**LAPORAN TUGAS KECIL 1**

**PENYELESAIAN PERSOALAN CONVEX HULL DENGAN ALGORITMA BRUTE FORCE**

**IF2211 Strategi Algoritma**

Oleh:

**Daffa Pratama Putra 13518033**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2019**

**BAB I**

**ALGORITMA BRUTE FORCE**

Algoritma *brute force* adalah salah satu algoritma yang digunakan untuk menyelesaikan persoalan komputasi. Pendekatan yang dilakukan oleh algoritma *brute force* adalah secara *straight forward* atau secara langsung. Biasanya algoritma ini digunakan untuk menyelesaikan persoalan yang sederhana, langsung, dan jelas persoalannya. Contoh penggunaan algoritma *brute force* adalah untuk mencari elemen terbesar/terkecil pada senarai, mencari elemen yang ada pada senarai, menghitung faktorial, dan lain sebagainya,

Persoalan *Convex Hull* dapat diselesaikan oleh algoritma *brute force*. *Brute force* yang digunakan adalah mencoba semua kombinasi titik pada himpunan titik yang diberikan, hingga didapat himpunan titik pembentuk *Convex Hull*. Metode yang digunakan adalah dengan penyelesaian secara matematis menggunakan pertidaksamaan garis lurus. Garis lurus dapat dibuat dari 2 titik. Langkah – langkah untuk menyelesaikan persoalan tersebut adalah (asumsikan titik – titik telah tersedia) :

1. Pilih salah sau titik dari himpunan titik yang telah diberikan. Misalkan himpuan tersebut adalah himpunan S.
2. Pilih titik lain dari himpunan S sebagai kemungkinan titik pembentuk *Convex Hull* dan cari persamaan garisnya dari kedua titik tersebut.
3. Lakukan pengujian kepada titik sisa dari himpunan S untuk menguji keberadaan titik tersebut terhadap garis yang telah dibentuk.
4. Apabila semua titik uji berada pada di salah satu sisi terhadap garis, maka kedua titik tersebut merupakan anggota himpunan titik pembentuk *Convex Hull*. Jika sebaliknya, maka ulangi langkah ke-2.
5. Ulangi langkah 1-4 hingga semua kemungkinan titik pembentuk *Convex Hull* didapat.

Pertidaksamaan garis lurus dari 2 titik dapat dihitung dengan menggunakan koordinat kedua titik tersebut, yakni :

Bentuk umum pertidaksamaan garis

, dengan

,

, dan

.

**BAB III**

**KODE PROGRAM**

Program yang dibuat untuk mencari *Convex Hull* menggunakan bahasa C dengan memanfaatkan tipe data bentukan. Tipe bentukan yang digunakan adalah POINT dan SET dengan spesifikasi sebagai berikut :

|  |
| --- |
| typedef struct {      int X; /\* Absis   \*/      int Y; /\* Ordinat \*/;  } POINT;  typedef struct {      POINT \*Tab; /\* Container menyimpan elemen \*/      int MaxEl;  /\* Jumlah maksimum elemen \*/  } SET;  #define Absis(P) (P).X  #define Ordinat(P) (P).Y  #define Elmt(S, i) (S).Tab[(i)]  #define Tab(S) (S).Tab  #define MaxEl(S) (S).MaxEl |

Dalam program ini juga dibutuhkan fungsi-fungsi yang digunakan untuk mencari *Convex Hull*. Berikut adalah *header* dari fungsi-fungsi tersebut :

|  |
| --- |
| POINT MakePOINT (int X, int Y);  /\* Membentuk POINT dari komponen X dan Y \*/  void TulisPOINT (POINT P);  /\* Menuliskan POINT dalam format (X,Y) \*/  int Jarak (POINT P1, POINT P2);  /\* Menghitung jarak antara 2 titik \*/  void MakeEmpty(SET \*S, int maxel);  /\* Membuat array dinamis POINT dengan elemen sejumlah N+1 \*/  bool EQ (POINT P1, POINT P2);  /\* Mengecek apakah P1 dan P2 adalah titik yang sama \*/  int isLeft(POINT Pa, POINT Pb, POINT Px);  /\* Mengecek apakah titik Px berada di sebelah "kiri" dari garis L \*/  int isOneSide(int N, int \*Side);  /\* Mengecek apakah semua titik yang dicek \*/  /\* berada di salah satu sisi atau tidak \*/  bool Search(SET S, POINT P);  /\* Mencari apakah P merupakan elemen dari list \*/ |

Fungsi yang menjadi *core* dalam menentukan *Convex Hull* adalah fungsi isLeft dan isOneSide. Fungsi isLeft digunakan untuk mengecek apakah titik uji berada di sebelah “kanan” atau “kiri” garis Jika berada di sebelah kiri, akan diberi angka 1 pada suatu *array* dan sebaliknya, jika berada di sebelah kanan, akan diberi angka 0. Proses pengecekan apakah di sebelah kiri atau kanan, menggunakan pertidaksamaan garis lurus. Dengan memasukkan titik uji (x,y) ke dalam pertidaksamaan garis lurus (Ax + By < C), akan bisa ditentukan keberadaan titik uji tersebut. Lalu setelah tiap titik uji telah diujikan, fungsi isOneSide akan mengecek apakah dalam *array* yang menyimpan keberadaan titik uji, memiliki semua elemen yang bernilai sama. Jika semua elemen bernilai 0 atau 1, akan dikembalikan angka 1, begitupun sebaliknya. Implementasi dari fungsi-fungsi di atas, dapat dilihat pada *source code* program.

Algoritma *Brute Force* diimplementasikan pada program utama. Algoritma tersebut adalah sebagai berikut.

|  |
| --- |
| /\* Looping titik acuan \*/  ctr = 1;  for(i=1; i<=N; i++){      /\* Looping titik kemungkinan convex \*/      for(j=1; j<=N; j++){          if (i!=j){              /\* Checking titik \*/              count = 1;              for(k=1; k<=N; k++){                  if ((k!=i) && (k!=j)){                      Left[count] = isLeft(Elmt(S, i), Elmt(S, j), Elmt(S, k));                      count++;                  }              }              /\* Apakah satu sisi atau tidak \*/              if (isOneSide(N, Left) == 1){                  if (!Search(Convex, Elmt(S, i))){                      Elmt(Convex, ctr) = Elmt(S, i);                      ctr++;                  }              }          }      }  } |

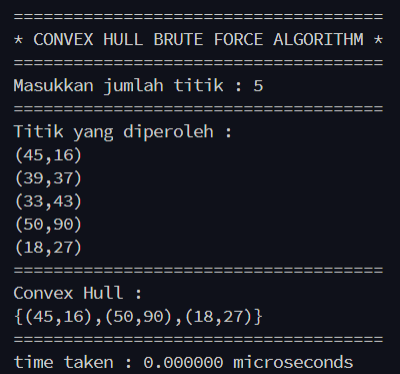
Algoritma *Brute Force* ini menggunakan tiga kali perulangan. Perulangan yang pertama adalah mengambil titik acuan dari titik-titik yang dibangkitkan. Perulangan kedua adalah mengambil titik lain, sehingga dapat membentuk garis dengan titik acuan. Perulangan ketiga adalah mengecek semua titik uji dengan garis yang sudah didapat dari titik acuan dan titik lain. Sehingga kompleksitas algoritma pada persoalan *Convex Hull* menggunakan algoritma *brute force* adalah O(n3).

**BAB IV**

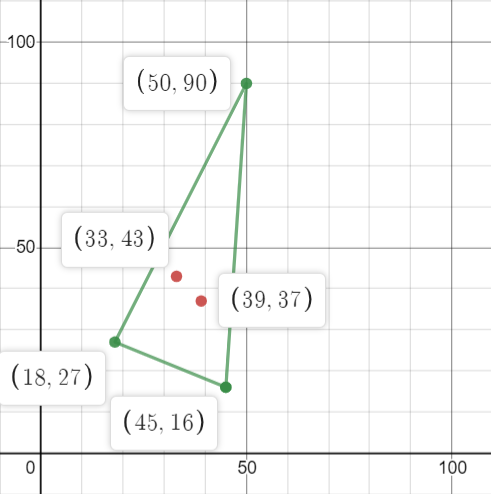
**UJI COBA**

1. **Uji Coba dengan N = 5**

Berikut adalah hasil keluaran program jika diberi masukan jumlah titik sebanyak 5.

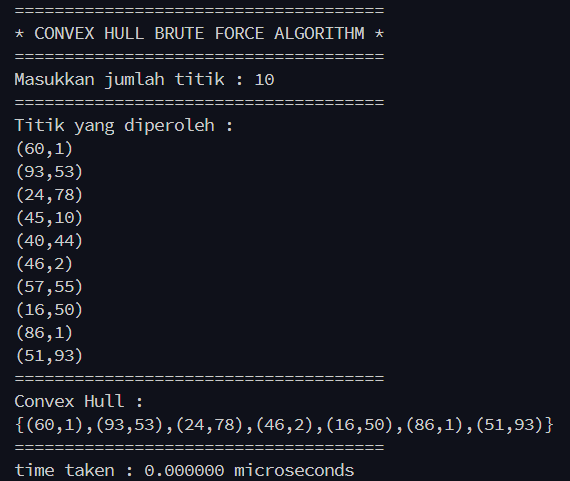


Untuk menguji kebenaran program, berikut adalah visualisasi keluaran tersebut.

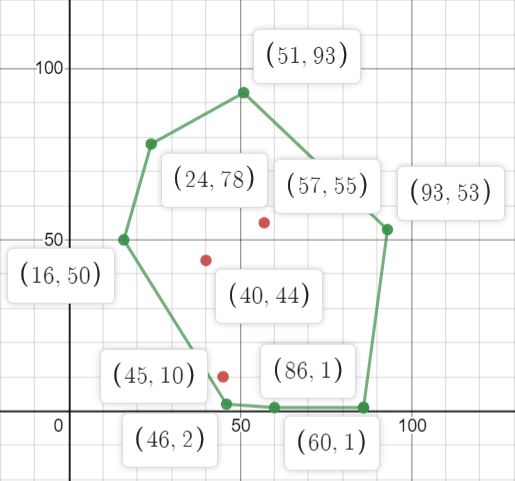


1. **Uji Coba dengan N = 10**

Berikut adalah hasil keluaran program jika diberi masukan jumlah titik sebanyak 10.

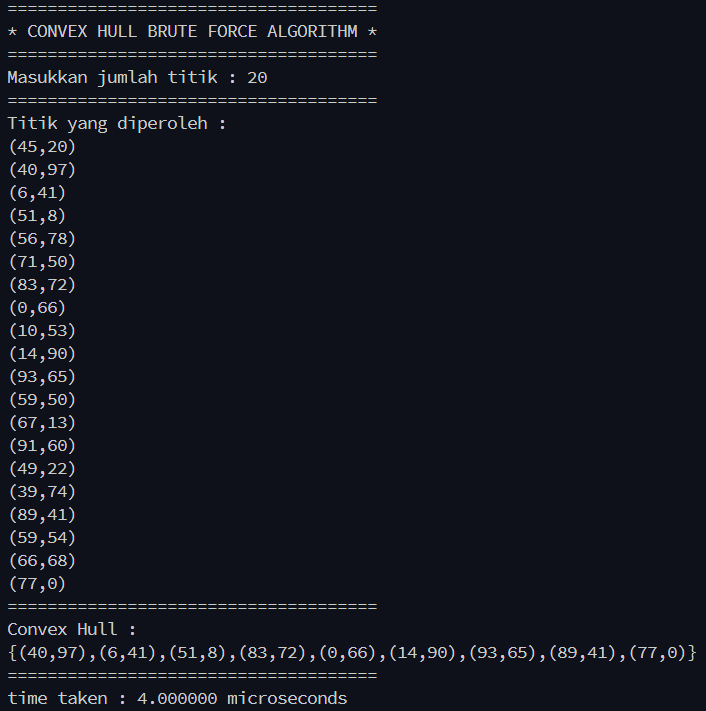


Untuk menguji kebenaran program, berikut adalah visualisasi keluaran tersebut.

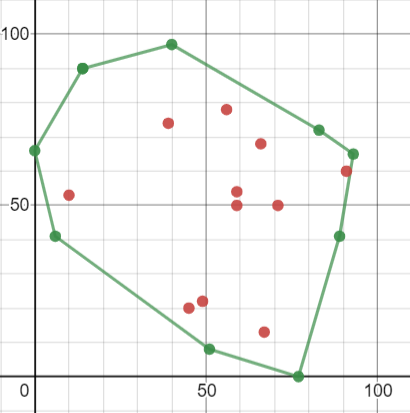


1. **Uji Coba dengan N = 20**

Berikut adalah hasil keluaran program jika diberi masukan jumlah titik sebanyak 20.



Untuk menguji kebenaran program, berikut adalah visualisasi keluaran tersebut.



Penilaian Asisten

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi | V |  |
| 1. Program berhasil *running* | V |  |
| 1. Program dapat menerima input dan menuliskan output | V |  |
| 1. Luaran sudah benar untuk semua *n* | V |  |