

LAPORAN TUGAS KECIL 2
IF2211 STRATEGI ALGORITMA
MEMBANGUN KURVA BEZIER DENGAN ALGORITMA TITIK TENGAH BERBASIS
DIVIDE AND CONQUER



Oleh:

Hayya Zuhailii Kinasih

13522102

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024

DAFTAR ISI

DAFTAR ISI	1
BAB I ALGORITMA MEMBANGUN BEZIER CURVE	2
1.1. Implementasi Algoritma Brute Force	2
1.2 Implementasi Algoritma Divide and Conquer	2
Langkah-langkah implementasi algoritma divide and conquer adalah sebagai berikut:	2
BAB 2 IMPLEMENTASI PROGRAM	3
BAB 3 PENGUJIAN	8
3.1 Hasil Pengujian	8
3.2 Analisis	12
LAMPIRAN	15

BAB I

ALGORITMA MEMBANGUN BEZIER CURVE

1.1. Implementasi Algoritma Brute Force

Langkah-langkah implementasi algoritma dengan brute force adalah sebagai berikut:

1. Menghitung titik tengah Q_n antara semua titik kontrol. Menghasilkan array yang terdiri atas titik awal, titik tengah Q_n , dan titik akhir.
2. Plot grafik array di atas.
3. Menghitung titik tengah R_n antara titik tengah Q_n yang telah didapatkan.
4. Jika jumlah iterasi lebih dari satu, sisipkan titik-titik R_n di antara titik-titik Q_n sesuai urutan.
5. Panggil kembali fungsi secara rekursif dengan mengurangi jumlah iterasi sebanyak satu.
6. Jika jumlah iterasi sama dengan satu, plot grafik titik awal, titik tengah R , dan titik akhir.

1.2 Implementasi Algoritma Divide and Conquer

Langkah-langkah implementasi algoritma divide and conquer adalah sebagai berikut:

1. Plot ketiga titik P_0 , P_1 , dan P_2 .
2. Jika jumlah iterasi lebih dari nol, hitung titik Q_0 dan Q_1 di antara ketiga titik awal, dan titik R_0 di antara titik-titik Q .
3. Panggil kembali fungsi secara rekursif dengan titik P_0 , Q_0 , dan R_0 . Jumlah iterasi dikurangi dengan satu. Ini akan memproses bagian kiri dari kurva.
4. Simpan R_0 dalam array untuk kurva final.
5. Panggil kembali fungsi secara rekursif dengan titik R_0 , Q_1 , dan P_2 . Jumlah iterasi dikurangi dengan satu. Ini akan memproses bagian kanan dari kurva.
6. Fungsi berakhir ketika jumlah iterasi = 0.
7. Plot kurva final.

BAB 2

IMPLEMENTASI PROGRAM

File bruteforce.py

```
from util import mid_point
import matplotlib.pyplot as plt

def outer_midpoints(xpoints, ypoints):
    xpoints.insert(1, mid_point(xpoints[0], xpoints[1]))
    ypoints.insert(1, mid_point(ypoints[0], ypoints[1]))
    n = len(xpoints)
    for i in range(2, n-1):
        xpoints[i] = mid_point(xpoints[i], xpoints[i+1])
        ypoints[i] = mid_point(ypoints[i], ypoints[i+1])

def inner_midpoints(xpoints, ypoints):
    innerx = []
    innery = []
    n = len(xpoints)
    for i in range(1, n-2):
        innerx.append(mid_point(xpoints[i], xpoints[i+1]))
        innery.append(mid_point(ypoints[i], ypoints[i+1]))
    return innerx, innery

def bezierBruteForce(xpoints, ypoints, iter):

    outer_midpoints(xpoints, ypoints)
    plt.plot(xpoints, ypoints, 'k.-', linewidth=0.25)
    innerx, innery = inner_midpoints(xpoints, ypoints)

    if (iter == 1):
        innerx.insert(0, xpoints[0])
        innery.insert(0, ypoints[0])
        innerx.append(xpoints[-1])
        innery.append(ypoints[-1])
```

```

plt.plot(innerx, innery, 'b.-', linewidth=2)
else:
    for i in range (1, len(innerx) + 1):
        xpoints.insert(i*2, innerx[i-1])
        ypoints.insert(i*2, innery[i-1])

    bezierBruteForce(xpoints, ypoints, iter-1)

```

File dividenconquer.py

```

from util import mid_point
import matplotlib.pyplot as plt

def mid_of_three(xpoints, ypoints):
    x1, x2 = mid_point(xpoints[0], xpoints[1]), mid_point(xpoints[1],
xpoints[2])
    y1, y2 = mid_point(ypoints[0], ypoints[1]), mid_point(ypoints[1],
ypoints[2])
    return [x1, mid_point(x1, x2), x2], [y1, mid_point(y1, y2), y2]

def bezierDnC(xpoints, ypoints, iter, final):

    if (iter > 0):
        xmid, ymid = mid_of_three(xpoints, ypoints)
        plt.plot(xmid, ymid, 'k.-', linewidth=0.25)
        #left
        bezierDnC([xpoints[0]] + xmid[:2], [ypoints[0]] + ymid[:2],
iter-1, final)
        final[0].append(xmid[1])
        final[1].append(ymid[1])
        #right
        bezierDnC(xmid[-2:] + [xpoints[2]], ymid[-2:] + [ypoints[2]],
iter-1, final)

```

File main.py

```
from dividenconquer import bezierDnC
from bruteforce import bezierBruteForce
from util import file_input, manual_input
import matplotlib.pyplot as plt
import time

inp_method = int(input("1. Command line\n2. File\n Pilih metode input:
"))
while (inp_method != 1 and inp_method != 2):
    print("Input tidak sesuai.")
    inp_method = int(input("Pilih metode input: "))

if (inp_method == 1):
    xpoints, ypoints, n_iter = manual_input()
else:
    xpoints, ypoints, n_iter = file_input()

# Divide and Conquer
start = time.time()
plt.figure(1)
final = [[], []]
bezierDnC(xpoints, ypoints, n_iter, final)
plt.plot([xpoints[0]] + final[0] + [xpoints[2]], [ypoints[0]] + final[1]
+ [ypoints[2]], 'b.-', linewidth=2)
elapsed = time.time() - start
plt.title("Divide and Conquer")
plt.figtext(0, 0, 'Elapsed time = ' + str(elapsed))

#Brute Force
start = time.time()
plt.figure(2)
plt.plot(xpoints, ypoints, 'k.-', linewidth=0.25)
bezierBruteForce(xpoints, ypoints, n_iter)
elapsed = time.time() - start
plt.title("Brute Force")
plt.figtext(0, 0, 'Elapsed time = ' + str(elapsed))
```

```
plt.show()
```

File util.py

```
import os

def mid_point (x1, x2):
    return (x1+x2)/2

def file_input():
    file = input("\nTuliskan nama file: ")
    file_path = "../test/" + file

    while not os.path.exists(file_path):
        print("File tidak ditemukan. Silahkan input ulang.")
        file = input("Tuliskan nama file yang akan di cek: ")
        file_path = "../test/" + file

    xpoints = []
    ypoints = []

    f = open(file_path, "r")
    for i in range(3):
        line = f.readline().strip()
        print(f"P{i}:", line)
        x, y = line.split()
        xpoints.append(int(x))
        ypoints.append(int(y))

    n_iter = int(f.readline().strip())
    print("Jumlah iterasi:", n_iter)

    f.close()
    return xpoints, ypoints, n_iter

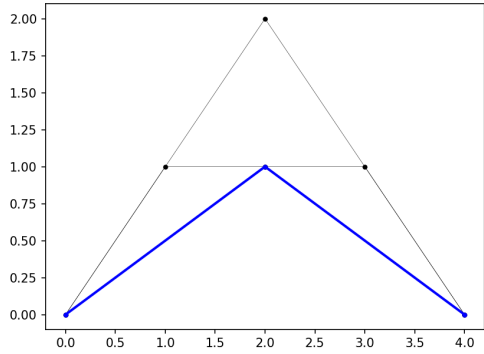
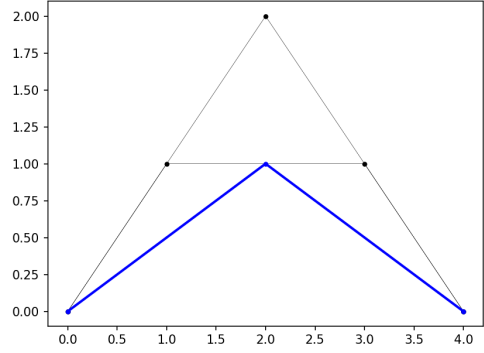
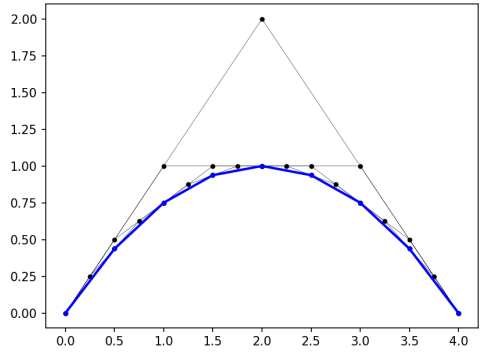
def manual_input():
    xpoints = []
    ypoints = []
```

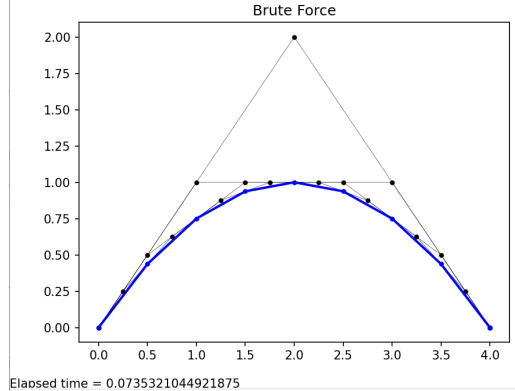
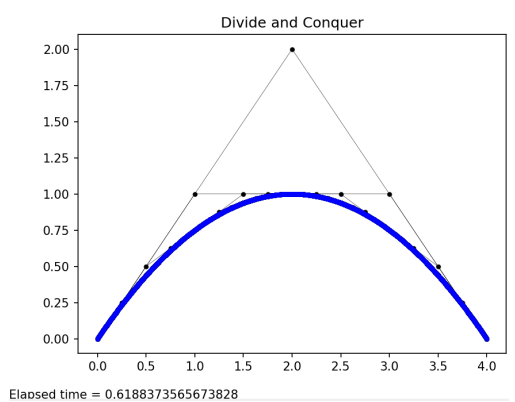
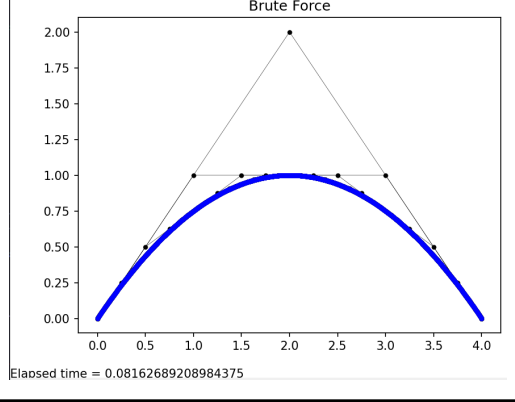
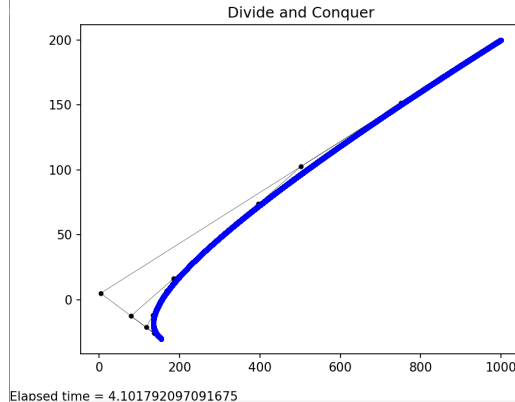
```
for i in range (1, 4):  
    point = input(f"Masukkan titik ke-{i}: ").split()  
    xpoints.append(int(point[0]))  
    ypoints.append(int(point[1]))  
  
n_iter = int(input("Masukkan jumlah iterasi: "))  
  
return xpoints, ypoints, n_iter
```


BAB 3

PENGUJIAN

3.1 Hasil Pengujian

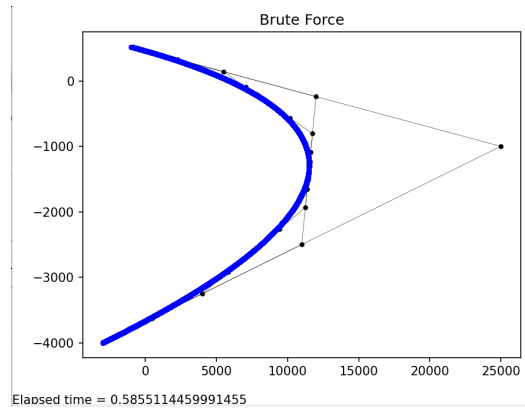
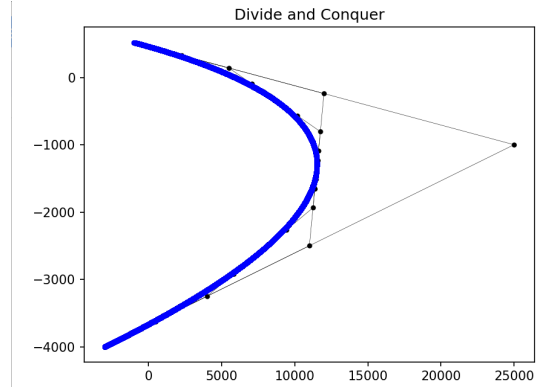
No	Input	Hasil Divide and Conquer
1	<pre> Tuliskan nama file: 1.txt P0: 0 0 P1: 2 2 P2: 4 0 Jumlah iterasi: 1 </pre>	<div style="text-align: center;">  <p>Divide and Conquer</p> <p>Elapsed time = 0.16120123863220215</p> </div> <div style="text-align: center;">  <p>Brute Force</p> <p>Elapsed time = 0.07367277145385742</p> </div>
2	<pre> Tuliskan nama file: 2.txt P0: 0 0 P1: 2 2 P2: 4 0 Jumlah iterasi: 3 </pre>	<div style="text-align: center;">  <p>Divide and Conquer</p> <p>Elapsed time = 0.1629798412322998</p> </div>

		 <p>Elasped time = 0.0735321044921875</p>
3	<pre> Tuliskan nama file: 3.txt P0: 0 0 P1: 2 2 P2: 4 0 Jumlah iterasi: 10 </pre>	 <p>Elasped time = 0.6188373565673828</p>  <p>Elasped time = 0.08162689208984375</p>
4	<pre> Tuliskan nama file: 4.txt P0: 155 -30 P1: 4 5 P2: 1000 200 Jumlah iterasi: 13 </pre>	 <p>Elasped time = 4.101792097091675</p>

		<p>Brute Force</p> <p>Elapsed time = 0.0958988665344238</p>
5	<p>Masukkan titik ke-1: -20 -3 Masukkan titik ke-2: 15 30 Masukkan titik ke-3: -60 41 Masukkan jumlah iterasi: 15</p>	<p>Divide and Conquer</p> <p>Elapsed time = 16.210403442382812</p> <p>Brute Force</p> <p>Elapsed time = 0.22480249404907227</p>

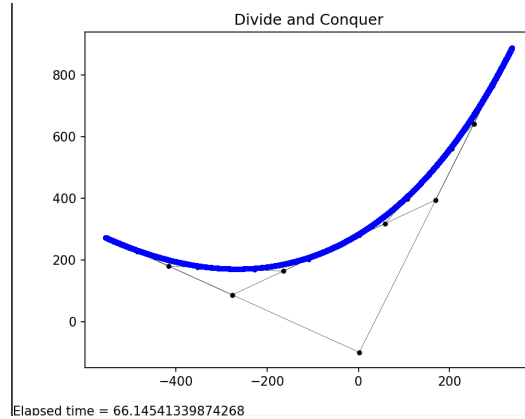
6

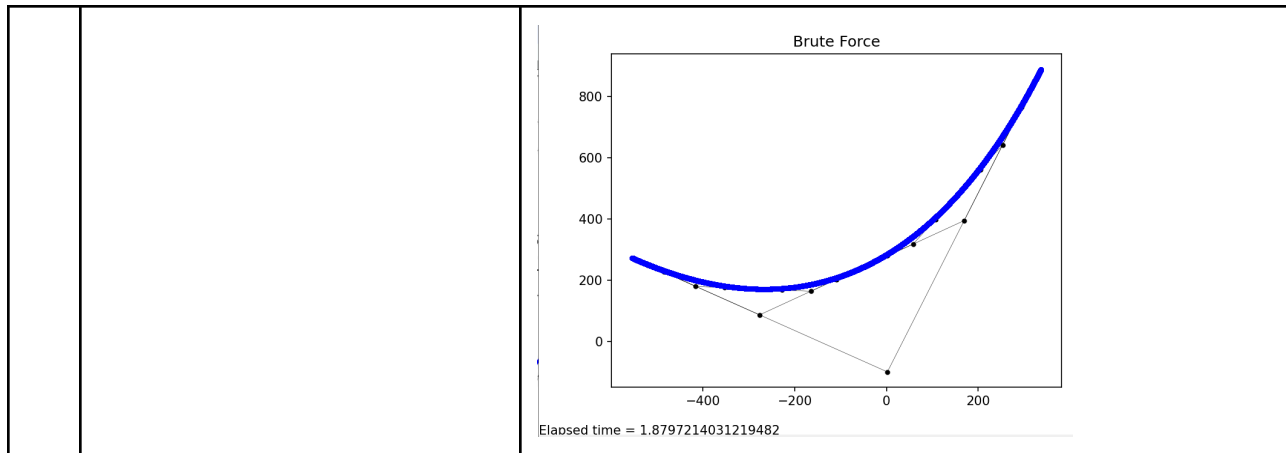
Masukkan titik ke-1: -1000 520
 Masukkan titik ke-2: 25000 -999
 Masukkan titik ke-3: -3000 -3999
 Masukkan jumlah iterasi: 16



7

Masukkan titik ke-1: -555 273
 Masukkan titik ke-2: 2 -100
 Masukkan titik ke-3: 337 890
 Masukkan jumlah iterasi: 17





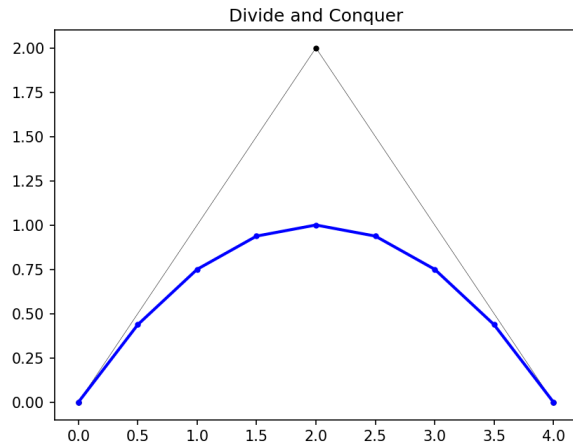
3.2 Analisis

Berdasarkan perbandingan uji, terlihat jelas bahwa algoritma divide and conquer memakan waktu yang lebih banyak untuk memberikan hasil yang sama. Waktu yang dibutuhkan algoritma divide and conquer meningkat dengan cepat seiring bertambahnya jumlah iterasi yang harus dilakukan. Algoritma brute force melakukan perhitungan secara rekursif sebanyak n kali, dimana setiap pemanggilan fungsi, jumlah operasi yang dilakukan meningkat dengan persamaan $3 * 2^n - 2$, atau dapat disimpulkan algoritma brute force memiliki kompleksitas $O(2^n)$. Sedangkan algoritma divide and conquer membagi persoalan menjadi dua dan memroses kedua belah secara rekursif, dengan relasi rekurens $T(n) = 2T(n/2)$, sehingga memiliki kompleksitas $O(n)$.

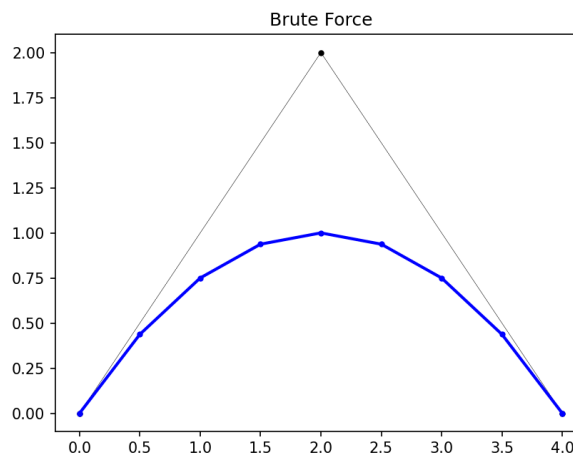
Berdasarkan hal tersebut, seharusnya algoritma divide and conquer lebih cepat membentuk kurva bezier dibandingkan algoritma brute force. Namun, algoritma divide and conquer melakukan plotting pada graf jauh lebih sering dibanding algoritma brute force, dan hal tersebut memakan waktu yang cukup banyak seiring bertambahnya jumlah iterasi. Hal tersebut terlihat ketika fungsi dijalankan tanpa menggambarkan iterasi-iterasi antara pada graf.

Input	Hasil
-------	-------

```
Tuliskan nama file: 2.txt
P0: 0 0
P1: 2 2
P2: 4 0
Jumlah iterasi: 3
```

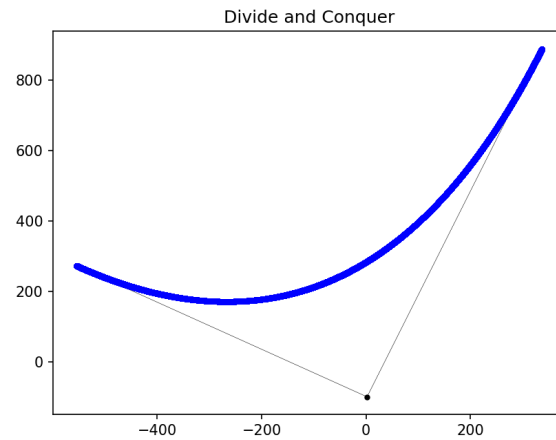


Elapsed time = 0.20959949493408203

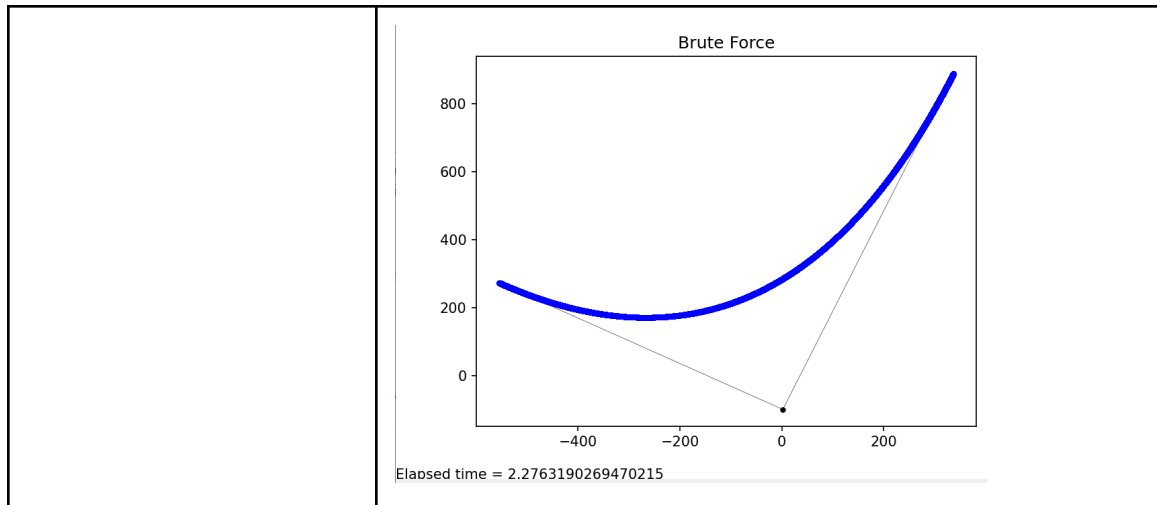


Elapsed time = 0.09444689750671387

```
Tuliskan nama file: 7.txt
P0: -555 273
P1: 2 -100
P2: 337 890
Jumlah iterasi: 17
```



Elapsed time = 0.560492992401123



Terlihat pada tabel bahwa untuk jumlah iterasi yang kecil algoritma brute force masih lebih cepat, namun ketika jumlah iterasi semakin besar, algoritma divide and conquer lebih efisien. Algoritma divide and conquer juga sudah memberikan hasil yang tepat karena hasilnya sama dengan algoritma brute force.

LAMPIRAN

Repository Github

https://github.com/hayyazk/Tucil2_13522102.git

Checklist Pengerjaan

Poin	Ya	Tidak
1. Program berhasil dijalankan.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2. Program dapat melakukan visualisasi kurva Bézier.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3. Solusi yang diberikan program optimal.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4. [Bonus] Program dapat membuat kurva untuk n titik kontrol	<input type="checkbox"/>	<input checked="" type="checkbox"/>
5. [Bonus] Program dapat melakukan visualisasi proses pembuatan kurva.	<input type="checkbox"/>	<input checked="" type="checkbox"/>