LAPORAN TUGAS KECIL 2

IF 2211 STRATEGI ALGORITMA

SEMESTER 2 2022 – 2023

Mencari Pasangan Titik Terdekat 3D dengan Algoritma Divide and Conquer

Disusun Oleh :

Ezra M C M H (13521073)

PROGRAM STUDI TEKNIK ELEKTRO DAN INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

1. Spesifikasi Tugas

Mencari sepasang titik terdekat dengan Algoritma Divide and Conquer sudah dijelaskan di dalam kuliah. Persoalan tersebut dirumuskan untuk titik pada bidang datar (2D). Pada Tucil 2 kali ini Anda diminta mengembangkan algoritma mencari sepasang titik terdekat pada bidang 3D. Misalkan terdapat n buah titik pada ruang 3D. Setiap titik P di dalam ruang dinyatakan dengan koordinat P = (x, y, z). Carilah sepasang titik yang mempunyai jarak terdekat satu sama lain. Jarak dua buah titk P1 = (x1, y1, z1) dan P2 = (x2, y2, z2) dihitung dengan rumus Euclidean berikut:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Buatlah program dalam dalam Bahasa C/C++/Java/Python/Golang/Ruby/Perl (pilih salah satu) untuk mencari sepasang titik yang jaraknya terdekat datu sama lain dengan menerapkan algoritma divide and conquer untuk penyelesaiannya, dan perbandingannya dengan Algoritma Brute Force. Masukan program: - n - titik-titik (dibangkitkan secara acak) dalam koordinat (x, y, z) Luaran program - sepasang titik yang jaraknya terdekat dan nilai jaraknya - banyaknya operasi perhitungan rumus Euclidian - waktu riil dalam detik (spesifikasikan komputer yang digunakan) - Bonus 1 (Nilai = 7,5) penggambaran semua titik dalam bidang 3D, sepasang titik yang jaraknya terdekat ditunjukkan dengan warna yang berbeda dari titik lainnya. Bonus 2 (nilai = 7,5): Generalisasi program anda sehingga dapat mencari sepasang titik terdekat untuk sekumpulan vektor di Rn , setiap vektor dinyatakan dalam bentuk x = (x1, x2, …, xn)

2. Algoritma Divide and Conquer

Algoritma “Divide and Conquer” adalah suatu pendekatan untuk memecahkan masalah yang kompleks dengan membaginya menjadi beberapa bagian yang lebih kecil dan kemudian memecahkan setiap bagian tersebut secara terpisah, sebelum akhirnya menggabungkan hasilnya menjadi solusi akhir yang lengkap.

Pendekatan "Divide and Conquer" terdiri dari tiga langkah utama:

1. Divide (Membagi): Masalah besar dibagi menjadi beberapa bagian yang lebih kecil yang dapat diselesaikan dengan lebih mudah. Pemecahan masalah ini dilakukan secara rekursif hingga mencapai masalah dasar yang sederhana dan mudah diselesaikan.
2. Conquer (Menaklukkan): Setiap bagian kecil dari masalah tersebut diselesaikan secara independen menggunakan pendekatan yang sama. Jika masalah dasar sudah cukup sederhana, maka masalah dapat diselesaikan secara langsung tanpa lagi membaginya.
3. Combine (Menggabungkan): Setelah setiap bagian kecil terselesaikan, hasilnya digabungkan untuk menghasilkan solusi akhir untuk masalah awal.

Algoritma "Divide and Conquer" digunakan dalam banyak aplikasi komputasi, seperti dalam algoritma pengurutan data, perhitungan nilai perkalian matriks, dan dalam pengembangan teknik pemrograman dinamis. Dengan membagi masalah menjadi bagian yang lebih kecil dan menyelesaikannya secara independen, algoritma "Divide and Conquer" dapat mempercepat pemrosesan dan menghasilkan solusi yang lebih efisien untuk masalah kompleks.

3. Source Code Program

Main.cpp

```cpp
#include <iostream>
#include <cmath>
#include <ctime>
#include <cfloat>

using namespace std;

struct Point {
    double x, y, z;
};

double euclidean_distance(Point p1, Point p2) {
    double dx = p1.x - p2.x;
    double dy = p1.y - p2.y;
    double dz = p1.z - p2.z;
    return sqrt(dx*dx + dy*dy + dz*dz);
}

double closest_pair(Point P[], int n, int& count) {
    if (n <= 3) {
        double min_dist = DBL_MAX;
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                double d = euclidean_distance(P[i], P[j]);
                count++;
                if (d < min_dist) {
                    min_dist = d;
                }
            }
        }
        return min_dist;
    }

    int mid = n / 2;
    Point mid_point = P[mid];

    double dleft = closest_pair(P, mid, count);
    double dright = closest_pair(P + mid, n - mid, count);

    double d = min(dleft, dright);

```

```
    Point strip[n];
    int j = 0;
    for (int i = 0; i < n; i++) {
        if (abs(P[i].x - mid_point.x) < d) {
            strip[j] = P[i];
            j++;
        }
    }

    double min_strip = d;
    for (int i = 0; i < j; i++) {
        for (int k = i + 1; k < j && (strip[k].y - strip[i].y) < min_strip; k++) {
            double d_strip = euclidean_distance(strip[i], strip[k]);
            count++;
            if (d_strip < min_strip) {
                min_strip = d_strip;
            }
        }
    }
    return min(d, min_strip);
}

int main() {
    int n;
    int a;

    cout << "Pada program ini koordinat berada di rentang 0 < x,y,z < 10000" << endl;
    cout << "Minimal banyaknya titik yang dimasukkan 2 \n\n";
    cout << "Masukkan banyaknya titik: ";
    cin >> n;
    cout << endl;

    a = n;
    Point points[n];
    srand(time(0));
    for (int i = 0; i < n; i++) {
        points[i].x = ((float)rand()) / (float)RAND_MAX * 10000;
        points[i].y = ((float)rand()) / (float)RAND_MAX * 10000;
        points[i].z = ((float)rand()) / (float)RAND_MAX * 10000;
    }

```

```
    int count1 = 0;
    double min_distance = INFINITY;
    int i_min, j_min;

    clock_t strt = clock();
    for (int i = 0; i < a-1; i++) {
        for (int j = i+1; j < a; j++) {
            double dist = euclidean_distance(points[i], points[j]);
            count1++;
            if (dist < min_distance) {
                min_distance = dist;
                i_min = i;
                j_min = j;
            }
        }
    }
    clock_t stop = clock();

    clock_t start = clock();
    int count = 0;
    double min_dist = closest_pair(points, n, count);
    clock_t end = clock();

    cout << "Algoritma divide and conquer "<< endl ;
    cout << "Titik terdekat adalah ( " <<points[i_min].x << ", " << points[i_min].y << ", " << points[i_min].z << ") dan ("
        << points[j_min].x << ", " << points[j_min].y << ", " << points[j_min].z << ") dengan jarak : "<< min_distance << endl;
    cout << "Banyaknya operasi perhitungan: " << count << endl;
    cout << "Waktu eksekusi: " << double(end - start) / CLOCKS_PER_SEC * 1000 << " ms" << endl << endl;

    cout << "Algoritma Brute Force "<< endl ;
    cout << "Titik terdekat adalah ( " <<points[i_min].x << ", " << points[i_min].y << ", " << points[i_min].z << ") dan ("
        << points[j_min].x << ", " << points[j_min].y << ", " << points[j_min].z << ") dengan jarak : "<< min_distance << endl;
    cout << "Banyaknya operasi perhitungan: " << count1 << endl;
    cout << "Waktu eksekusi: " << double(stop-strt) / CLOCKS_PER_SEC * 1000 << " ms" << endl << endl;

    cout<< "Processor komputer yang digunakan AMD Ryzen 7 5800H ";
    return 0;
}
```

4. Screenshot

a. input n = 16

```
Pada program ini koordinat berada di rentang 0 < x,y,z < 10000
Minimal banyaknya titik yang dimasukkan 2

Masukkan banyaknya titik: 16

Algoritma divide and conquer
Titik terdekat adalah ( 261.849, 7194.43, 4036.68) dan (451.979, 7424.54, 4612.57) dengan jarak : 648.647
Banyaknya operasi perhitungan: 27
Waktu eksekusi: 0 ms

Algoritma Brute Force
Titik terdekat adalah ( 261.849, 7194.43, 4036.68) dan (451.979, 7424.54, 4612.57) dengan jarak : 648.647
Banyaknya operasi perhitungan: 120
Waktu eksekusi: 0 ms

Processor komputer yang digunakan AMD Ryzen 7 5800H
```

Waktu eksekusi 0 ms dikarenakan waktu eksekusi yang sangat kecil

b. input n = 64

```
Pada program ini koordinat berada di rentang 0 < x,y,z < 10000
Minimal banyaknya titik yang dimasukkan 2

Masukkan banyaknya titik: 64

Algoritma divide and conquer
Titik terdekat adalah ( 1399.88, 9095.43, 4286.94) dan (1026.64, 9271.22, 4213.39) dengan jarak : 419.07
Banyaknya operasi perhitungan: 159
Waktu eksekusi: 0 ms

Algoritma Brute Force
Titik terdekat adalah ( 1399.88, 9095.43, 4286.94) dan (1026.64, 9271.22, 4213.39) dengan jarak : 419.07
Banyaknya operasi perhitungan: 2016
Waktu eksekusi: 0 ms

Processor komputer yang digunakan AMD Ryzen 7 5800H
```

Waktu eksekusi 0 ms dikarenakan waktu eksekusi yang sangat kecil

c. input n = 128

```
Pada program ini koordinat berada di rentang 0 < x,y,z < 10000
Minimal banyaknya titik yang dimasukkan 2

Masukkan banyaknya titik: 128

Algoritma divide and conquer
Titik terdekat adalah ( 4628.74, 2813.81, 8552.81) dan (4674.52, 2713.4, 8699.91) dengan jarak : 183.889
Banyaknya operasi perhitungan: 372
Waktu eksekusi: 0 ms

Algoritma Brute Force
Titik terdekat adalah ( 4628.74, 2813.81, 8552.81) dan (4674.52, 2713.4, 8699.91) dengan jarak : 183.889
Banyaknya operasi perhitungan: 8128
Waktu eksekusi: 1 ms

Processor komputer yang digunakan AMD Ryzen 7 5800H
```

Waktu eksekusi 0 ms dikarenakan waktu eksekusi yang sangat kecil

d. input n = 1000

```
Pada program ini koordinat berada di rentang 0 < x,y,z < 10000
Minimal banyaknya titik yang dimasukkan 2

Masukkan banyaknya titik: 1000

Algoritma divide and conquer
Titik terdekat adalah ( 3065.58, 1442, 4395.28) dan (3008.82, 1439.25, 4374.83) dengan jarak : 60.3974
Banyaknya operasi perhitungan: 4265
Waktu eksekusi: 0 ms

Algoritma Brute Force
Titik terdekat adalah ( 3065.58, 1442, 4395.28) dan (3008.82, 1439.25, 4374.83) dengan jarak : 60.3974
Banyaknya operasi perhitungan: 499500
Waktu eksekusi: 35 ms

Processor komputer yang digunakan AMD Ryzen 7 5800H
```

Waktu eksekusi 0 ms dikarenakan waktu eksekusi yang sangat kecil

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

Link Repository : https://github.com/ezramcmh/Tucil2_13521073