**LAPORAN TUGAS KECIL 1**

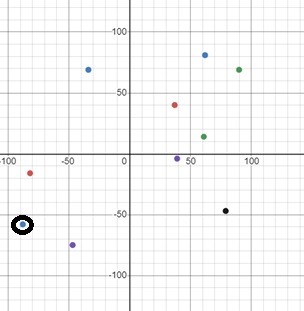
**STRATEGI ALGORITMA 2020**

Oleh: Naufal Arfananda Ghifari/13518096 (K-3)

1. **Konsep Algoritma yang digunakan**
2. Strategi Algoritma

Strategi algoritma *Brute Force* yang saya lakukan untuk menyelesaikan permasalahan *Convex Hull* adalah sebagai berikut:

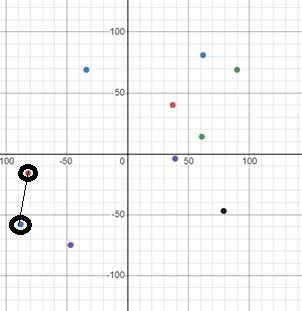
1. Mencari titik paling kiri P1;



Gambar 1

Titik palling kiri

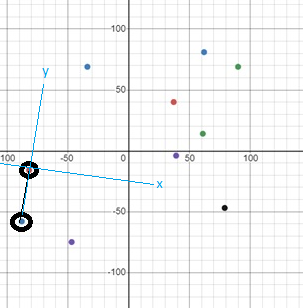
1. Hubungkan P1 pada titik lain yang mempunyai gradien paling tinggi, P2;



Gambar 2

P1 dihubungkan pada P2

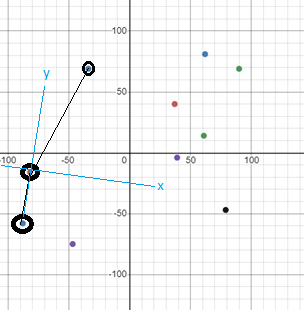
1. Ubah posisi sumbu x dan sumbu y sehingga sumbu x tegak lurus dengan garis antara P1 dan P2;



Gambar 3

Pembuatan sumbu x dan y semu

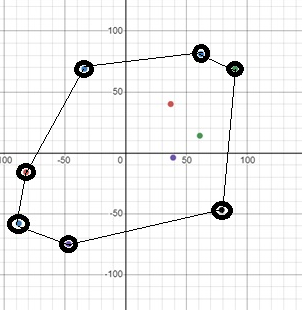
1. Hubungkan P2 pada titik lain yang mempunyai gradien paling tinggi sesuai sumbu baru, P3;



Gambar 4

P2 dihubungkan pada P3

1. Lakukan langkah 3 dan 4 berulang kali hingga kembali ke P1.



Gambar 5

Hasil Akhir Convex Hull

1. Konsep Matematis yang terlibat
2. Menghitung gradien
3. Pemutaran sumbu

Untuk pemutaran sumbu gradien, m1 dan m2 dimaksudkan dalam bentuk tangen. Sehingga dimungkinkan pengurangan m2 dengan m1 dengan memakai konsep yang telah dipelajari sebagai berikut.

1. Notasi Big-O

Misalkan N adalah jumlah titik yang dimasukkan dan count adalah jumlah titik yang merupakan convex hull, maka kompleksitas program dapat dinyatakan sebagai

P(N) = N x count

Sehingga kemungkinan terbaik adalah hanya ada satu titik yang merupakan *Convex Hull* sehingga

P(N) = N

Dan kemungkinan terburuk adalah semua titik merupakan C*onvex Hull* sehingga

P(N) = N x N

Jadi, kompleksitas rata-rata adalah

Sehingga Kompleksitas Algoritma *Brute Force* yang saya pakai adalah O(n2)

1. **Kode program**
2. Type Bentukan, konstanta, fungsi dan prosedur

|  |
| --- |
| struct point{  int x;  int y;  };  const float inf = std::numeric\_limits<float>::infinity();  const struct point foo = {-999, -998};  float grad(point P1, point P2);  //Menghasilkan gradien antara titik P1 dan P2  float mins(float m1, float m2);  //Menghasilkan gradien baru m1 yang sudah dirotasi sudutnya dengan m2  float ortho(float m);  //Menghasilkan gradien yang tegak lurus dengan garis m  bool cmpr(point P1, point P2);  //Membandingkan dua titik P1. P2. Menghasilkan true jika sama  void BacaPoint(point \*P);  //Membaca point P |

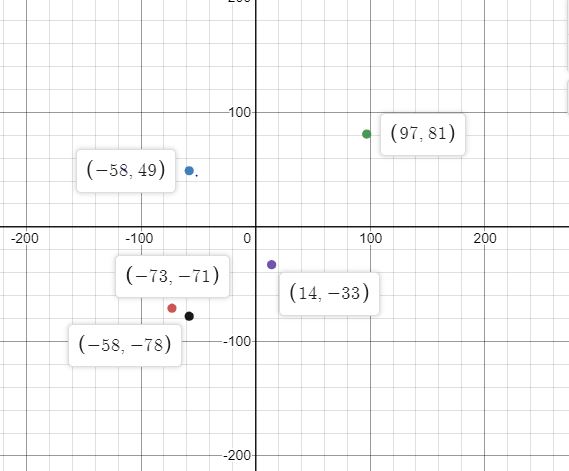
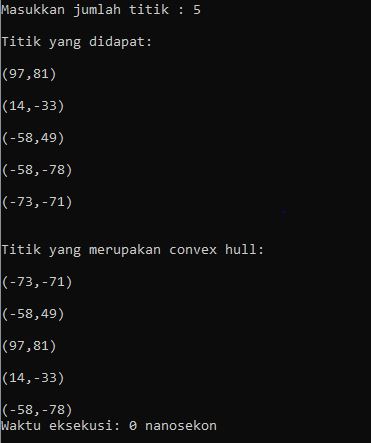
1. Masukan dan keluaran

|  |
| --- |
| /\*Prekondisi:  N adalah integer  F, P adalah point  i dan count adalah integer  A adalah array yang menampung semua titik yang ada  B adalah array yang menampung titik hasil  duration.count() merupakan waktu eksekusi algoritma brute force  \*/  //Membaca jumlahtitik/ N  do{  cout << "Masukkan jumlah titik : ";  cin >> N;  }while(N<=0);  //Mencetak semua titik yang dipilih acak  cout <<"\nTitik yang didapat: \n";  for (i = 0; i<N; i++){  cout << "\n(";  cout << A[i].x;  cout << ",";  cout << A[i].y;  cout << ")\n";  }  //Mencetak titik yang merupakan Convex Hull  cout <<"\n\nTitik yang merupakan convex hull:\n";  for (i = 0; i<count; i++){  cout << "\n(";  cout << B[i].x;  cout << ",";  cout << B[i].y;  cout << ")\n";  }  //Mencetak waktu eksekusi program  cout << "Waktu eksekusi: "<< duration.count() << " nanosekon" << endl; |

1. Algoritma *brute force*

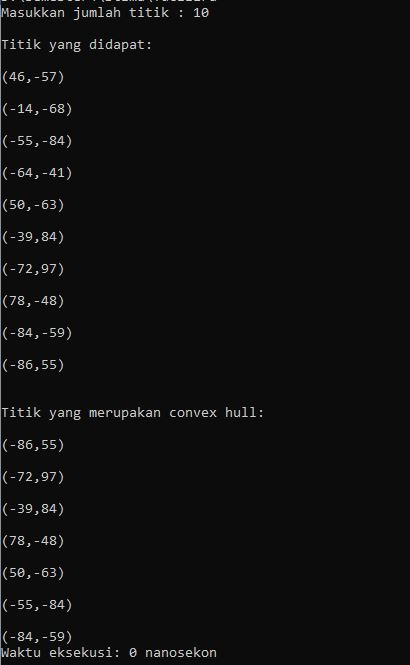
|  |
| --- |
| /\*Prekondisi:  N adalah jumlah titik  F, P adalah point  i dan count adalah integer  A adalah array yang menampung semua titik yang ada  B adalah array yang menampung titik hasil  \*/  BacaPoint(&F);  for (i=0; i<N-1; i++){  BacaPoint(&P);  if (P.x< F.x or (P.x == F.x and P.y < F.y)){  A[i] = F;  F =P;  }else{  A[i] = P;  }  }  A[N-1] = F;  B[0] = F;  count++;  //Memulai brute force tahap 3  //Cari titik kedua  float m,mi;  int index=0;  while (cmpr(A[index],F) && index <N-1 ){  A[index] = foo;  index++;  }  P = A[index];  i = index +1;  m= grad(F, P);  for (; i<N-1;i++){  if (!cmpr(F, A[i]) && !cmpr(A[i] , foo)){  mi = grad(F,A[i]);  if (m==inf){  if (mi==inf && A[i].y>P.y){  P= A[i];  index =i;  }  }else{  if (mi> m || mi == inf || (mi==m && m>0 && A[i].y> P.y)|| (mi==m && m<0 && A[i].y< P.y)||(mi==m && m==0 && A[i].x> P.x)){  P = A[i];  index =i;  m= mi;  }  }  }else if (cmpr(F,A[i]))  {  A[i] =foo;  }  }  point P0 = F;  if (!cmpr( P , F)){  B[1] = P;  A[index] = foo;  P0 = P;  count++;  }  //Cari titik-titik selanjutnya  float m4;  while(!cmpr(P0 ,F)){  m4 = ortho(m);  //cari P  index =0;  m=inf;  do {  if (cmpr(A[index], foo)){  index++;  }else if (cmpr(A[index], P0)){  A[index] = foo;  index++;  }  else{  m = mins(grad(P0, A[index]), m4);  index++;  }  }while((cmpr(A[index-1], foo) || cmpr(A[index-1], P0) || m== inf)&& index <N);  index--;  P = A[index];  i = index+1;  //Cari gradien tertinggi  for (; i< N;i++){  if (!cmpr(P0 ,A[i]) && !cmpr(A[i] , foo)){  mi = mins(grad(P0,A[i]),m4);  if (mi == inf){  if(i != N-1){  A[i] = foo;  }  }  else if(mi> m || mi==m) {  if (mi==m ){  if((A[index].y>P0.y &&A[i].y>A[index].y) || (A[index].y<P0.y &&A[i].y <A[index].y) || (A[index].y==P0.y && A[index].x> P0.x && A[i].x >A[index].x) || (A[index].y==P0.y && A[index].x< P0.x && A[i].x <A[index].x)){  P = A[i];  index = i;  m = mi;  }  }  else{  P = A[i];  index = i;  m = mi;  }  }  }else if(cmpr(P0, A[i])){  A[i] =foo;  }  }  if (!cmpr(P, F)){  B[count] = P;  count++;  m= grad(P,P0);  A[index] = foo;  }  P0 =P;  } |

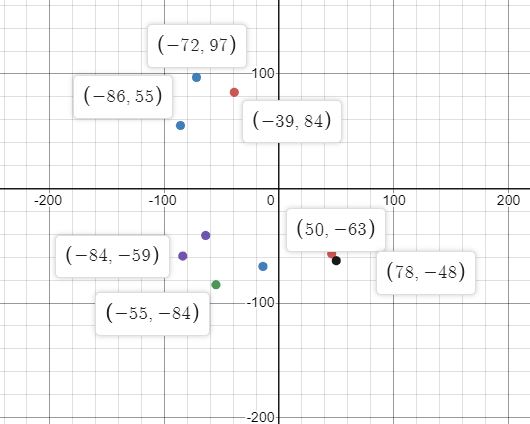
1. **Testing**
2. 5 titik



Gambar 6

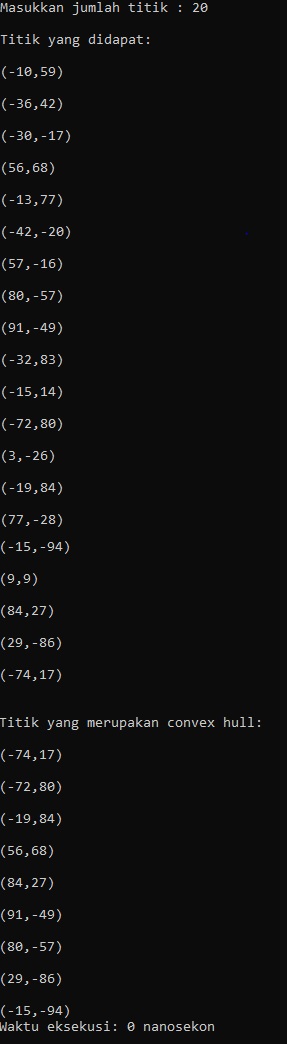
Kiri: gambaran 5 titik hasil tampilan desmos, Kanan: Hasil eksekusi program

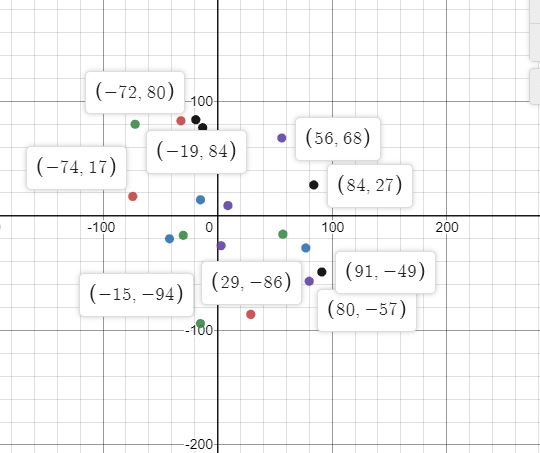
1. 10 titik



Gambar 7

Kiri: gambaran 10 titik hasil tampilan desmos, Kanan: Hasil eksekusi program

1. 20 titik



Gambar 8

Kiri: gambaran 10 titik hasil tampilan desmos, Kanan: Hasil eksekusi program

1. **List tambahan**

|  |  |  |
| --- | --- | --- |
| 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 |  |

**Daftar pustaka**

<https://www.desmos.com/>

<http://informatika.stei.itb.ac.id/~rinaldi.munir/>

Edwin, J. Purcell, Ridgon. 2003. Kalkulus. Erlangga : Jakarta