

**TUGAS KECIL I**  
**IF2211 STRATEGI ALGORITMA**  
**PENYELESAIAN PERSOALAN CONVEX HULL DENGAN**  
**ALGORITMA *BRUTE FORCE***

**LAPORAN TUGAS BESAR**

Diajukan untuk memenuhi tugas mata kuliah IF2211 Strategi Algoritma

Oleh

**MUHAMMAD FAUZAN AL-GHIFARI**  
**(13518112 - K01)**



**INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**

**2020**

## A. Algoritma *Brute Force* dan Kompleksitasnya

Algoritma *Brute Force* merupakan jenis algoritma yang bersifat straight forward dan memiliki kompleksitas yang tinggi. Pemecahan masalah dengan *brute force* dilakukan berdasarkan pada pernyataan soal dan defenisi konsep.

Pada Tugas Kecil 1 ini saya menyelesaikan permasalahan pembuatan *convex hull* dari sejumlah titik acak menggunakan algoritma *brute force* dalam bahasa C++. Berikut adalah potongan algoritma *brute force* yang telah saya buat beserta kompleksitasnya.

```
auto start = high_resolution_clock::now(); // Mulai menghitung waktu
// Memulai algoritma convex hull
// Menentukan titik terkiri sebagai titik pertama
lm = 0;
for (i = 0; i < n; i++){
    if(arr[i].x < arr[lm].x){
        lm = i;
    }
}

// Menentukan titik berikutnya yg termasuk himpunan
// titik pembentuk convex hull
if (n <= 2){
    hull.push_back(arr[0]);
    hull.push_back(arr[1]);
}
else{
    current = lm;
    prev = 999;
    hull.push_back(arr[current]);
    Stop1 = false;
    do{
        i = 0;
        Stop2 = false;
        while (Stop2 == false and i < n){
            j = 0;
            Stop3 = false;
            result = 0;
            total = 0;
            count = 0;

            while(Stop3 == false and j < n){
                if (i != current and i != prev and i != j and current != j){
                    result = IsCH(arr[current], arr[i], arr[j]);
                    if (result == 100){
                        Jika terdapat titik yang kolinear Do nothing
                    }
                    else if (result != 100){
                        count++;
                        total += result;
                    }
                }
                j++;
            }
            Stop3 = true;
        }
        Stop2 = true;
    }while(count < n);
}
```

```

        if (count == n-2 and ((total/count == 1) or (total/count == 2))){
            if (i != lm) {
                hull.push_back(arr[i]);
                prev = current;
                current = i;
                Stop2 = true;
            }
            else if (i == lm){
                Stop1 = true;
            }
        }
    }
}
j++;
}
i++;
}
} while (!(Stop1)); //sudah sekali putaran dan mencapai titik awal
hull.push_back(arr[lm]);
}

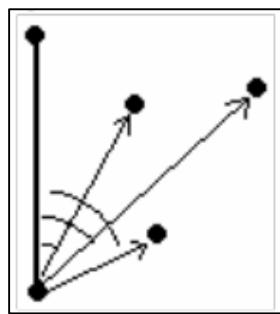
auto stop = high_resolution_clock::now(); // Berhenti menghitung waktu

```

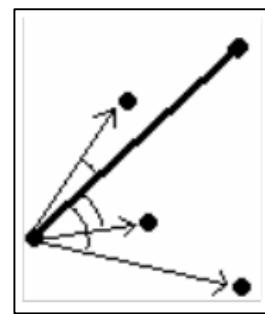
### Penjelasan Algoritma *Brute Force*

Metode Brute Force yang saya buat ini menggunakan tiga buah loop, yaitu satu buah loop *do-while* dan dua buah loop *while*. Awalnya program menentukan titik paling kiri dari kumpulan titik yang tersedia sebagai acuan *convex hull*.

1. Loop pertama (*do-while*) berfungsi untuk mengecek setiap titik apakah memenuhi kriteria sebagai pembentuk *convex hull* atau tidak.
2. Loop kedua (*while*) berfungsi untuk mencari titik selanjutnya yang mungkin memenuhi kriteria *convex hull* dari titik yang dipilih dari loop pertama.
3. Loop ketiga (*while*) berfungsi untuk memasangkan titik yang sudah dipilih pada loop pertama dan kedua dengan semua titik yang ada dan mengeceknya dengan fungsi *isCH*. Jika semua titik yang dipasangkan pada loop ketiga menghasilkan hasil yang sama semua (semua titik berada di sebelah kiri atau kanan dari persamaan garis yang dibentuk oleh titik di loop 1 dan loop 2) maka titik tersebut termasuk pembangun *convex hull*.



Termasuk penyusun convex hull



Tidak termasuk penyusun convex hull

## Kompleksitas Algoritma

Algoritma yang saya buat menggunakan loop *do-while* dan *while*, sehingga memungkinkan adanya kasus terbaik dan terburuk ditemukannya kumpulan titik penyusun *convex hull* tergantung kepada penyusunan daftar titik yang tersedia.

### 1. Loop 1

#### a. Kasus Terbaik $O(3)$

Terjadi saat seluruh titik dapat dilingkupi oleh hanya 3 titik dan loop 1 langsung menemukan ketiga titik tersebut secara berturut-turut.

#### b. Kasus Terburuk $O(n)$

Terjadi saat titik terakhir yang memenuhi kriteria *convex hull* dicek pada urutan terakhir, sehingga semua titik harus dicek terlebih dahulu.

### 2. Loop 2

#### a. Kasus Terbaik $O(1)$

Terjadi saat loop 2 langsung menemukan titik yang jika dipasangkan dengan titik pada loop 1 membentuk sebuah persamaan garis yang menyebabkan semua titik selain titik tersebut berada di sebelah kanan atau kiri dari garis.

#### b. Kasus terburuk $O(n)$

Terjadi saat titik terakhir yang memenuhi kriteria *convex hull* dicek pada urutan terakhir, sehingga semua titik harus dicek terlebih dahulu.

### 3. Loop 3 $O(n)$

Semua titik harus dicek apakah berada pada sebelah kiri atau kanan dari persamaan garis yang dibentuk, sehingga kompleksitasnya  $O(n)$ .

Sehingga dari pemaparan di atas dapat disimulkan bahwa kompleksitas algoritma *Brute Force* pada pencarian titik penyusun *Convex Hull* sebagai berikut

- a. Pada kasus terbaik adalah  $T(n) = O(3n)$
- b. Pada kasus terburuk adalah  $T(n) = O(n^3)$

## B. Kode Program

```
#include <iostream>
#include <vector>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <algorithm>
#include <chrono>
#include "matplotlibcpp.h"

using namespace std::chrono;
using namespace std;
namespace plt = matplotlibcpp;

struct Point{
    // Struktur Point
    int x;
    int y;
};

void CetakPoint (Point p){
    // Prosedur untuk mencetak point
    cout << "(" << p.x << "," << p.y << ")";
}

int IsCH (Point a, Point b, Point c){
    // Menentukan sebuah titik termasuk penyusun
    // dari convex hull atau bukan
    // Menggunakan persamaan garis yang dibentuk
    // oleh dua buah titik a dan b
    // (y-y1)*(x2-x1)-(x-x1)*(y2-y1)
    float nilai;
    nilai = (c.y-a.y)*(b.x-a.x) - (c.x-a.x)*(b.y-a.y);
    if (nilai == 0){
        return 100;           // ada di garis
    }

    else if (nilai > 0){
        return 1;             // ada di kanan garis
    }
}
```

```

        else{ // nilai < 0
            return 2;           // ada di kiri garis
        }

    }

int main() {
    int n;                      // jumlah titik
    int i, j;                   // variabel untuk Loop
    int lm;                     // index Leftmost
    int current, prev;
    float result, total, count;
    bool Stop1, Stop2, Stop3;

    //Meminta input banyaknya titik dari user
    cout << "Masukkan banyaknya titik : ";
    cin >> n;
    while (n < 2 or n <= 0){
        cout << "Minimal n bernilai 2" << endl;
        cout << "Masukkan banyaknya titik : ";
        cin >> n;
    }
    vector<Point> arr;
    vector<Point> hull;
    Point p[n];

    // Membangun titik secara random
    srand(time(0));
    cout << endl << "Daftar Poin : " << endl;
    for (i = 0; i < n; i++){
        p[i].x = rand() % 1000;
        p[i].y = rand() % 1000;
        arr.push_back(p[i]);
        CetakPoint(arr[i]);
        cout << endl;
    }
    cout << endl;
}

```

Kode di atas merupakan persiapan awal. Pertama program meminta input jumlah titik dari user sebanyak n. Kemudian program membangun pasangan titik (*Point*) sebanyak n secara random dan mencetaknya ke layar. Pada program kali ini angka random yang mungkin muncul adalah berkisar dari 0 sampai 1000. Kemudian program dilanjutkan dengan algoritma *Brute Force* yang sudah dijelaskan sebelumnya pada bagian A.

```

auto start = high_resolution_clock::now(); // Mulai menghitung waktu
// Memulai algoritma convex hull
// Menentukan titik terkiri sebagai titik pertama
lm = 0;
for (i = 0; i < n; i++){
    if(arr[i].x < arr[lm].x){
        lm = i;
    }
}

// Menentukan titik berikutnya yg termasuk himpunan
// titik pembentuk convex hull
if (n <= 2){
    hull.push_back(arr[0]);
    hull.push_back(arr[1]);
}
else{
    current = lm;
    prev = 999;
    hull.push_back(arr[current]);
    Stop1 = false;
    do{
        i = 0;
        Stop2 = false;
        while (Stop2 == false and i < n){
            j = 0;
            Stop3 = false;
            result = 0;
            total = 0;
            count = 0;

            while(Stop3 == false and j < n){
                if (i != current and i != prev and i != j and current != j){
                    result = IsCH(arr[current], arr[i], arr[j]);
                    if (result == 100){
                        // Jika terdapat titik yang kolinear Do Nothing
                    }
                    else if (result != 100){
                        count++;
                        total += result;
                    }
                }
                if (count == n-2 and ((total/count == 1) or (total/count == 2))){
                    if (i != lm) {
                        hull.push_back(arr[i]);
                        prev = current;
                        current = i;
                        Stop2 = true;
                    }
                    else if (i == lm){
                        Stop1 = true;
                    }
                }
                j++;
            }
            i++;
        }
    } while (!Stop1); //sudah sekali putaran dan mencapai titik awal
    hull.push_back(arr[lm]);
}

auto stop = high_resolution_clock::now(); // Berhenti menghitung waktu

```

Setelah algoritma *Brute Force* dijalankan, titik-titik yang memenuhi kriteria *Convex Hull* akan disimpan pada `Vector<Point> hull` untuk kemudian dicetak ke layar dan ditampilkan visualisasi datanya. Untuk visualisasi data saya menggunakan `matplotlib` yang sudah dimodifikasi sehingga dapat digunakan pada C++. Persyaratan untuk bisa menampilkan visualisasi data pada program saya adalah menggunakan Linux yang sudah terinstall C++, python 3.7, python-dev `matplotlib` dan `numpy`. Untuk lebih jelas dapat membaca file `readme`.

```
// Mencetak titik-titik convex hull
cout << "Titik penyusun convex hull : " << endl;
cout << "[";
for (int i = 0; i < hull.size()-1; i++) {
    CetakPoint(hull[i]);
    cout << ", ";
}
CetakPoint(hull[hull.size()-1]);
cout << "]" << endl;

auto duration = duration_cast<microseconds>(stop - start).count();
cout << "Waktu yang dibutuhkan untuk membentuk convex hull : " << duration
    << " ms" << endl;

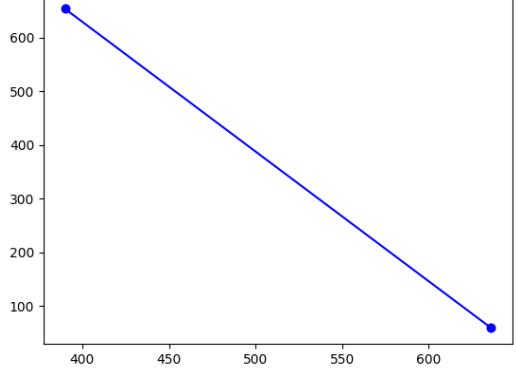
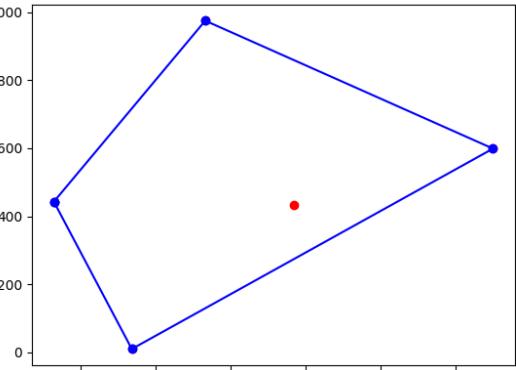
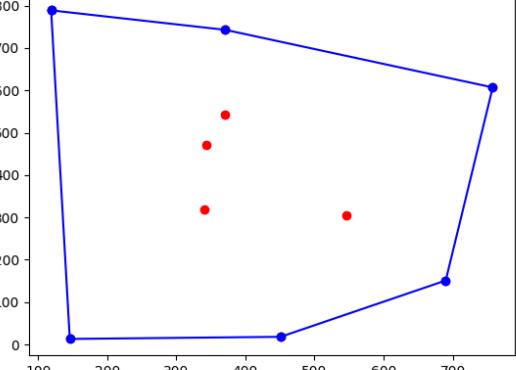
std::vector<int> a;
std::vector<int> b;
std::vector<int> c;
std::vector<int> d;

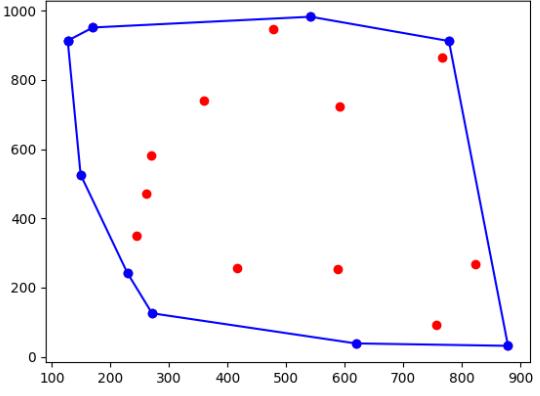
for (i = 0; i < hull.size(); i++){
    a.push_back(hull[i].x);
    b.push_back(hull[i].y);
}

for (i = 0; i < arr.size(); i++){
    c.push_back(arr[i].x);
    d.push_back(arr[i].y);
}

plt::plot(c,d,"ro");
plt::plot(a,b,"bo-");
plt::show();
return 0;
}
```

### C. Screenshot Input-Output program

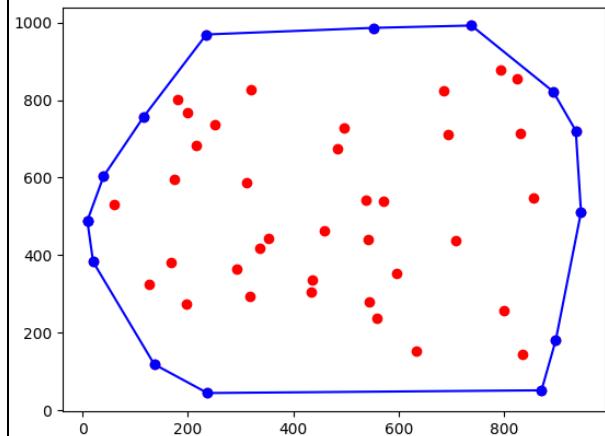
Input	Output	Visualisasi
2	<p>Masukkan banyaknya titik : 2</p> <p>Daftar Poin :</p> <p>(390,654) (636,59)</p> <p>Titik penyusun convex hull :</p> <p>[(390,654), (636,59)]</p> <p>Waktu yang dibutuhkan untuk membentuk convex hull : 15 microseconds</p>	
5	<p>Masukkan banyaknya titik : 5</p> <p>Daftar Poin :</p> <p>(233,975) (184,10) (132,443) (292,434) (425,599)</p> <p>Titik penyusun convex hull :</p> <p>[(132,443), (233,975), (425,599), (184,10), (132,443)]</p> <p>Waktu yang dibutuhkan untuk membentuk convex hull : 32 microseconds</p>	
10	<p>Masukkan banyaknya titik : 10</p> <p>Daftar Poin :</p> <p>(451,18) (341,318) (690,151) (371,543) (119,789) (546,305) (343,472) (146,13) (371,743) (758,607)</p> <p>Titik penyusun convex hull :</p> <p>[(119,789), (146,13), (451,18), (690,151), (758,607), (371,743), (119,789)]</p>	

	<p>Waktu yang dibutuhkan untuk membentuk convex hull : 111 microseconds</p>	
20	<p>Masukkan banyaknya titik : 20</p> <p>Daftar Poin :</p> <ul style="list-style-type: none"> <li>(127,913)</li> <li>(879,32)</li> <li>(478,947)</li> <li>(149,526)</li> <li>(270,582)</li> <li>(360,741)</li> <li>(778,912)</li> <li>(170,951)</li> <li>(767,863)</li> <li>(542,982)</li> <li>(588,255)</li> <li>(823,268)</li> <li>(271,126)</li> <li>(229,242)</li> <li>(245,349)</li> <li>(591,724)</li> <li>(262,470)</li> <li>(756,93)</li> <li>(417,257)</li> <li>(619,39)</li> </ul> <p>Titik penyusun convex hull :</p> <ul style="list-style-type: none"> <li>[(127,913), (149,526), (229,242), (271,126), (619,39), (879,32), (778,912), (542,982), (170,951), (127,913)]</li> </ul> <p>Waktu yang dibutuhkan untuk membentuk convex hull : 487 microseconds</p> 	
50	<p>Masukkan banyaknya titik : 50</p> <p>Daftar Poin :</p> <ul style="list-style-type: none"> <li>(596,352)</li> <li>(235,969)</li> <li>(936,721)</li> <li>(483,674)</li> <li>(800,258)</li> <li>(497,729)</li> <li>(216,682)</li> <li>(831,715)</li> <li>(126,326)</li> <li>(251,738)</li> </ul>	

(336,418)  
(312,587)  
(946,511)  
(545,279)  
(893,822)  
(9,489)  
(174,597)  
(458,462)  
(318,293)  
(136,119)  
(552,986)  
(200,768)  
(20,383)  
(835,146)  
(709,438)  
(237,45)  
(856,549)  
(633,154)  
(60,530)  
(433,305)  
(352,443)  
(794,878)  
(40,604)  
(693,710)  
(898,181)  
(181,802)  
(167,381)  
(570,539)  
(116,757)  
(686,825)  
(196,275)  
(871,52)  
(824,856)  
(559,237)  
(738,992)  
(542,442)  
(435,337)  
(320,827)  
(293,365)  
(538,543)

Titik penyusun convex hull :  
[(9,489), (20,383), (136,119), (237,45),  
(871,52), (898,181), (946,511), (936,721),  
(893,822), (738,992), (552,986),  
(235,969), (116,757), (40,604), (9,489)]

Waktu yang dibutuhkan untuk membentuk  
convex hull : 5441 microseconds



**Spesifikasi komputer yang digunakan**

Nama Laptop : Asus E202SA  
OS : Lubuntu 19.04  
RAM : 2048 MB / 2GB  
Processor : Intel(R) Celeron(R) CPU N3060 @1.60Ghz (2 CPUs), ~1.6Ghz

No	Point	Ya	Tidak
1	Program berhasil dikompilasi	✓	
2	Program berhasil <i>running</i>	✓	
3	Program dapat menerima input dan menuliskan output	✓	
4	Luaran sudah benar untuk semua n	✓	