**LAPORAN**

**TUGAS KECIL 1 STRATEGI ALGORITMA (IF2211)**

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

****

Disusun oleh:

13518027 – Jundullah

**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 *Di* dan *Dj* dari kumpulan *n* titik.

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

Dimana .

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

* , untuk semua titik berada di salah satu sisi garis,
* , untuk semua titik berada di sisi lainnya,
* , untuk semua titik berada pada garis.

Segmen garis yang menghubungkan dua titik *Di* dan *Dj* dari himpunan S adalah bagian dari *convex hull-*nya jika dan hanya jika semua titik lain dari himpunan itu terletak di sisi yang sama dari perpanjangan segmen garis lurus melalui dua titik ini. Untuk memeriksa semua titik selain *Di* dan *Dj*  terletak pada sisi yang sama dari garis *DiDj*, kita dapat memeriksa apakah persamaan berada pada kategori yang sama (semua titik termasuk ke kategori pertama atau semua titik termasuk ke kategori kedua) dengan catatan bahwa titik-titik yang termasuk ke kategori ketiga tidak perlu diperhitungkan karena titik tersebut tidak berada pada sisi manapun.

Secara prosedural, algoritma *bruteforce* yang digunakan adalah sebagai berikut:

1. Mencari kombinasi dari semua garis (dua buah titik) dari himpunan S yang dapat dibentuk.
2. Untuk setiap garis lakukan pengecekan terhadap semua titik dari himpunan S.

* Jika semua titik yang diperiksa berada pada sisi yang sama, garis yang dicek merupakan bagian dari *Convex Hull.*
* Jika terdapat setidaknya satu buah titik yang terletak pada sisi yang berlawanan terhadap garis yang sedang dicek, garis tersebut bukan merupakan bagian dari *Convex Hull.*

1. Setelah semua garis berhasil dicek, *Convex Hull* dari semua titik adalah titik-titik yang terlibat di dalam himpunan garis yang didapat dari algoritma pada poin tiga.

Kompleksitas dari algoritma *bruteforce* tersebut, jika dihitung berdasarkan banyaknya *if* yang digunakan untuk melakukan pemeriksaan semua titik terhadap semua garis yang dicek adalah:

**BAB III**

**IMPLEMENTASI**

Untuk membuat program untuk menyelesaikan persoalan ini, penulis menggunakan bahasa Java dalam beberapa *file* java dengan kode program sebagai berikut:

dot.java

|  |
| --- |
| **public class** dot{  **private float x**;  **private float y**;   **public** dot(**float** absis, **float** ordinat){  **this**.**x** = absis;  **this**.**y** = ordinat;  }  **public float** getX(){  **return this**.**x**;  }  **public float** getY(){  **return this**.**y**;  } } |

line.java

|  |
| --- |
| **public class** line{  **private** dot **titikPertama**;  **private** dot **titikKedua**;   **public** line(dot one ,dot two){  **this**.**titikPertama** = one;  **this**.**titikKedua** = two;  }  **public float** getGradient(){  **float** selisihX = -**this**.**titikPertama**.getX() + **this**.**titikKedua**.getX();  **float** selisihY = -**this**.**titikPertama**.getY() + **this**.**titikKedua**.getY();  **return** selisihY/selisihX;  }  **public** dot getTitikPertama() {  **return titikPertama**;  }  **public** dot getTitikKedua() {  **return titikKedua**;  }  **public int** getX1(){  **return** (**int**) **this**.**titikPertama**.getX();  }  **public int** getX2(){  **return** (**int**) **this**.**titikKedua**.getX();  }  **public int** getY1(){  **return** (**int**) **this**.**titikPertama**.getY();  }  **public int** getY2(){  **return** (**int**) **this**.**titikKedua**.getY();  }  **public int** getA(){  **return** (**int**) (**this**.**titikKedua**.getY()-**this**.**titikPertama**.getY());  }  **public int** getB(){  **return** (**int**) (**this**.**titikPertama**.getX()-**this**.**titikKedua**.getX());  }  **public int** getC(){  **return** (**int**) (**this**.**titikKedua**.getY()\***this**.**titikPertama**.getX()-  **this**.**titikPertama**.getY()\***this**.**titikKedua**.getX());  }  **public void** switchdot(){  dot temp = **this**.**titikPertama**;  **this**.**titikPertama** = **this**.**titikKedua**;  **this**.**titikKedua** = temp;  } } |

Grafik.java

|  |
| --- |
| **import** java.awt.\*; **import** java.util.ArrayList;  **public class** Grafik **extends** Canvas {   dot[] dotList;  ArrayList<line> lineList;  **int** width = 400;  **int** height = 400;  **final int** dotSize = 4;   **public** Grafik(dot[] dots, ArrayList<line> lines) {  **this**.dotList = dots;  **this**.lineList = lines;  }  **public void** paint(Graphics g){  width = getWidth();  height = getHeight();  g.drawLine(0,height/2,width,height/2);  g.drawLine(width/2,0,width/2,height);  g.setColor(**new** Color(255, 170, 4));  **for** (dot p : dotList) {  g.fillOval((**int**) p.getX() - dotSize / 2 + width/2, height/2 - (**int**) (p.getY()) - dotSize / 2, dotSize, dotSize);  }  g.setColor(**new** Color(33, 110,255));  **for** (line line : lineList) {  dot p;  g.drawLine(width/2+line.getX1(),height/2 -line.getY1(),width/2+line.getX2(),height/2 -line.getY2());  p = line.getTitikPertama();  g.fillOval((**int**) p.getX() - dotSize / 2 + width/2-1, height/2 - (**int**) (p.getY()) - dotSize / 2-1, dotSize+2, dotSize+2);  p = line.getTitikKedua();  g.fillOval((**int**) p.getX() - dotSize / 2 + width/2-1, height/2 - (**int**) (p.getY()) - dotSize / 2-1, dotSize+2, dotSize+2);  }  } } |

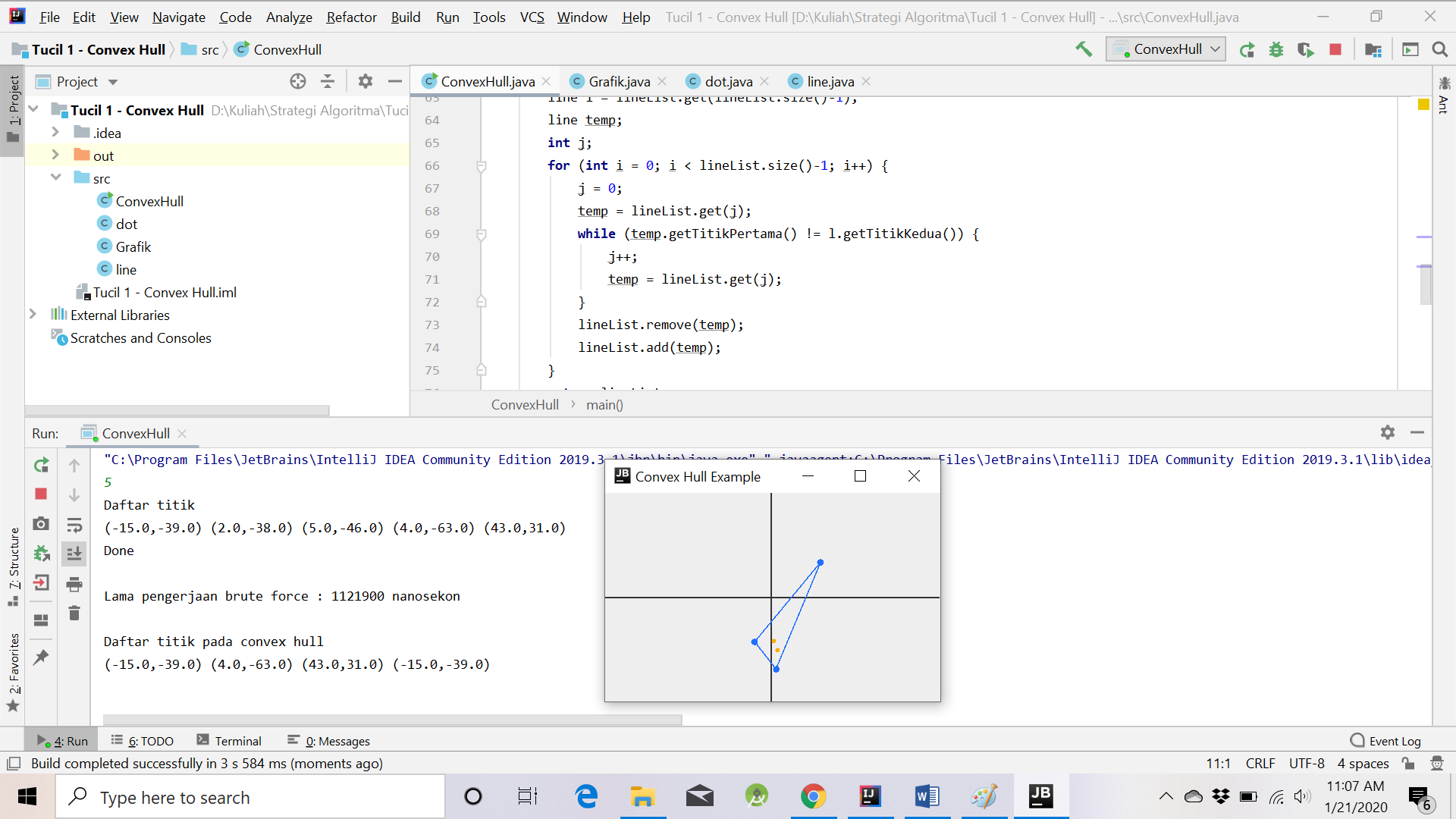
ConvexHull.java

|  |
| --- |
| **import** javax.swing.JFrame; **import** java.awt.\*; **import** java.util.ArrayList; **import** java.util.Random; **import** java.util.Scanner;  **public class** ConvexHull{  **public static void** main(String[] args){  Scanner scanner = **new** Scanner(System.***in***);  **int** dotCount = scanner.nextInt();  System.***out***.println(**"Daftar titik"**);  dot[] dotList = *createRandomDot*(dotCount);  System.***out***.println();  ArrayList<line> lineList = **new** ArrayList<>();  line temp;  **int** countLeft,countRight;   **if**(dotCount>1) {  **long** bruteForceStartTime = System.*nanoTime*();  **for** (**int** i = 0; i < dotCount; i++) {  **boolean** udah = **false**;  **for** (**int** j = i+1; j < dotCount; j++) {  **if**(dotList[i].getX()!=dotList[j].getX() || dotList[i].getY()!=dotList[j].getY()) {  countLeft = 0;  countRight = 0;  temp = **new** line(dotList[i], dotList[j]);  **for** (**int** k = 0; k < dotCount; k++) {  **if** (temp.getA() \* dotList[k].getX() + temp.getB() \* dotList[k].getY() > temp.getC()) {  countLeft++;  }  **if** (temp.getA() \* dotList[k].getX() + temp.getB() \* dotList[k].getY() < temp.getC()) {  countRight++;  }  }  **if** (countLeft == 0 || countRight == 0) {  **if** (udah) temp.switchdot();  lineList.add(temp);  udah = **true**;  }  }  }  }  **long** bruteForceEndTime = System.*nanoTime*();  *drawGUI*(dotList,lineList);  System.***out***.println();  System.***out***.println(**"Lama pengerjaan brute force : "**+(bruteForceEndTime-bruteForceStartTime) + **" nanosekon"**);  System.***out***.println();  System.***out***.println(**"Daftar titik pada convex hull"**);  **for**(line p : lineList){  *printDot*(p.getTitikPertama());  **if** (lineList.indexOf(p)%10==9){  System.***out***.println();  }  }  *printDot*(lineList.get(0).getTitikPertama());  System.***out***.println();  }  }   **private static** dot[] createRandomDot(**int** size) {  dot[] result = **new** dot[size];  Random random = **new** Random();  **for** (**int** i = 0; i<size; i++){  result[i] = **new** dot(random.nextInt(200)-100,random.nextInt(200)-100);  *printDot*(result[i]);  **if**(i%10==9){  System.***out***.println();  }  }  **return** result;  }   **private static void** drawGUI(dot[] dotList, ArrayList<line> lineList) {  JFrame jFrame = **new** JFrame(**"Convex Hull Example"**);  Canvas canvas = **new** Grafik(dotList,lineList);  canvas.setSize(400,400);  jFrame.add(canvas);  jFrame.pack();  jFrame.setVisible(**true**);   System.***out***.println(**"Done"**);  }   **private static void** printDot(dot print){  System.***out***.print(**"("**+print.getX() + **","** + print.getY()+**") "**);  }  } |

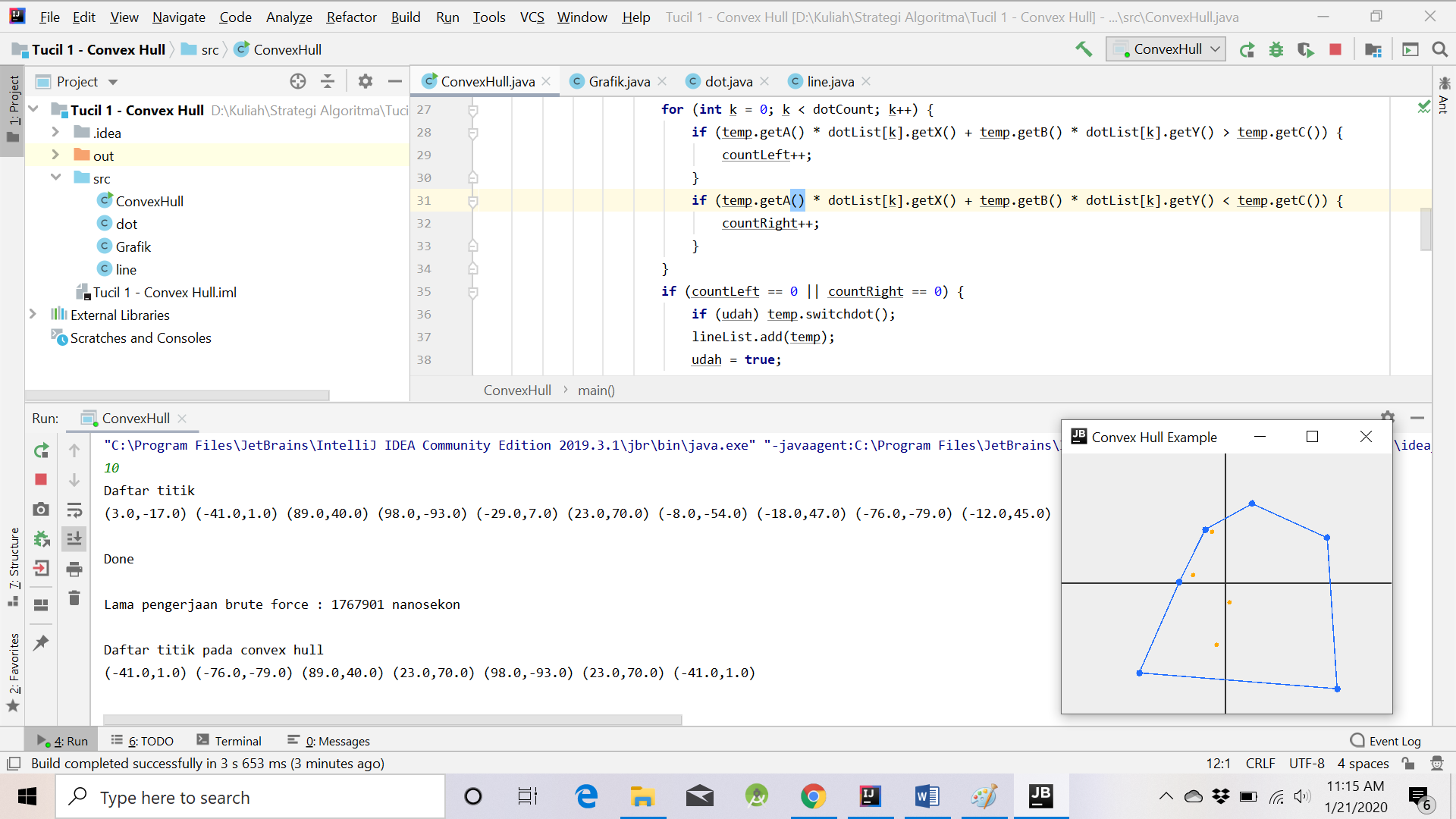
**BAB IV**

**UJI KASUS**

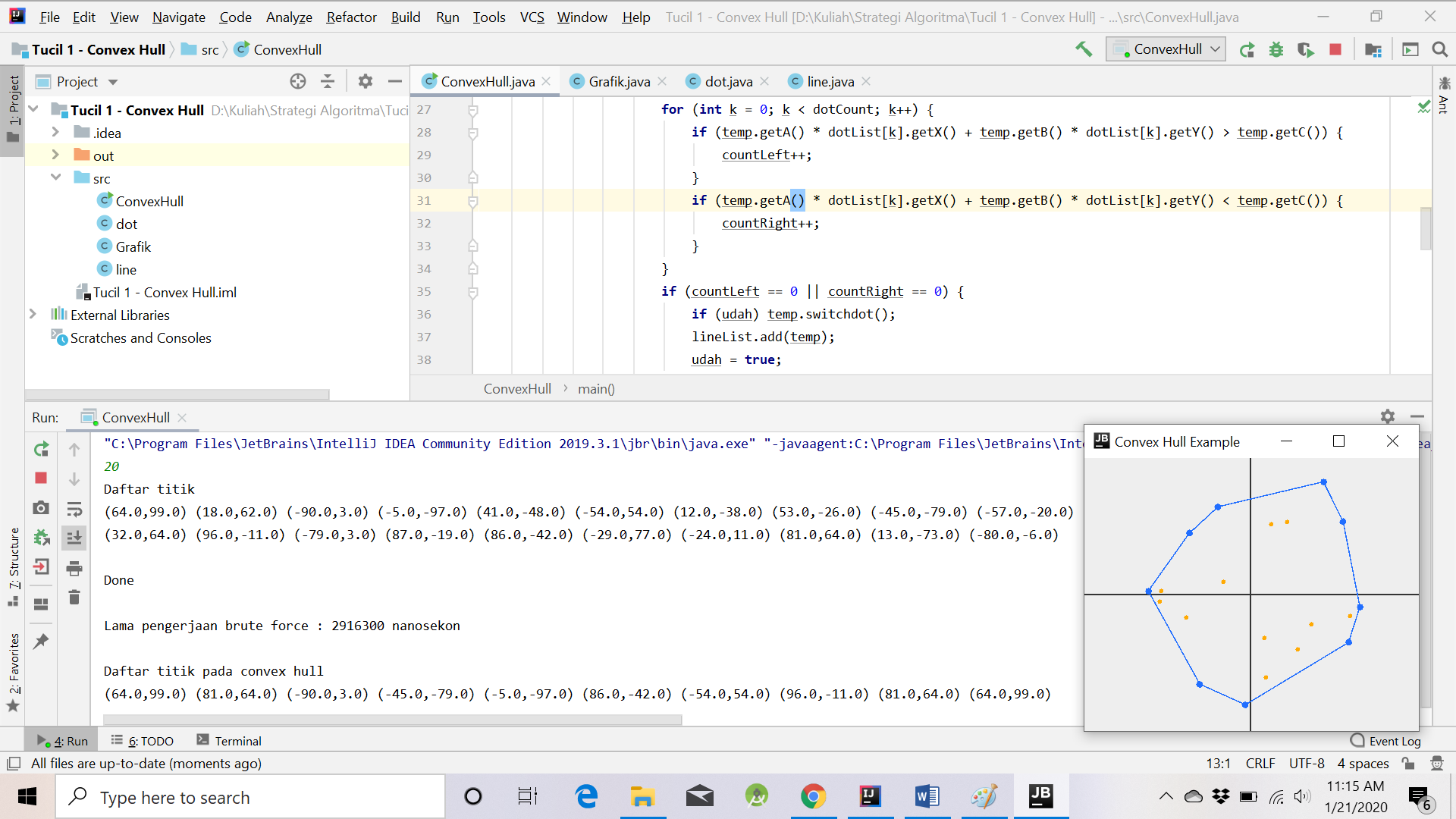
* *Screen-shot* input-output program untuk n = 5



* *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 : AMD Ryzen 7 3700U ~2.3GHz processor with Radeon Vega Mobile Gfx, 8GB DDR3 memory (RAM), 512 GB SSD, 64-bit Operating System.

Catatan : Tidak ada algoritma *sorting* pada program sehingga daftar titik pada *convex hull* yang ditampilkan tidak sesuai urutan *clockwise* maupun *counter-clockwise*.

**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>

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