**LAPORAN**

**TUGAS KECIL 1 IF2211 STRATEGI ALGORITMA**

**Penyelesaian Persoalan Convex Hull Dengan Algoritma Brute Force**

****

Disusun oleh:

13518117 – Muhammad Firas

**TEKNIK INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**

**2020**

**BAB I**

**TEORI DASAR**

Salah satu hal penting dalam komputasi geometri adalah menentukan *convex hull* dari kumpulan titik. Himpunan titik pada bidang planar disebut *convex* jika untuk sembarang dua titik pada bidang tersebut (misal *p* dan *q*), seluruh segmen garis yang berakhir di *p* dan *q* berada pada himpunan tersebut. Contoh gambar 1 adalah poligon yang *convex*, sedangkan gambar 2 menunjukkan contoh yang *non-convex*.



Gambar 2: Non-convex

Gambar 1: Convex

*Convex Hull* dari himpunan titik S adalah himpunan *convex* terkecil yang mengandung S. Untuk dua titik, maka convex hull berupa garis yang menghubungkan 2 titik tersebut. Untuk tiga titik yang terletak pada satu garis, maka *convex hull* adalah sebuah garis yang menghubungkan dua titik terjauh. Sedangkan *convex hull* untuk tiga titik yang tidak terletak pada satu garis adalah sebuah segitiga yang menghubungkan ketiga titik tersebut. Untuk titik yang lebih banyak dan tidak terletak pada satu garis, maka convex hull berupa poligon *convex* dengan sisi berupa garis yang menghubungkan beberapa titik pada S. Contoh *convex hull* untuk delapan titik dapat dilihat pada gambar 3.



Gambar 3: Convex Hull untuk delapan titik

**BAB II**

**ANALISIS PERSOALAN**

Pada penyelesaian persoalan *convex hull* ini, digunakan algoritma *brute force*. Himpunan titik S akan dibangkitkan secara acak sebanyak *n* titik, lalu akan ditarik garis lurus yang menghubungkan dua titik *Pi* dan *Pj* dari kumpulan *n* titik.

Garis lurus melalui *Pi* = (*xi*, *yi*) dan *Pj* = (*xj*, *yj*) dapat didefinisikan melalui persamaan:

Dimana .

Garis lurus *PiPj* akan membentuk dua bidang di antara garis tersebut yang mana:

* , untuk semua titik berada di salah satu bidang,
* , untuk semua titik berada di bidang lainnya,
* , untuk titik yang menyinggung garis *PiPj*.

Segmen garis yang menghubungkan dua titik *Pi* dan *Pj* dari kumpulan *n* titik adalah bagian dari *convex hull-*nya jika dan hanya jika semua titik lain dari himpunan itu terletak di sisi yang sama dari garis lurus melalui dua titik ini. Jadi, untuk memeriksa apakah titik-titik tertentu terletak pada sisi yang sama dari garis, kita dapat memeriksa apakah persamaan memiliki tanda yang sama di setiap titik-titik ini.

Mengulangi percobaan ini untuk setiap pasangan titik akan menghasilkan himpunan segmen garis yang membentuk *convex hull*. Kompleksitas waktu dari algoritma ini adalah *O*(*n*3).

**BAB III**

**IMPLEMENTASI**

Untuk membuat program untuk menyelesaikan persoalan ini, penulis menggunakan bahasa Java dengan kode program sebagai berikut.

Class Point.java

|  |
| --- |
| public *class* Point {  *int* X; //absis  *int* Y; //ordinat  public Point (*int* x, *int* y)  {  *this.X* = x;  *this.Y* = y;  }  } |

Class Garis.java

|  |
| --- |
| public class Garis {  Point P1; //titik awal  Point P2; //titik akhir  public Garis(Point p1, Point p2)  {  this.P1 = p1;  this.P2 = p2;  }  } |

Program utama ConvexHull.java

|  |
| --- |
| import javax.swing.\*;  import java.awt.\*;  import java.util.ArrayList;  import java.util.Random;  import java.util.Scanner;  public class ConvexHull {  //Atribut  static int N;  static Point[] ArrP;  static ArrayList<Garis> ArrConvexHull;  static Scanner in = new Scanner(System.in);  public static void main(String[] args) {  BuatTitik();  CariConvexHull();  GUI();  } |

Prosedur membangkitkan titik secara acak

|  |
| --- |
| private static void BuatTitik() {  do {  System.out.print("Banyak titik yang akan dibuat : ");  N = in.nextInt();  if (N < 2){  System.out.println("Masukan harus N > 1");  System.out.println();  }  } while (N < 2);  ArrP = new Point[N];  Random r = new Random();  for (int i=0; i<N; i++){ //menentukan koordinat titik secara acak dari 0 sampai 300 sebanyak N  ArrP[i] = new Point(r.nextInt(301), r.nextInt(301));  }  ArrConvexHull = new ArrayList<>();  for (int i = 0; i < N; i++) {  System.out.printf("(%d,%d)\n", ArrP[i].X, ArrP[i].Y);  }  System.out.println();  } |

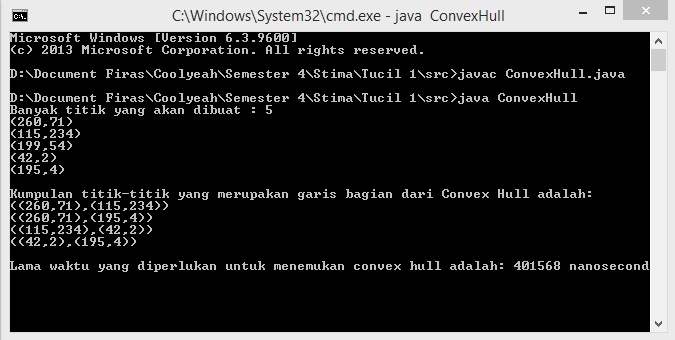
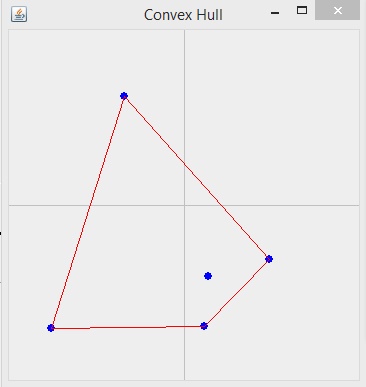
Prosedur pencarian titik *convex hull*

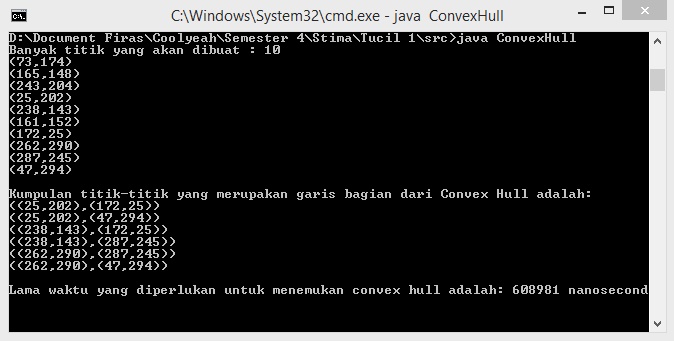
|  |
| --- |
| private static void CariConvexHull() {  int Persamaan1;  int Persamaan2;  int Count1;  int Count2;  Garis temp;  long StartTime = System.nanoTime(); //menghitung waktu pencarian convex hull dengan brute force  for (int i = 0; i < N; i++) {  for (int j = i + 1; j < N; j++) {  if ((ArrP[i].X != ArrP[j].X) && (ArrP[i].Y != ArrP[j].Y)) {  Persamaan2 = (ArrP[i].X \* ArrP[j].Y) - (ArrP[i].Y \* ArrP[j].X);  Count1 = 0;  Count2 = 0;  for (int k = 0; k < N; k++) {  Persamaan1 = (ArrP[j].Y - ArrP[i].Y) \* ArrP[k].X + (ArrP[i].X - ArrP[j].X) \* ArrP[k].Y;  if (Persamaan1 > Persamaan2){  Count1 = Count1 + 1;  }  if (Persamaan1 < Persamaan2){  Count2 = Count2 + 1;  }  }  if ((Count1 == 0) || (Count2 == 0)){  temp = new Garis(ArrP[i],ArrP[j]);  ArrConvexHull.add(temp);  }  }  }  }  long EndTime = System.nanoTime();  System.out.println("Kumpulan titik-titik yang merupakan garis bagian dari Convex Hull adalah:");  for (int i = 0; i < ArrConvexHull.size(); i++) {  System.out.printf("((%d,%d),(%d,%d))\n", ArrConvexHull.get(i).P1.X, ArrConvexHull.get(i).P1.Y, ArrConvexHull.get(i).P2.X, ArrConvexHull.get(i).P2.Y);  }  System.out.println();  System.out.println("Lama waktu yang diperlukan untuk menemukan convex hull adalah: " + (EndTime-StartTime) + " nanosecond");  System.out.println();  } |

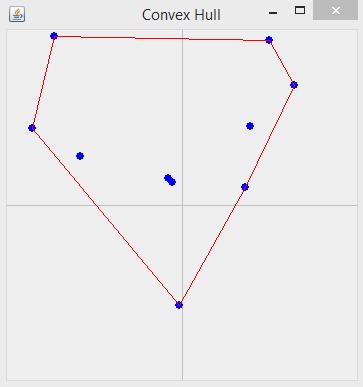
**BAB IV**

**UJI KASUS**

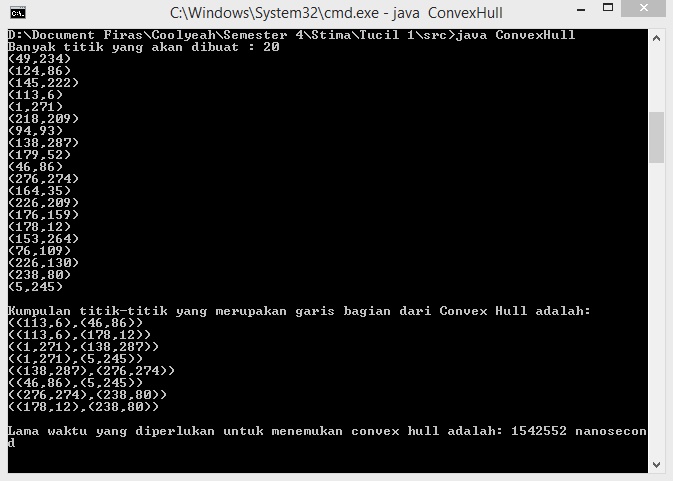
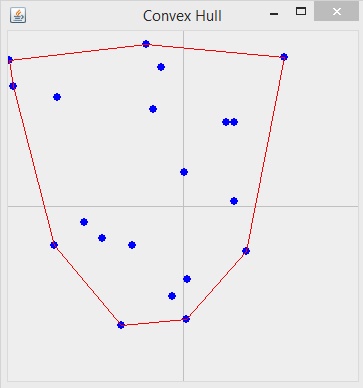
* *Screen-shot* input-output program untuk n = 5



* *Screen-shot* input-output program untuk n = 10



* *Screen-shot* input-output program untuk n = 20



Spesifikasi komputer/laptop yang digunakan untuk mengeksekusi program ini adalah sebagai berikut : Intel Core i7-5500U 2.4GHz processor, Nvidia Geforce 940M 2GB graphics card, 8GB DDR3 memory (RAM), 1 TB HDD, 64-bit Operating System, x64 based processor.

|  |  |  |
| --- | --- | --- |
| 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* | 🗸 |  |

**DAFTAR PUSTAKA**

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

<http://mslab.csie.asia.edu.tw/~ktduc/AA/Lecs/Chapter03%20Duc%20Anany%20V.%20Levitin%203e.pdf>