**LAPORAN TUGAS KECIL 1**

IF2211 – STRATEGI ALGORITMA

**Penyelesaian Persoalan *Convex Hull***

**dengan Algoritma Brute Force**

****

Disusun oleh

Tony Eko Yuwono

13518030

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2020**

# **Algoritma**

* 1. **Algoritma Brute Force untuk mencari *Convex Hull***

1. Membangkitkan titik-titik secara acak sebanyak N, kemudian meletakkan titik-titik tersebut ke dalam sebuah larik.
2. Memilih sebuah titik secara urut sesuai indeks pada larik (misalkan titik tersebut bernama P1), mulai dari indeks pertama hingga indeks ke-N (for i = 1 to n).
3. Memilih sebuah titik yang lain untuk dihubungkan dengan titik pertama (misalkan titik tersebut bernama P2) secara terurut pada larik (for j = i to n).
4. Menghubungkan P1 dan P2 yang dipilih untuk setiap perulangan sehingga membentuk sebuah garis ax + by = c.
5. Melakukan pengecekan terhadap semua titik selain P1 dan P2 terhadap garis ax + by = c yang telah dibentuk kedua titik tersebut. Jika semua titik berada pada satu sisi yang sama (seluruh titik (x,y) selain P1 dan P2 berada pada sisi ax + by < c atau ax + by > c), maka P1 dan P2 merupakan titik-titik pembentuk convex hull.
6. Memasukkan titik P1 dan P2 ke dalam sebuah himpunan penyelesaian.
7. Mengulangi langkah ke 1-6 hingga semua titik telah teruji.
   1. **Kompleksitas Algoritma**

* Mengulangi penentuan titik pertama (for i = 1 to n) membutuhkan n langkah,
* Mengulangi penentuan titik kedua (for j = i to n) membutuhkan n-i langkah,
* Membentuk sebuah garis dan melakukan pengecekan letak setiap titik terhadap garis membutuhkan n-2 + 1 langkah untuk setiap perulangannya,
* Menambahkan titik-titik pembentuk Convex Hull ke dalam himpunan penyelesaian membutuhkan 1 langkah setiap perulangannya (dengan menggunakan prinsip Hash pada HashSet bahasa pemrograman Java),

Langkah secara keseluruhan adalah:

Sehingga kompleksitas algoritma secara keseluruhan adalah .

1. **Kode Program**

Program dibuat dengan bahasa Java dengan kode sebagai berikut:

1. **Main.java**

|  |
| --- |
| import java.util.\*;  import java.awt.\*;  class Main {      static Scanner input = new Scanner(System.in);      public static void main(String[] args) {          // Kamus          int N, x, y, i;          double startTime, endTime;          Vector<Point> P = new Vector<Point>();          Set<Point> result = new HashSet<Point>();          Set<Point[]> ConvexHull = new HashSet<Point[]>();          // Algoritma          System.out.print("Masukkan banyaknya titik yang diinginkan: ");  N = input.nextInt();          if (N > 1) {              System.out.println("\n"+N+" Titik Acak:");              for (i = 0; i < N; i++) {                  x = getRandomInteger(-630,630); // -630 <= x <= 630                  y = getRandomInteger(-340,320); // -340 <= y <= 320                  P.add(new Point(x,y));    System.out.println("Titik ke "+(i+1)+": "+"("+x+","+y+ ")");              }              /\* ======= CONVEX HULL ======= \*/              startTime = System.nanoTime();              ConvexHull = SearchConvexHull(P);              endTime = System.nanoTime();              /\* =========================== \*/              result.addAll(ConvertSetTupleToSet(ConvexHull));              System.out.print("\nHimpunan Titik Pembentuk Convex Hull: ");              System.out.print("{");              for (Point A : result) {                  System.out.print("(" + (int) A.getX() + "," +  (int) A.getY() + "),");              }              System.out.println("\b}");              System.out.println("\nWaktu untuk menemukan Convex Hull: "+  (endTime - startTime) / 1000000 +" ms");  // Plotting titik-titik dan menggambar convex hull              new Image(P, ConvexHull);          } else {              // N valid jika N > 1              System.out.println("Masukkan tidak sesuai!");          }      }      static Set<Point[]> SearchConvexHull(Vector<Point> P) {          /\* Kamus Lokal \*/          int i, j, k;          double a, b, c;          boolean stop, less;          Set<Point[]> result = new HashSet<Point[]>(); // set berisi array yang berisi 2 Point pembentuk convex hull          Point[] line;          /\* Algoritma \*/          if (P.size() > 2) {              for (i = 0; i < P.size(); i++) {                  for (j = i+1; j < P.size(); j++) {                      // Membuat persamaan garis: ax + by = c                      a = (P.get(j)).getY() - (P.get(i)).getY();  b = (P.get(i)).getX() - (P.get(j)).getX();                      c = (P.get(i)).getX() \* (P.get(j)).getY() –  (P.get(i)).getY() \* (P.get(j)).getX();                      k = 0;                      stop = false;                      // Inisiasi pengecekan                      while (((P.get(k).equals(P.get(i))) ||  (P.get(k).equals(P.get(j)))) && k < P.size()) { k++; }                      if (a \* P.get(k).getX() + b \* P.get(k).getY()< c)  { less = true; }                      else { less = false; }                      // Looping pengecekan                      while (k < P.size() && !stop) {                      // Stop saat ditemukan satu titik yang berada di sisiyang berbeda dengan titik yang lain pada garis ax+by=c                       if (!(P.get(k).equals(P.get(i))) &&  !(P.get(k).equals(P.get(j)))) {                               if ((less && (a \* P.get(k).getX() + b \* P.get(k).getY() > c)) || (!less && (a \* P.get(k).getX() + b \* P.get(k).getY() < c))) {                                      stop = true;                                  }                        }                        if (!stop) { k++; }                   }                          // Jika !stop, semua titik berada di kiri atau di kanan garis (P[i] dan P[j] titik2 pembentuk  convex hull)                      if (!stop) {                      // add ke Set sesuai dengan urutan p1.x < p2.x  line = new Point[]{P.get(i), P.get(j)};                          result.add(line);                      }                  }              }          } else {              // Kasus untuk 2 titik, 2 titik tersebut adalah pembentuk  convex hull              line = new Point[]{P.get(0), P.get(1)};              result.add(line);          }          return result;      }      static int getRandomInteger(int min, int max) {          return (int) (Math.random() \* ((max - min) + 1)) + min;      }      static Set<Point> ConvertSetTupleToSet(Set<Point[]> SetTupleOfLine) {          Set<Point> result = new HashSet<Point>();          for(Point[] A : SetTupleOfLine) { //Iterate Set              for (int it = 0; it < A.length; it++) { //Iterate Array                  result.add(A[it]);              }          }          return result;      }  } |

1. **Image.java**

|  |
| --- |
| import java.util.\*;  import java.awt.\*;  import javax.swing.\*;  import java.awt.Point;  public class Image extends JFrame {      Vector<Point> vectorOfPoints;      Set<Point[]> SetTupleOfLine;      int CenterX, CenterY;      ImageIcon img = new ImageIcon("../src/img/icon.png");      public Image(Vector<Point> vector, Set<Point[]> line) {          super("Convex Hull (Brute Force Algorithm) - 13518030");          getContentPane().setBackground(new Color(239,252,239));          setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);          setIconImage(img.getImage());          setSize(1280,720);          setVisible(true);          // Pusat Koordinat          this.CenterX = 1280/2;          this.CenterY = 720/2;          this.vectorOfPoints = vector;          this.SetTupleOfLine = line;      }      public void paint(Graphics g) {          super.paint(g);          Graphics2D g2 = (Graphics2D) g;          // Draw Sumbu X, Y          g.setColor(new Color(148,211,172));          g2.setStroke(new BasicStroke(2));          g2.drawLine(this.CenterX, 0, this.CenterX, 720);          g2.drawLine(0, this.CenterY, 1280, this.CenterY);          // Draw tulisan pusat koordinat dan koordinat batas          g.setColor(new Color(18,202,214));          g.drawString("(0,320)", this.CenterX-45, 45);          g.drawString("(0,-340)", this.CenterX-50, 705);          g.drawString("(630,0)", 1230, this.CenterY+15);          g.drawString("(-630,0)", 10, this.CenterY+15);          // Draw X+ dan Y+          g.drawString("(0,0)", this.CenterX+3, this.CenterY-5);          g.setFont(new Font("TimesRoman", Font.PLAIN, 20));          g.drawString("x+", 1250, this.CenterY-5);          g.drawString("y+", this.CenterX+5, 45);          // Draw Convex Hull          g.setColor(new Color(255,182,119));          g2.setStroke(new BasicStroke(3));          for (Point[] A : this.SetTupleOfLine) {              int x1 = CenterX + (int)A[0].getX();  int y1 = CenterY - (int)A[0].getY();              int x2 = CenterX + (int)A[1].getX();  int y2 = CenterY - (int)A[1].getY();              g2.drawLine(x1, y1, x2, y2);          }          // Draw All Points          g.setColor(new Color(212,93,121));          for (int i = 0; i < this.vectorOfPoints.size(); i++) {              int x = CenterX + (int)this.vectorOfPoints.get(i).getX();              int y = CenterY - (int)this.vectorOfPoints.get(i).getY();              g.fillOval(x-2, y-2, 5, 5);          }      }  } |

1. ***Screenshot* Program**

Program dijalankan pada komputer dengan spesifikasi:

Processor : Intel Core i7-8585U

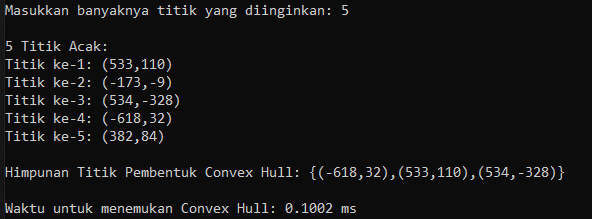
Memori : 16 GB DDR4

Hard Disk : 1TB SSD M.2 PCIe

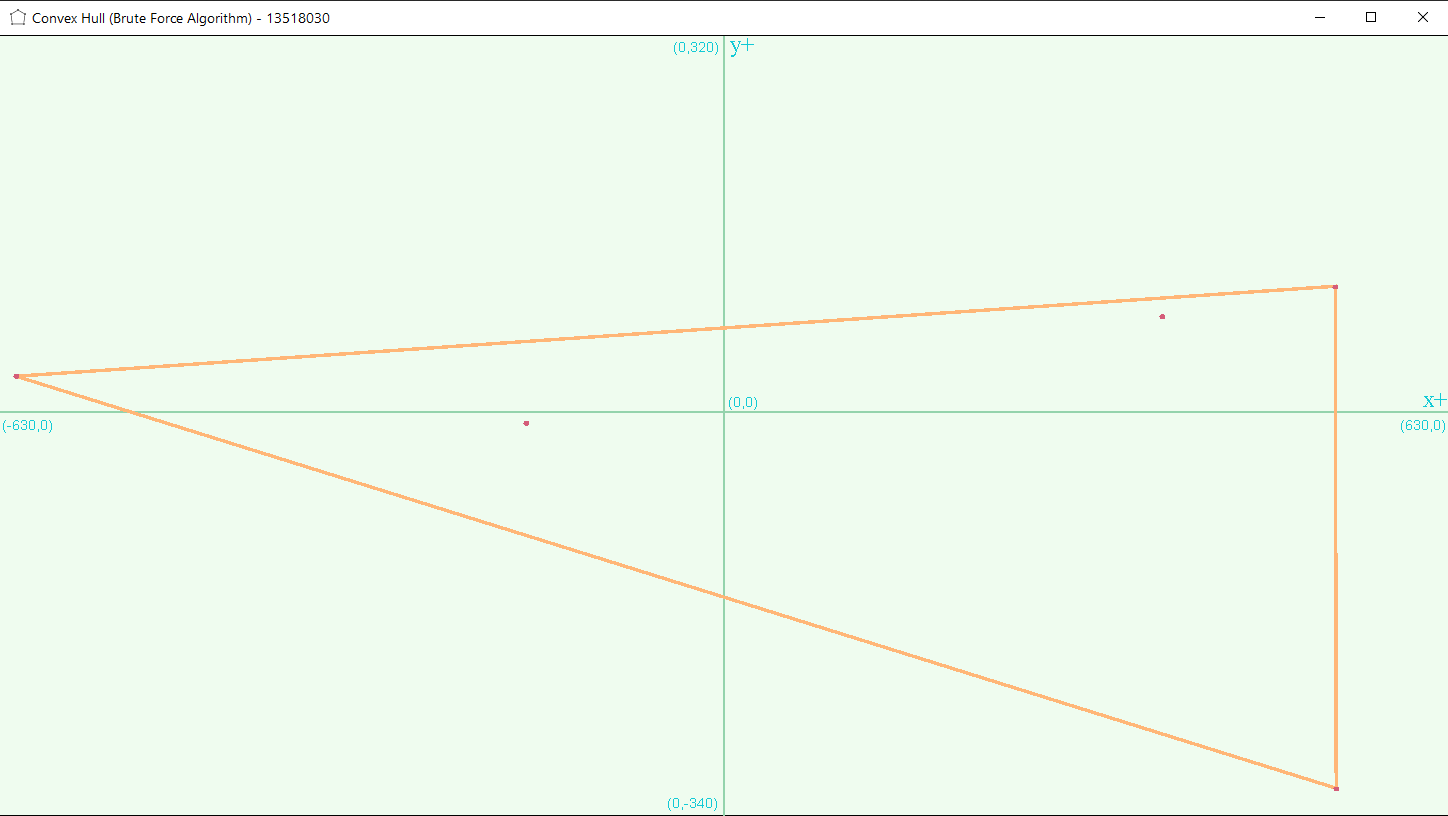
Kartu Grafis : NVIDIA GeForce MX 230 (2GB GDDR5)

1. **N = 5**

Pada percobaan dengan banyak titik (N) = 5, dibangkitkan titik-titik dengan koordinat (533,110), (-173,-9), (534,-328), (-618,32), (382,84). Dengan menggunakan algoritma brute force (waktu pencarian: 0.1002 ms) didapatkan himpunan pembentuk *Convex Hull* berjumlah 3 titik yaitu:

{(-618,32), (533,110), (534,-328)}

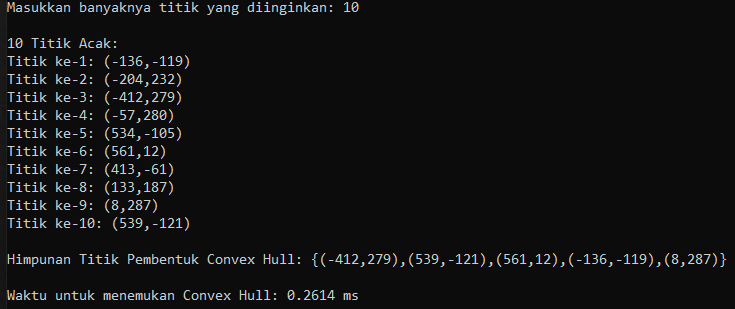
Gambar 1. *Convex Hull* untuk jumlah titik N = 5



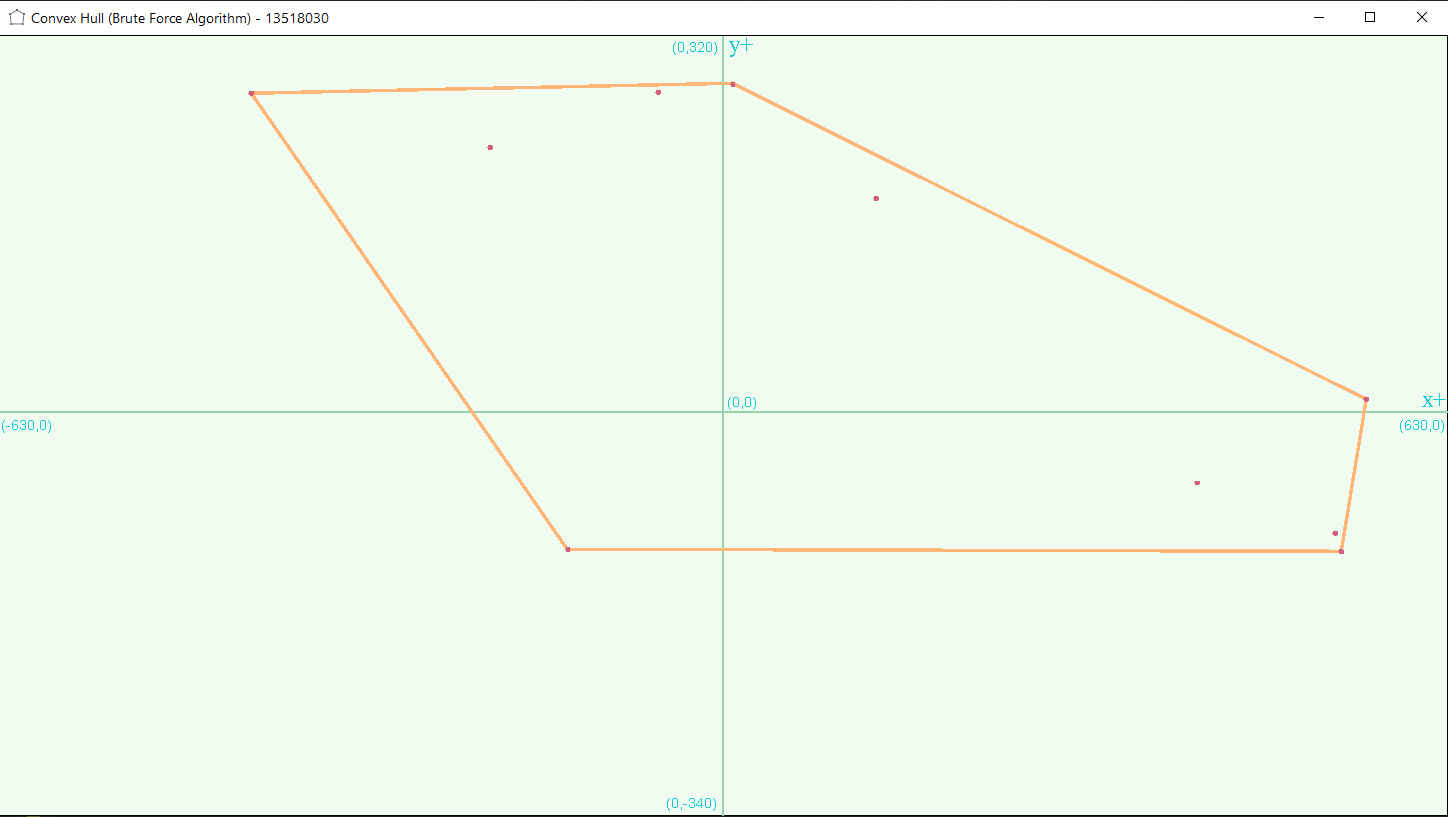
Gambar 2. Plotting *Convex Hull* untuk jumlah titik N = 5

1. **N = 10**

Pada percobaan dengan banyak titik (N) = 10, dibangkitkan titik-titik dengan koordinat (-136,-119), (-204,232), (-412,279), (-57,280), (534,-105), (561,12), (413,-61), (133,187), (8,287), (539,-121). Dengan menggunakan algoritma brute force (waktu pencarian: 0.2614 ms) didapatkan himpunan pembentuk *Convex Hull* berjumlah 5 titik yaitu:

{(-412,279), (539,-121), (561,12), (-136,-119), (8,287)} 

Gambar 3. *Convex Hull* untuk jumlah titik N = 10

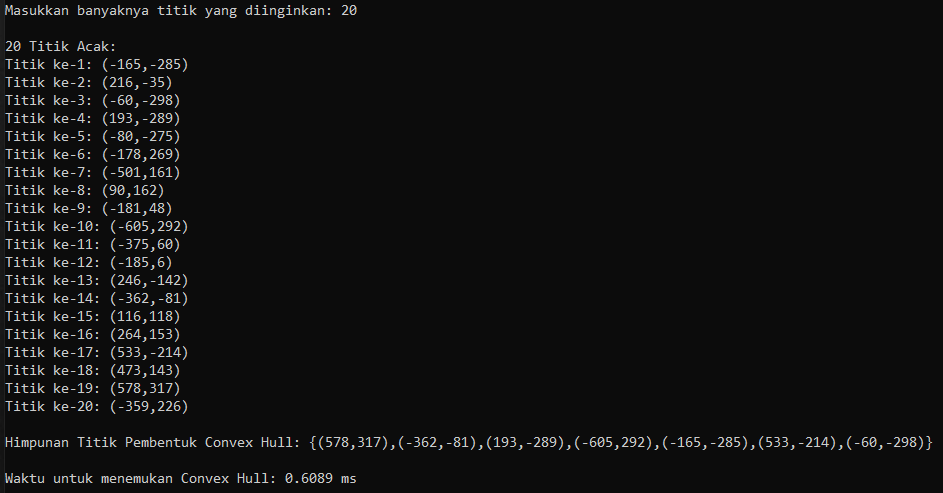
****

Gambar 4. Plotting *Convex Hull* untuk jumlah titik N = 10

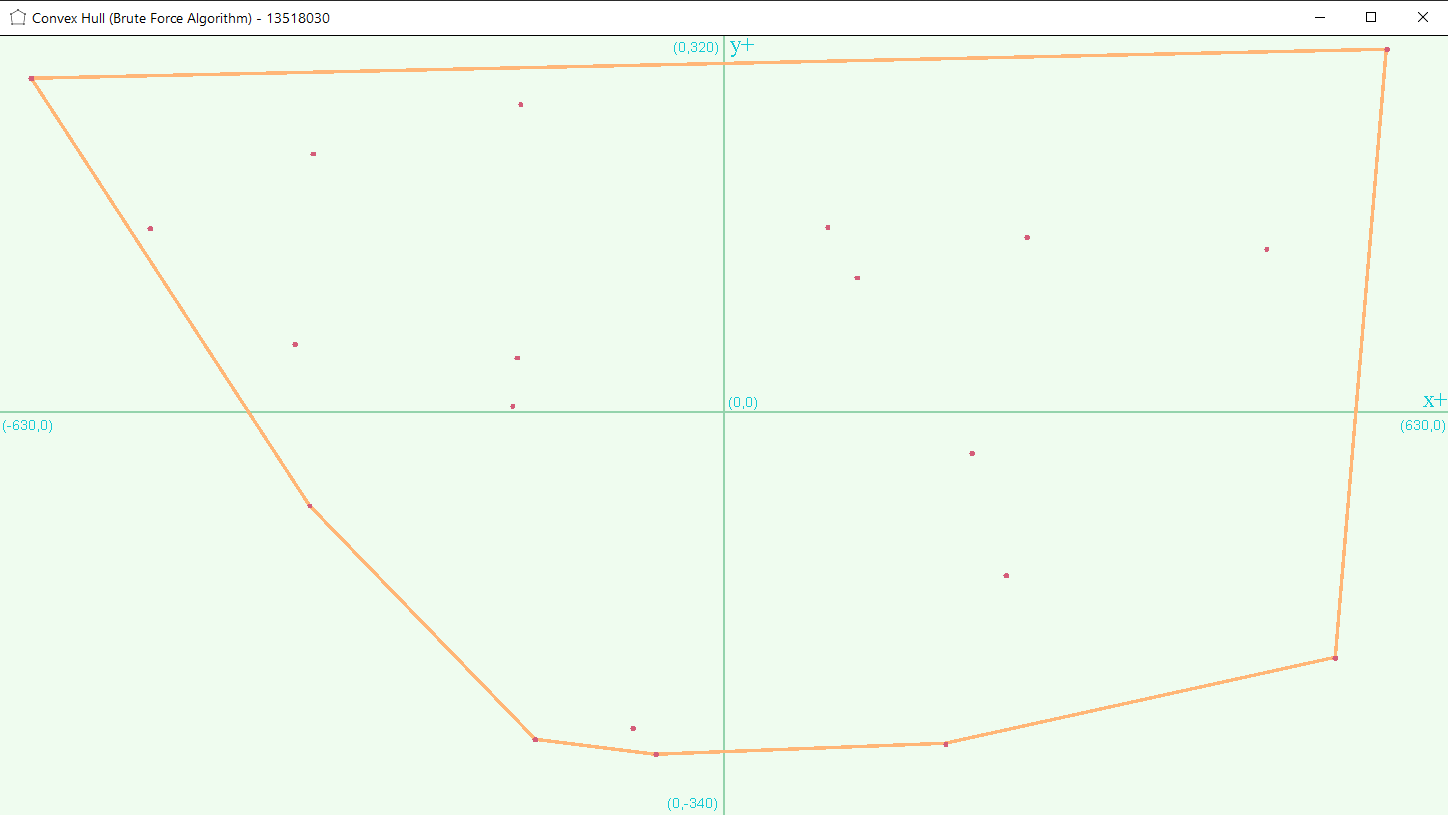
1. **N = 20**

Pada percobaan dengan banyak titik (N) = 20, dibangkitkan titik-titik dengan koordinat berikut: (-165,-285), (216,-35), (-60,-298), (193,-289), (-80,-275), (-178,269), (-501,161), (90,162), (-181,48), (-605,292), (-375,60), (-185,6), (246,-142), (-362,-81), (116,118), (264,153), (533,-214), (473,143), (578,317), (-359,226). Dengan menggunakan algoritma brute force (waktu pencarian: 0.6089 ms) didapatkan himpunan pembentuk *Convex Hull* berjumlah 7 titik yaitu:

{(578,317), (-362,-81), (193,-289), (-605,292), (-165,-285), (533,-214), (-60,-298)}



Gambar 5. *Convex Hull* untuk jumlah titik N = 20



Gambar 6. Plotting *Convex Hull* untuk jumlah titik N = 20

1. **Tabel Penilaian**

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