TUGAS KECIL 1
IF2211 STRATEGI ALGORITMA

CONVEX HULL ALGORITHM BRUTE FORCE

Disusun oleh:

Jones Napoleon
13518086

Mata Kuliah Strategi Algoritma
Semester 2 Tahun Ajaran 2019/2020

Prodi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung 2020

DAFTAR ISI

DAFTAR ISI

BAB I

CONVEX HULL DAN ALGORITMA BRUTE FORCE

1.1. *OVERVIEW*

Convex Hull merupakan himpunan *convex* terkecil yang mengandung atau mengenkapsulasi semua himpunan titik yang ada, biasanya merupakan titik-titik yang berada pada "lapisan terluar" himpunan titik tersebut. Pada program ini, semua antar-titik diasumsi dihubungkan hanya dengan garis lurus.

Secara dasar, program menerima *input* berupa jumlah titik (anggap *N* buah), kemudian program akan menggenerasikan titik-titik sebanyak *N* kali. Setelah itu, program akan memproses benar tidaknya masing-masing kombinasi dua titik (satu garis) merupakan titik yang sesuai dengan *Convex Hull*. Kumpulan titik-titik yang sesuai tersebut kemudian dikembalikan.

1.2. STRUKTUR DATA

Secara intuitif, program membutuhkan model berupa *point* (titik) yang mempunyai struktur data *absis* (X) dan *ordinat* (Y) – keduanya bertipe *float*. Selain itu, program juga memerlukan model berupa *array of point* yang akan menampung semua titik baik setelah digenerasi secara acak maupun setelah diproses dan siap dikembalikan. Mengikuti konvensi Institut Teknologi Bandung, indeks di *array* dimulai dari 1.

```
typedef struct {
    float X;
    float Y;
} POINT;

#define Ordinat(P) (P).Y
#define Absis(P) (P).X
```

Gambar 1. Struktur Data *POINT*

```
typedef struct {
  POINT *TI;
  int Neff;
  int MaxEl;
} TabPOINT;

#define Neff(T) (T).Neff
#define TI(T) (T).TI
#define Elmt(T, i) (T).TI[(i)]
#define MaxEl(T) (T).MaxEl
```

Gambar 2. Struktur Data *array of POINT*

1.3. ALGORITMA

Program setelah menyimpan kumpulan titik pada suatu *array* (*array_init*) dengan cara digenerasi secara acak, selanjutnya akan melakukan iterasi secara kombinatorial. Misalkan ada *N* buah titik. Program akan mulai dari titik di *array* berindeks 1 dengan melakukan pengecekan titik di *array* berindeks 1 dengan titik di *array* berindeks 2. Kemudian titik di *array* berindeks 1 dengan titik di *array* berindeks 3, hingga titik di *array* berindeks 1 dengan titik di *array* berindeks N. Satu iterasi elemen *array* selesai. Kemudian titik di *array* berindeks 2 dengan titik di *array* berindeks 3, sampai dengan

titik di *array* berindeks *N*, hingga akhirnya kombinasi titik di *array* berindeks *N*-1 dengan titik di *array* berindeks *N*.

Pengecekan dua titik diatas dilakukan dengan melakukan persamaan garis:

$$y = b + m\ (x - a)$$

, dimana ***m*** merupakan gradien kedua titik tersebut dengan ***b*** dan ***a*** secara berturut-turut merupakan ordinat dan absis dari salah satu titik tersebut.

Program kemudian mengiterasi semua titik selain kedua titik percobaan tersebut dengan cara disubstitusikan nilai absisnya ke persamaan tersebut. Jika nilai ***y*** yang dihasilkan lebih besar dari nilai ordinat titik tersebut, maka program akan mengembalikan ***true***, atau ***false*** jika sebaliknya (konvensi penulis). Nilai ***false*** atau ***true*** ini kemudian akan di-*append* ke suatu *array* baru hingga semua titik lainnya telah diuji dengan persamaan garis tersebut. Akhirnya kedua titik tersebut dikatakan titik-titik pada *convex hull* jika dan hanya jika semua *array boolean* tersebut memiliki nilai *boolean* yang sama.

Dari paragraf pertama dilakukannya iterasi titik secara kombinatorial, kompleksitasnya *big-O*-nya telah berupa $O(n^2)$ karena untuk setiap titik dilakukan pemasangan dengan titik-titik lainnya. Kemudian dari percobaan masing-masing titik dengan persamaan garis tersebut meningkatkan kompleksitasnya sebesar $O(n)$, sehingga kompleksitas akhir programnya adalah $O(n^3)$.

BAB II

IMPLEMENTASI KODE PROGRAM DALAM BAHASA C

2.1. HEADER *POINT.H*

Di bawah ditampilkan keempat fungsi dan prosedur yang ada pada header *point.h*

```
POINT MakePOINT (float X, float Y);
void BacaPOINT (POINT * P, int max);
void TulisPOINT (POINT P);
float Gradien (POINT P1, POINT P2);
```

Gambar 3. Fungsi dan prosedur pada header *point.h*

Fungsi **MakePOINT** mengembalikan sebuah **POINT** dari input parameter absis (*X*) dan ordinat (*Y*). Prosedur **TulisPOINT** menuliskan sebuah **POINT** dalam bentuk "(<**absis**>, <**ordinat**>)". Kedua fungsi tidak ditampilkan lagi *screenshot* metodenya mengingat abstraksinya terlalu rendah.

Di bawah ditampilkan prosedur **BacaPOINT** yang membentuk suatu *point* dengan acak dan batasan nilai ***max*** pada kedua absis dan ordinatnya. Fungsi **Gradien** yang menerima *input* dua titik juga ditampilkan dibawah. Mengingat gradient membagi perbedaan ordinat dengan perbedaan absis, maka jika kedua titik memiliki absis yang sama, maka secara matematika akan menghasilkan nilai tak terhingga. Untuk mengkompensasi hal tersebut, dilakukan pengecekan kesamaan nilai absis terlebih dahulu dan dikembalikan suatu nilai relatif besar (9999) untuk mengatasi *error* tersebut.

```
void BacaPOINT (POINT * P, int max){
    float x, y;
    x = rand( ) % max;
    y = rand( ) % max;
    *P = MakePOINT(x, y);
}
```

Gambar 4.Implementasi prosedur **BacaPOINT**

```
float Gradien (POINT P1, POINT P2){
    if(Absis(P1) == Absis(P2)){
        return 9999;
    }
    return (Ordinat(P2) - Ordinat(P1)) /(Absis(P2) - Absis(P1));
}
```

Gambar 5. Implementasi fungsi **Gradien**

2.2. HEADER *ARRAY.H*

Sesuai metode yang ditulis diatas, program membutuhkan beberapa metode dasar untuk operasi *array* sebagai berikut.

```
void MakeEmpty(TabPOINT *T, int maxel);
void BacaIsiTabPOINT(TabPOINT *T);
void TulisIsiTabPOINT(TabPOINT T);
boolean Determinasi(TabPOINT T, int a, int b);
```

Gambar 6. Fungsi dan prosedur pada header *array.h*

MakeEmpty, **BacaIsiTabPOINT** dan **TulisIsiTabPOINT** merupakan prosedur *generic* yang hanya melakukan metode seperti bagaimana semestinya (tidak ditampilkan lagi karena merupakan bentuk abstraksi biasa dari program pada mata kuliah Algoritma dan Struktur Data).

MakeEmpty menerima *pointer array* dan nilai ***maxel*** sebagai parameternya dan menginisiasi *array baru TABPOINT* dengan jumlah ***maxel***. **BacaIsiTabPOINT** awalnya membaca *input user* dengan suatu nilai ***N***, dan kemudian mengisi *array* yang disediakan pada parameternya dengan memanggil *procedure* **BacaPOINT** dari header *point.h* sebanyak ***N*** iterasi. Di sisi lain, **TulisIsiTabPOINT** memanggil *procedure* **TulisPOINT** untuk iterasi setiap elemennya, dipisahkan dengan sebuah koma (,), dan dimulai dan diakhiri dengan [].

Fungsi **Determinasi** menerima *array of POINT* (**T**), indeks elemen ke-***a*** pada *array* **T** (titik pertama) dan indeks elemen ke-***b*** pada *array* **T** (titik kedua) sebagai parameternya. Kemudian, untuk setiap titik selain elemen *array berindeks-**a*** dan elemen *array berindeks-**b***, dilakukan perbandingan sesuai yang telah dijelaskan di **Bagian 2.2**

```
boolean Determinasi(TabPOINT T, int a, int b){
  boolean B, state;
  float Y1 = Ordinat(Elmt(T, a)) +  Gradien(Elmt(T, b), Elmt(T, a))
             * ( Absis(Elmt(T, 1)) - Absis(Elmt(T, a)) );
  state = (Y1 >= Ordinat(Elmt(T, 1)));
  B = state;
  int i = 2;
  while(i <= Neff(T) && B == state){
    if(i != a && i != b){
      float Y = Ordinat(Elmt(T, a)) +  Gradien(Elmt(T, b), Elmt(T, a))
                * ( Absis(Elmt(T, i)) - Absis(Elmt(T, a)) );
      B = (Y >= Ordinat(Elmt(T, i)));
    }
    i ++;
  }
  return B == state;
}
```

Gambar 7. Implementasi fungsi **Determinasi**

2.3. *MAIN PROGRAM*

Program utama berjalan sebagai berikut. Dimulai dengan membaca *array of POINT* (**BacaIsiTabPOINT**), menginisiasi sebuah array **THull** dan mengiterasikan semua kombinasi pasangan *POINT* dengan semua titik selain pasangan tersebut, dan men-**DETERMINASI**-kan benar tidaknya pasangan *POINT* tersebut merupakan pasangan titik (garis) *convex hull*. Sampai sini, *array* **THull** akan berisi semua *POINT* sebanyak dua kali. Oleh karena itu, dibuatlah fungsi **Singelisasi** yang mengembalikan *array* **THull** dengan kondisi semua elemen tidak ada yang ganda, sebelum akhirnya menampilkan **THull** final. Berikut ditampilkan program utama (**main**) dan fungsi **Singelisasi**. Program juga akhirnya akan menampilkan waktu pemrosesan algoritma *convex hull*. Akan tetapi, untuk skala kecil (1 <= ***N*** <= 100), waktu pemrosesan tidak biasanya diatas 5 milidetik.

```
int main(){
    TabPOINT T, THull;
    BacaIsiTabPOINT(&T);
    float start = clock();
    MakeEmpty(&THull, Neff(T) * 2 + 3);
    int iTHull = IdxMin;
    for(int i = IdxMin; i <= Neff(T); i ++){
        for(int j = i; j <= Neff(T); j ++){
            if(i != j){
                if(Determinasi(T, i, j)){
                    Neff(THull) += 2;
                    Absis(Elmt(THull, iTHull)) = Absis(Elmt(T, i));
                    Ordinat(Elmt(THull, iTHull)) = Ordinat(Elmt(T, i));
                    Absis(Elmt(THull, iTHull + 1)) = Absis(Elmt(T, j));
                    Ordinat(Elmt(THull, iTHull + 1)) = Ordinat(Elmt(T, j));
                    iTHull += 2;
                }
            }
        }
    }
    float end = clock();
    Singelisasi(&THull);
    printf("\nOnly-points result output\n");
    TulisIsiTabPOINT(THull);
    printf("\n==========================================================================");
    printf("Program were running for %.3f miliseconds", ((float)end - start));
    printf("specifically for the convex-hull algorithm,");
    printf(" and regardless of the console logging system.\n");
    printf("==========================================================================");
    return 0;
}
```

Gambar 8. Program utama

```
void Singelisasi(TabPOINT *T){
  for(int i = 1; i <= Neff(*T); i ++){
    float X = Absis(Elmt(*T, i));
    float Y = Ordinat(Elmt(*T, i));
    for(int j = i + 1; j <= Neff(*T); j ++){
      if(X == Absis(Elmt(*T, j)) && Y == Ordinat(Elmt(*T, j))){
        Absis(Elmt(*T, j)) = Absis(Elmt(*T, Neff(*T)));
        Ordinat(Elmt(*T, j)) = Ordinat(Elmt(*T, Neff(*T)));
        Neff(*T) --;
      }
    }
  }
}
```

Gambar 9. Fungsi **Singelisasi**

BAB III

SCREENSHOT HASIL PROGRAM

3.1. INPUT BERJUMLAH 5

```
Program compiled and run
=====================
Input number of points
=====================
5
================================================================================================================================
Kondisi Awal
[(85.00, 38.00),(15.00, 67.00),(83.00, 45.00),(14.00, 52.00),(40.00, 20.00)]
================================================================================================================================

Only-points result output
[(85.00, 38.00),(83.00, 45.00),(15.00, 67.00),(40.00, 20.00),(14.00, 52.00)]
================================================================================================================================
Program were running for 0.000 miliseconds specifically for the convex-hull algorithm, and regardless of the console logging system.
================================================================================================================================
```

```
Program compiled and run
=====================
Input number of points
=====================
5
================================================================================================================================
Kondisi Awal
[(35.00, 37.00),(38.00, 69.00),(31.00, 89.00),(99.00, 13.00),(8.00, 37.00)]
================================================================================================================================

Only-points result output
[(31.00, 89.00),(99.00, 13.00),(8.00, 37.00)]
================================================================================================================================
Program were running for 0.000 miliseconds specifically for the convex-hull algorithm, and regardless of the console logging system.
================================================================================================================================
```

3.2. INPUT BERJUMLAH 10

```
Program compiled and run
====================
Input number of points
====================
10
================================================================================================================================
Kondisi Awal
[(90.00, 13.00),(87.00, 32.00),(36.00, 94.00),(94.00, 62.00),(61.00, 40.00),(32.00, 35.00),(1.00, 97.00),(21.00, 19.00),(57.00, 7.00),(8.00, 83.00)]
================================================================================================================================

Only-points result output
[(36.00, 94.00),(94.00, 62.00),(57.00, 7.00),(1.00, 97.00),(21.00, 19.00)]
================================================================================================================================
Program were running for 0.000 miliseconds specifically for the convex-hull algorithm, and regardless of the console logging system.
================================================================================================================================
```

```
Program compiled and run
====================
Input number of points
====================
10
================================================================================================================================
Kondisi Awal
[(19.00, 58.00),(28.00, 95.00),(32.00, 78.00),(71.00, 19.00),(63.00, 77.00),(61.00, 84.00),(3.00, 8.00),(5.00, 62.00),(98.00, 78.00),(2.00, 11.00)]
================================================================================================================================

Only-points result output
[(28.00, 95.00),(5.00, 62.00),(2.00, 11.00),(98.00, 78.00),(71.00, 19.00),(3.00, 8.00)]
================================================================================================================================
Program were running for 0.000 miliseconds specifically for the convex-hull algorithm, and regardless of the console logging system.
================================================================================================================================
```

3.3. INPUT BERJUMLAH 20

```
====================
Input number of points
====================
20
==============
[(37.00, 36.00),(89.00, 78.00),(43.00, 45.00),(2.00, 55.00),(32.00, 74.00),(67.00, 61.00),(7.00, 37.00),(83.00
, 34.00),(14.00, 22.00),(15.00, 7.00),(65.00, 9.00),(77.00, 62.00),(9.00, 56.00),(96.00, 66.00),(43.00, 6.00),
(88.00, 1.00),(46.00, 35.00),(92.00, 35.00),(83.00, 69.00),(18.00, 86.00)]
==============
Only-points result output
[(89.00, 78.00),(96.00, 66.00),(88.00, 1.00),(18.00, 86.00),(2.00, 55.00),(15.00, 7.00)]
==============
Program were running for 0.000 miliseconds specifically for the convex-hull algorithm, and regardless of the c
onsole logging system.
```

```
Input number of points
====================
20
==============
[(73.00, 71.00),(61.00, 33.00),(97.00, 69.00),(56.00, 2.00),(89.00, 99.00),(79.00, 84.00),(35.00, 91.00),(82.0
0, 65.00),(85.00, 3.00),(40.00, 97.00),(63.00, 79.00),(40.00, 76.00),(33.00, 88.00),(22.00, 34.00),(89.00, 21.
00),(23.00, 19.00),(76.00, 96.00),(73.00, 8.00),(45.00, 95.00),(12.00, 59.00)]
==============
Only-points result output
[(97.00, 69.00),(89.00, 99.00),(12.00, 59.00),(89.00, 21.00),(56.00, 2.00),(85.00, 3.00),(35.00, 91.00),(23.00
, 19.00),(40.00, 97.00)]
==============
Program were running for 0.000 miliseconds specifically for the convex-hull algorithm, and regardless of the c
onsole logging system.
```

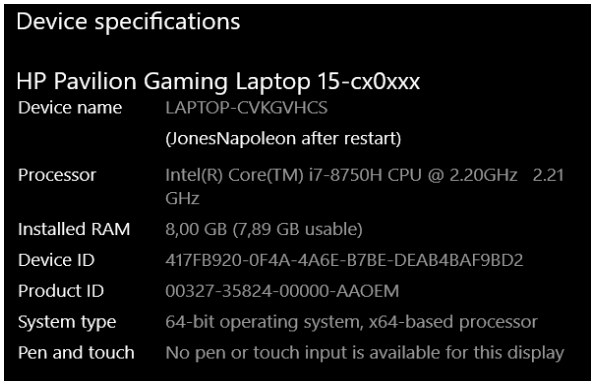
BAB IV

LAIN-LAIN

4.1. PENILAIAN INDIVIDU

No	Poin	Ya	Tidak
1.	Program berhasil dikompilasi	√	
2.	Program berhasil running	√	
3.	Program dapat menerima input dan menuliskan output.	√	
4.	Luaran sudah benar untuk semua n	√	

4.2. SPESIFIKASI PERANGKAT KERAS YANG DIGUNAKAN



Gambar 10. Spesifikasi Laptop