, 14.270376, 8.320260) Distance: 0.384098 0.3168 ms  ----- Divide and conquer algorithm ----- Number of operations: 2017 Closest pair of points: (14.981242, 14.575138, 8.227813) and (15.195963, 14.270376, 8.320260) Distance: 0.384098 0.342 ms Do you want to save the points to a file and plot them? (y/n): n Exiting... </pre>
128	<pre> Enter number of points (must be greater than or equal to 2): 128 Enter the dimension: 3 Dimension: 3 Number of points: 128  Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz  ----- Brute force algorithm ----- Number of operations: 8129 Closest pair of points: (4.421475, 6.032028, 0.432975) and (4.069461, 5.549784, 0.092123) Distance: 0.687497 1.2401 ms  ----- Divide and conquer algorithm ----- Number of operations: 4043 Closest pair of points: (4.069461, 5.549784, 0.092123) and (4.421475, 6.032028, 0.432975) Distance: 0.687497 0.7362 ms Do you want to save the points to a file and plot them? (y/n): n Exiting... </pre>

1000	<pre> Enter number of points (must be greater than or equal to 2): 1000 Enter the dimension: 3 Dimension: 3 Number of points: 1000  Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz  ----- Brute force algorithm ----- Number of operations: 499501 Closest pair of points: (1.397233, 1.697325, 17.203244) and (1.382063, 1.705219, 17.130886) Distance: 0.0743517 76.1819 ms  ----- Divide and conquer algorithm ----- Number of operations: 31078 Closest pair of points: (1.382063, 1.705219, 17.130886) and (1.397233, 1.697325, 17.203244) Distance: 0.0743517 6.5043 ms Do you want to save the points to a file and plot them? (y/n): n Exiting... </pre>
Masukan yang tidak sesuai	<pre> Enter number of points (must be greater than or equal to 2): 10a00 Enter the dimension: 3 Dimension: 3 Number of points: 10  Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz  ----- Brute force algorithm ----- Number of operations: 46 Closest pair of points: (11.017740, 12.479443, 14.071938) and (14.097717, 14.672246, 12.553024) Distance: 4.07452 0.0143 ms  ----- Divide and conquer algorithm ----- Number of operations: 46 Closest pair of points: (11.017740, 12.479443, 14.071938) and (14.097717, 14.672246, 12.553024) Distance: 4.07452 0.0133 ms Do you want to save the points to a file and plot them? (y/n): n Exiting... </pre>
	<pre> Enter number of points (must be greater than or equal to 2): 1 Invalid input. Must be greater than or equal to 2: 1000 Enter the dimension: 0 Invalid input. Must be from 1 to 10 (inclusive): 3 Dimension: 3 Number of points: 1000  Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz  ----- Brute force algorithm ----- Number of operations: 499501 Closest pair of points: (4.661481, 9.302379, 5.469532) and (4.758716, 9.360383, 5.397924) Distance: 0.133966 75.9189 ms  ----- Divide and conquer algorithm ----- Number of operations: 30926 Closest pair of points: (4.661481, 9.302379, 5.469532) and (4.758716, 9.360383, 5.397924) Distance: 0.133966 6.717 ms Do you want to save the points to a file and plot them? (y/n): n Exiting... </pre>

Dimensi  $n > 3$  (contoh yang dipakai adalah dimensi 10)

Test case (ukuran N)	Screenshot
16	<pre> Enter number of points (must be greater than or equal to 2): 16 Enter the dimension: 10 Dimension: 10 Number of points: 16  Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz  ----- Brute force algorithm ----- Number of operations: 121 Closest pair of points: (10.078845, 18.357886, 12.534344, 3.006379, 3.752638, 8.026691, 7.226141, 10.726669, 9.078306, 14.511348) and (9.913325, 12.277278, 11.249827, 6.524814, 6.652230, 5.272032, 4.426205, 8.651846, 12.886110, 15.192349) Distance: 9.70227 0.0634 ms  ----- Divide and conquer algorithm ----- Number of operations: 121 Closest pair of points: (9.913325, 12.277278, 11.249827, 6.524814, 6.652230, 5.272032, 4.426205, 8.651846, 12.886110, 15.192349) and (10.078845, 18.357886, 12.534344, 3.006379, 3.752638, 8.026691, 7.226141, 10.726669, 9.078306, 14.511348) Distance: 9.70227 0.065 ms </pre>
64	<pre> Enter number of points (must be greater than or equal to 2): 64 Enter the dimension: 10 Dimension: 10 Number of points: 64  Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz  ----- Brute force algorithm ----- Number of operations: 2017 Closest pair of points: (2.435927, 10.787817, 18.074857, 6.823902, 14.910959, 3.306583, 8.332226, 12.648748, 10.658381, 17.201347) and (0.899857, 6.855713, 15.690084, 8.577708, 13.079801, 5.391012, 8.546338, 11.395181, 14.924682, 14.908886) Distance: 7.70428 0.9121 ms  ----- Divide and conquer algorithm ----- Number of operations: 2017 Closest pair of points: (0.899857, 6.855713, 15.690084, 8.577708, 13.079801, 5.391012, 8.546338, 11.395181, 14.924682, 14.908886) and (2.435927, 10.787817, 18.074857, 6.823902, 14.910959, 3.306583, 8.332226, 12.648748, 10.658381, 17.201347) Distance: 7.70428 0.9437 ms </pre>
128	<pre> Enter number of points (must be greater than or equal to 2): 128 Enter the dimension: 10 Dimension: 10 Number of points: 128  Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz  ----- Brute force algorithm ----- Number of operations: 8129 Closest pair of points: (2.609041, 9.339022, 14.931640, 8.194200, 11.858752, 4.226243, 8.667690, 14.948171, 14.615452, 1.406323) and (0.545802, 9.019314, 15.482641, 9.829320, 13.152789, 6.542169, 9.553836, 17.851957, 14.991983, 5.118894) Distance: 6.12508 3.6321 ms  ----- Divide and conquer algorithm ----- Number of operations: 6028 Closest pair of points: (0.545802, 9.019314, 15.482641, 9.829320, 13.152789, 6.542169, 9.553836, 17.851957, 14.991983, 5.118894) and (2.609041, 9.339022, 14.931640, 8.194200, 11.858752, 4.226243, 8.667690, 14.948171, 14.615452, 1.406323) Distance: 6.12508 2.9111 ms </pre>
1000	<pre> Enter number of points (must be greater than or equal to 2): 1000 Enter the dimension: 10 Dimension: 10 Number of points: 1000  Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz  ----- Brute force algorithm ----- Number of operations: 499501 Closest pair of points: (1.699215, 6.548788, 7.635439, 2.240172, 16.678067, 8.873968, 9.280033, 11.359031, 7.747741, 1.606483) and (1.822765, 6.368387, 7.663864, 2.007743, 14.526741, 9.524518, 10.401304, 8.709648, 5.608468, 1.050861) Distance: 4.27969 222.825 ms  ----- Divide and conquer algorithm ----- Number of operations: 278174 Closest pair of points: (1.699215, 6.548788, 7.635439, 2.240172, 16.678067, 8.873968, 9.280033, 11.359031, 7.747741, 1.606483) and (1.822765, 6.368387, 7.663864, 2.007743, 14.526741, 9.524518, 10.401304, 8.709648, 5.608468, 1.050861) Distance: 4.27969 131.421 ms </pre>

1000 0	<pre> Enter number of points (must be greater than or equal to 2): 10000 Enter the dimension: 10 Dimension: 10 Number of points: 10000  Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz  ----- Brute force algorithm ----- Number of operations: 49995001 Closest pair of points: (14.129969, 2.383146, 9.957376, 4.695121, 0.645960, 11.147245, 4.569764, 13.119425, 12.042313, 10.330694) and (13.154459, 3.087775, 9.554614, 6.110019, 0.952933, 12.576642, 5.183032, 12.977170, 13.399337, 10.758161) Distance: 2.85837 22284.9 ms  ----- Divide and conquer algorithm ----- Number of operations: 14885612 Closest pair of points: (14.129969, 2.383146, 9.957376, 4.695121, 0.645960, 11.147245, 4.569764, 13.119425, 12.042313, 10.330694) and (13.154459, 3.087775, 9.554614, 6.110019, 0.952933, 12.576642, 5.183032, 12.977170, 13.399337, 10.758161) Distance: 2.85837 6818.84 ms </pre>
-----------	--

#### 4. LINK GITHUB

Link Repo GitHub: [https://github.com/ammарasyad/Tucil2\\_13521136](https://github.com/ammарasyad/Tucil2_13521136)

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil <i>running</i>	✓	
3. Program dapat menerima masukan dan menuliskan luaran	✓	
4. Luaran program sudah benar (solusi <i>closest pair</i> benar)	✓	
5. Bonus 1 dikerjakan	✓	
6. Bonus 2 dikerjakan	✓	