LAPORAN TUGAS KECIL 2

IF2211 Strategi Algoritma

Membangun Kurva Bezier dengan Algoritma Titik Tengah berbasis Divide and Conquer

Oleh:

Samy Muhammad Haikal (13522151)

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

DAFTAR ISI

BAB 1
DESKRIPSI MASALAH

1.1 Algoritma Divide and Conquer

Algoritma divide and conquer merupakan paradigma algoritma yang terdiri dari tiga langkah utama: "Divide", "Conquer", dan "Combine". Paradigma ini digunakan untuk memecahkan masalah yang kompleks menjadi submasalah yang lebih kecil, kemudian memecahkan submasalah tersebut secara rekursif hingga menjadi cukup sederhana untuk dipecahkan secara langsung. Setelah itu, hasil dari submasalah tersebut digabungkan kembali untuk mendapatkan solusi dari masalah asal.

Contoh umum penggunaan algoritma Divide and Conquer adalah dalam pencarian biner (binary search) dan dalam beberapa algoritma pengurutan, seperti Merge Sort dan Quick Sort. Dalam pencarian biner, array dipecah menjadi dua bagian setiap kali, dan pencarian dilakukan pada salah satu bagian tergantung pada elemen tengah array. Dengan cara ini, pencarian dapat dilakukan dengan cepat pada data yang terurut.

1.2 Kurva Bezier

Kurva Bézier adalah kurva halus yang sering digunakan dalam desain grafis, animasi, dan manufaktur. Kurva ini dibuat dengan menghubungkan beberapa titik kontrol, yang menentukan bentuk dan arah kurva. Cara membuatnya cukup mudah, yaitu dengan menentukan titik-titik kontrol dan menghubungkannya dengan kurva. Kurva Bézier memiliki banyak kegunaan dalam kehidupan nyata, seperti pen tool, animasi yang halus dan realistis, membuat desain produk yang kompleks dan presisi, dan membuat font yang indah dan unik. Keuntungan menggunakan kurva Bézier adalah kurva ini mudah diubah dan dimanipulasi, sehingga dapat menghasilkan desain yang presisi dan sesuai dengan kebutuhan.

Sebuah kurva Bézier didefinisikan oleh satu set titik kontrol P0 sampai Pn , dengan n disebut order (n = 1 untuk linier, n = 2 untuk kuadrat, dan seterusnya). Titik kontrol pertama dan terakhir selalu menjadi ujung dari kurva, tetapi titik kontrol antara (jika ada) umumnya tidak terletak pada kurva. Pada gambar 1 diatas, titik kontrol pertama adalah P0, sedangkan titik kontrol terakhir adalah P3. Titik kontrol P1 dan P2 disebut sebagai titik kontrol antara yang tidak terletak dalam kurva yang terbentuk.

1.3 Algoritma de Casteljau dan Strategi penyelesaian masalah menggunakan pendekatan Divide and Conquer

Algoritma De Casteljau adalah algoritma yang digunakan untuk mengevaluasi kurva Bezier. Algoritma ini bekerja dengan membagi titik kendali kurva Bezier menjadi dua set titik kendali yang lebih kecil, kemudian mengulangi proses ini secara rekursif hingga hanya tersisa satu titik. Titik tersebut adalah titik pada kurva Bezier yang

dievaluasi. Berikut adalah contoh implementasi algoritma de casteljau dalam bahasa python:

```
def de_casteljau(t, coefs):
      beta = [c for c in coefs] # values in this list are
overridden
    n = len(beta)
    for j in range(1, n):
        for k in range(n - j):
            beta[k] = beta[k] * (1 - t) + beta[k + 1] * t
```

Dalam menyelesaikan persoalan pembuatan kurva bezier penulis menggunakan pendekatan *divide and conquer* dan *bruteforce.* Keuntungan dari strategi ini adalah algoritma ini bisa dipakai untuk n buah titik kontrol, jadi kita tidak terbatas pada tiga titik kontrol saja. Adapun langkah-langkah penyelesaian masalah dalam algoritma dapat dijelaskan sebagai berikut:

1. Program meminta input titik kontrol
2. Program meminta input banyak iterasi
3. Program akan menyalin titik kontrol menjadi dua array, yaitu left dan right, array left akan direverse
4. Program menambahkan titik kontrol pertama ke array kurva bezier
5. Program akan menerapkan algoritma de casteljau terhadap array left dan right sehingga masing-masing array akan menjadi titik kontrol sub kurva pada iterasi selanjutnya
6. Program mengkalkulasi titik tengah dari sub kurva kiri secara rekursif sebanyak iterasi-1
7. Program menambahkan titik tengah kurva utama ke array kurva bezier
8. Program mengkalkulasi titik tengah dari sub kurva kanan secara rekursif sebanyak iterasi-1
9. Program menambahkan titik kurva terakhir ke array kurva bezier
10. Program mengkalkulasi waktu eksekusi dan menampilkan visualisasi dari kurva yang dibentuk

1.4 Pembuatan kurva Bezier dengan menggunakan strategi Bruteforce

Algoritma Bruteforce untuk tiga titik kontrol ini menggunakan rumus yang ada di spek yaitu :.

$$R_0 = B(t) = (1-t)^2 P_0 + (1-t)tP_1 + t^2 P_2, \qquad t \in [0,1]$$

Perlu dilakukan pengulangan dari 0 ke 1 dengan increment 1/ (2^{iterasi} + 1) agar jumlah iterasi sama dengan algoritma divide and conquer. Adapun langkah-langkah penyelesaiannya adalah sebagai berikut:

1. Program meminta input tiga buah titik kontrol
2. Program meminta input banyak iterasi
3. Program mencari titik tengah dari ketiga titik dengan pengulangan dari 0 sampai 1 dengan increment 1/ ($2^{iterasi}$ + 1)
4. Program menambahkan titik tengah tersebut ke dalam array kurva bezier
5. Program mengkallkulasi waktu eksekusi dan menampilkan visualisasi dari kurva yang dibentuk
6.

1.5 Analisis Kompleksitas Algoritma

Kompleksitas algoritma penyususnan kurva bezier menggunakan algoritma bruteforce adalah $O(2^n$ dengan n sebagai n) jumlah iterasi dalam algoritma. Sedangkan kompleksitas waktunya adalahT(n) = $3.(2^n$ +1)+2. Sementara itu kompleksitas algoritma penyusunan kurva bezier menggunakan algoritma divide and conquer adalah $O(2^n$ dengan n sebagai n) jumlah iterasi dalam algoritma. Sementara itu kompleksitas waktunya adalahT(n) = (2^n -1)+1.

Dalam analisis perbandingan kompleksitas algoritma antara algoritma bruteforce dan algoritma divide and conquer untuk penyusunan kurva Bézier dengan tiga titik kontrol, kita dapat mengamati bahwa keduanya memiliki kompleksitas secara teoritis yang serupa yaitu $O(2^n)$. Meskipun kompleksitas sama, tetapi saat dijalankan algoritma divide and conquer akan lebih cepat dibanding algoritma bruteforce, hal ini terjadi karena $T(n)$ dari kedua algoritma berbeda. Algoritma bruteforce membutuhkan lebih banyak langkah untuk mencapai hasil iterasi dibandingkan divide and conquer. Untuk perbedaan kecepatan tidak begitu signifikan (perbedaannya kecil) karena secara pertumbuhan sama-sama eksponensial.

BAB II
IMPLEMENTASI ALGORITMA DALAM BAHASA PYTHON

Dalam pembuatan program ini, penulis menggunakan bahasa pemrograman python. Program terdiri dari file Bezier.py dan main.py

2.1 Bezier.py

File ini berisi algoritma kurva bezier menggunakan pendekatan divide and conquer dan bruteforce. Selain itu juga ada method untuk menampilkan kurva

Functions	Description
`def Bezier(control_points,iter,curve_points, mid_points):`	Fungsi untuk membentuk kurva bezier tiga titik dengan algoritma divide and conquer
`def BF(control_points,iter,curve_points, mid_points):`	Fungsi untuk membentuk kurva bezier tiga titik dengan algoritma bruteforce
`def BezierN(control_points, iter,mid_points, curve_points):`	Fungsi untuk membentuk kurva bezier dengan algoritma divide and conquer
`def BezierLeft(control_points, iter,mid_points, curve_points):`	Sub-fungsi dari BezierN, berfungsi untuk melakukan iterasi pada bagian kiri kurva
`def BezierRight(control_points, iter,mid_points, curve_points):`	Sub-fungsi dari BezierN, berfungsi untuk melakukan iterasi pada bagian kanan kurva
`def show(control_points,iter, ax, curve_points, mid_points):`	Fungsi untuk menampilkan kurva bezier menggunakan axis subplot pada matplotlib

2.2 main.py

File ini merupakan driver utama dari program ini. Hanya terdapat fungsi main untuk menjalankan program.

Methods	Description
`def main():`	Fungsi untuk menjalankan program utama

BAB III
SOURCE CODE PROGRAM

3.1 Repository Program

Repository program dapat diakses dengan tautan github berikut:
https://github.com/rendangmunir/Tucil2_13522151

3.2 Source Code Program

3.2.1 Bezier.py

```
def Bezier(control_points,iter,curve_points, mid_points):
    if (iter==0):
        if (curve_points==[]):
            curve_points=control_points.copy()
        else:
            curve_points.append(control_points[-1])
    else:
        if (curve_points==[]):
            curve_points.append(control_points[0])
                r0   =   (0.5*(control_points[0][0]+control_points[1][0]),
0.5*(control_points[0][1]+control_points[1][1]))
                r1   =   (0.5*(control_points[1][0]+control_points[2][0]),
0.5*(control_points[1][1]+control_points[2][1]))
        r2 = (0.5*(r0[0]+r1[0]), 0.5*(r0[1]+r1[1]))
        mid_points.append(r0)
        mid_points.append(r1)
        mid_points.append(r2)
        Bezier([control_points[0],r0,r2], iter-1,curve_points, mid_points)
                Bezier([r2,r1,control_points[-1]],  iter-1,curve_points,
mid_points)

def BezierN(control_points, iter,mid_points, curve_points):
    if(iter>0):
        n = len(control_points)-1
        left = control_points.copy()
        left = left[::-1]
        right = control_points.copy()
        curve_points.append(control_points[0])
        while(n>0):
            for i in range(n):
```

```
                            right[i]  =  (0.5*(right[i][0]+right[i+1][0]),
0.5*(right[i][1]+right[i+1][1]))
                                left[i]  =  (0.5*(left[i][0]+left[i+1][0]),
0.5*(left[i][1]+left[i+1][1]))
                mid_points.append(right[i])
            if(n==1):
                BezierLeft(left, iter-1, mid_points,curve_points)
                curve_points.append(right[0])
                BezierRight(right, iter-1, mid_points,curve_points)
                curve_points.append(control_points[-1])
            n-=1

def BezierLeft(control_points, iter,mid_points, curve_points):
    if(iter>0):
        n = len(control_points)-1
        left = control_points.copy()
        right = control_points.copy()
        right = right[::-1]
        while(n>0):
            for i in range(n):
                            right[i]  =  (0.5*(right[i][0]+right[i+1][0]),
0.5*(right[i][1]+right[i+1][1]))
                                left[i]  =  (0.5*(left[i][0]+left[i+1][0]),
0.5*(left[i][1]+left[i+1][1]))
                mid_points.append(right[i])
            if(n==1):
                BezierLeft(left, iter-1, mid_points,curve_points)
                curve_points.append(left[0])
                BezierRight(right, iter-1, mid_points,curve_points)
            n-=1

def BezierRight(control_points, iter,mid_points, curve_points):
    if(iter>0):
        n = len(control_points)-1
        left = control_points.copy()
        left = left[::-1]
        right = control_points.copy()
        while(n>0):
            for i in range(n):
```

```
                            right[i]  =  (0.5*(right[i][0]+right[i+1][0]),
0.5*(right[i][1]+right[i+1][1]))
                                left[i]  =  (0.5*(left[i][0]+left[i+1][0]),
0.5*(left[i][1]+left[i+1][1]))
                mid_points.append(right[i])
            if(n==1):
                BezierLeft(left, iter-1, mid_points,curve_points)
                curve_points.append(right[0])
                BezierRight(right, iter-1, mid_points,curve_points)
            n-=1


def show(control_points,iter, ax, curve_points):
    ax.clear()
    x_control, y_control = zip(*control_points)
    x_curve, y_curve = zip(*curve_points)
    ax.plot(x_curve, y_curve, label="Bezier Curve", color="blue")
    ax.scatter(x_curve, y_curve, color="blue")
    ax.plot(x_control, y_control, label="control Points", color="red")
    ax.scatter(x_control, y_control, color="red", label="Control Points")
    ax.set_xlabel("X")
    ax.set_ylabel("Y")
    ax.grid(True)
    ax.legend()
```

3.2.2 main.py

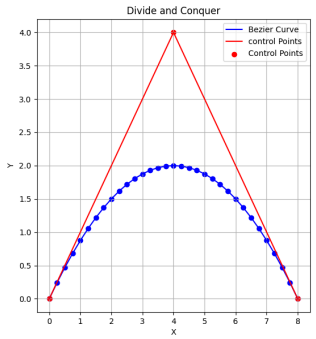
```
import Bezier
import matplotlib.pyplot as plt
import time

control_points = []
mid_points = []
curve_points = []
iter=0

def main():
    global control_points,mid_points,curve_points,iter
    print("Selamat datang di program Bezier curve!")
    print("1. Kurva Bezier 3 titik")
```

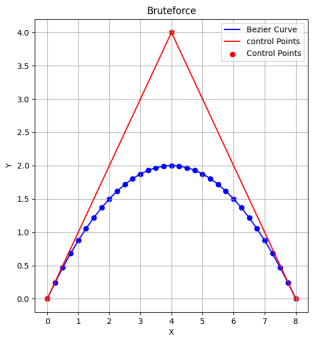
```
    print("2. Kurva Bezier n titik")
    mode = int(input("Pilihan: "))
    if(mode==1):
        for i in range(3):
                point = tuple(map(float, input(f"Masukkan titik {i+1}:
").split(" ")))
            control_points.append(point)
    else:
        n = int(input("Masukkan jumlah control points: "))
        for i in range(n):
                point = tuple(map(float, input(f"Masukkan titik {i+1}:
").split(" ")))
            control_points.append(point)
    iter = int(input("Masukkan jumlah iterasi: "))

    if(mode==1):
        fig, ax = plt.subplots(1,2)
        start1 = time.time()
        Bezier.BezierN(control_points, iter, mid_points, curve_points)
        end1 = time.time()
        duration1 = end1-start1
        print(f'Divide and Conquer: {duration1}')
        mid_points=[]
        curve_points=[]
        start2 = time.time()
        Bezier.Bezier(control_points, iter, curve_points, mid_points)
        end2 = time.time()
        duration2 = end2-start2
        print(f'Bruteforce: {duration2}')
        if(duration2>duration1):
            print("Algoritma Divide and Conquer Lebih cepat!")
        elif(duration1>duration2):
            print("Algoritma Bruteforce Lebih cepat!")
        else:
            print("Waktu eksekusi sama!")
        print(f'Selisih waktu eksekusi: {abs(duration2-duration1)}')



        for i in range(1,iter+1):
```

```
            mid_points=[]
            curve_points=[]
            Bezier.BezierN(control_points, i, mid_points, curve_points)
            Bezier.show(control_points,i, ax[0], curve_points)
            ax[0].set_title('Divide and Conquer')

            mid_points=[]
            curve_points=[]
            Bezier.Bezier(control_points, i, curve_points, mid_points)
            Bezier.show(control_points,i, ax[1], curve_points)
            ax[1].set_title('Bruteforce')
            plt.pause(1)
    else:
        fig, ax = plt.subplots(1)
        for i in range(1,iter+1):
            mid_points=[]
            curve_points=[]
            Bezier.BezierN(control_points, i, mid_points, curve_points)
            Bezier.show(control_points,i, ax, curve_points)
            ax.set_title('Divide and Conquer')
            plt.pause(1)
    plt.show()


if __name__ == '__main__':
    main()
```

BAB IV
TEST CASE

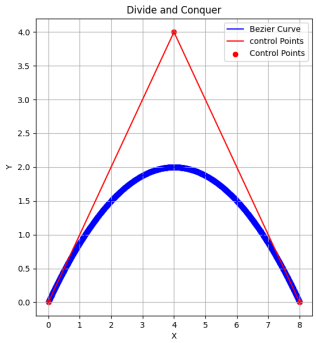
4.1 Test Case 1

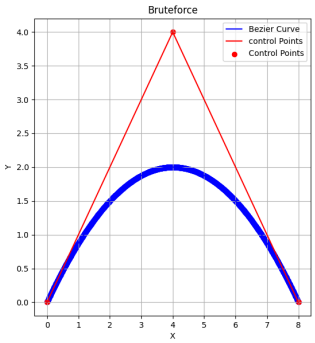





```
Selamat datang di program Bezier curve!
1. Kurva Bezier 3 titik
2. Kurva Bezier n titik
Pilihan: 1
Masukkan titik 1: 0 0
Masukkan titik 2: 4 4
Masukkan titik 3: 8 0
Masukkan jumlah iterasi: 5
Divide and Conquer: 0.0010361671447753906
Bruteforce: 0.0020360946655273438
Algoritma Divide and Conquer Lebih cepat!
Selisih waktu eksekusi: 0.0009999275207519531
```

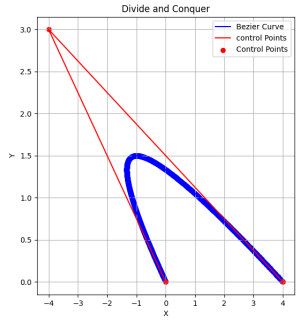
4.2 Test Case 2

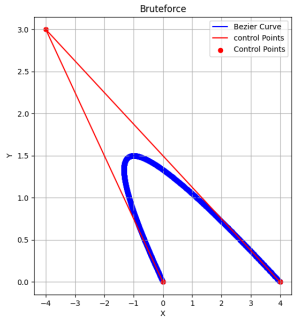





```
Selamat datang di program Bezier curve!
1. Kurva Bezier 3 titik
2. Kurva Bezier n titik
Pilihan: 1
Masukkan titik 1: 0 0
Masukkan titik 2: 4 4
Masukkan titik 3: 8 0
Masukkan jumlah iterasi: 10
Divide and Conquer: 0.0
Bruteforce: 0.002025127410888672
Algoritma Divide and Conquer Lebih cepat!
Selisih waktu eksekusi: 0.002025127410888672
```

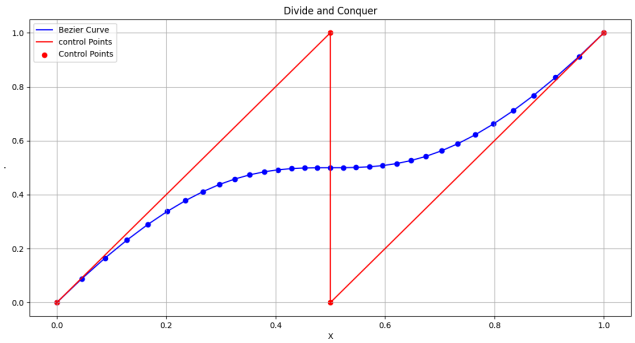
4.3 Test Case 3





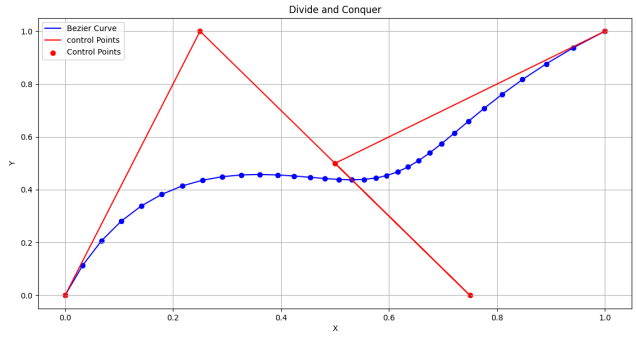

```
Selamat datang di program Bezier curve!
1. Kurva Bezier 3 titik
2. Kurva Bezier n titik
Pilihan: 1
Masukkan titik 1: 0 0
Masukkan titik 2: -4 3
Masukkan titik 3: 4 0
Masukkan jumlah iterasi: 10
Divide and Conquer: 0.011040925979614258
Bruteforce: 0.0020377635955810547
Algoritma Bruteforce Lebih cepat!
Selisih waktu eksekusi: 0.009003162384033203
```

4.4 Test Case 4



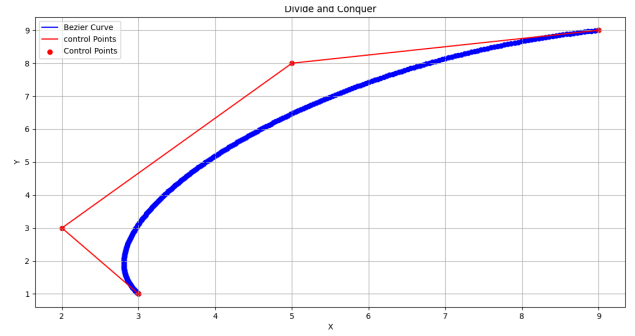

```
Selamat datang di program Bezier curve!
1. Kurva Bezier 3 titik
2. Kurva Bezier n titik
Pilihan: 2
Masukkan jumlah control points: 4
Masukkan titik 1: 0 0
Masukkan titik 2: 0.5 1
Masukkan titik 3: 0.5 0
Masukkan titik 4: 1 1
Masukkan jumlah iterasi: 5
```

4.5 Test Case 5



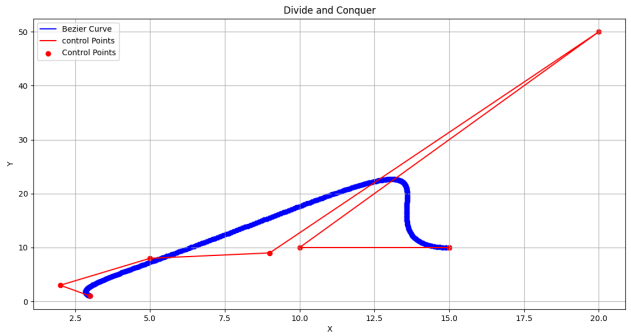

```
Selamat datang di program Bezier curve!
1. Kurva Bezier 3 titik
2. Kurva Bezier n titik
Pilihan: 2
Masukkan jumlah control points: 5
Masukkan titik 1: 0 0
Masukkan titik 2: 0.25 1
Masukkan titik 3: 0.75 0
Masukkan titik 4: 0.5 0.5
Masukkan titik 5: 1 1
Masukkan jumlah iterasi: 5
```

4. 6 Test Case 6



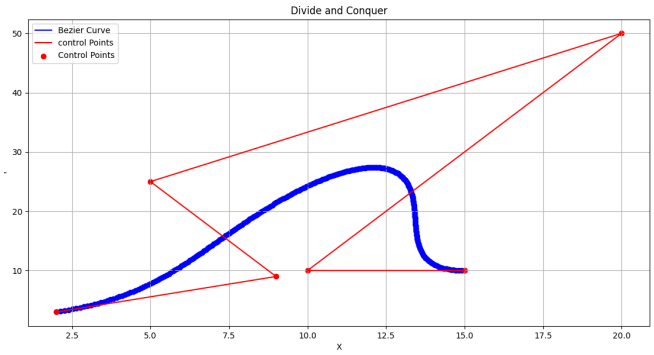

```
Selamat datang di program Bezier curve!
1. Kurva Bezier 3 titik
2. Kurva Bezier n titik
Pilihan: 2
Masukkan jumlah control points: 4
Masukkan titik 1: 3 1
Masukkan titik 2: 2 3
Masukkan titik 3: 5 8
Masukkan titik 4: 9 9
Masukkan jumlah iterasi: 8
```

4.7 Test Case 7




```
Selamat datang di program Bezier curve!
1. Kurva Bezier 3 titik
2. Kurva Bezier n titik
Pilihan: 2
Masukkan jumlah control points: 7
Masukkan titik 1: 3 1
Masukkan titik 2: 2 3
Masukkan titik 3: 5 8
Masukkan titik 4: 9 9
Masukkan titik 5: 20 50
Masukkan titik 6: 10 10
Masukkan titik 7: 15 10
Masukkan jumlah iterasi: 8
```

4.8 Test Case 8




```
Selamat datang di program Bezier curve!
1. Kurva Bezier 3 titik
2. Kurva Bezier n titik
Pilihan: 2
Masukkan jumlah control points: 6
Masukkan titik 1: 2 3
Masukkan titik 2: 9 9
Masukkan titik 3: 5 25
Masukkan titik 4: 20 50
Masukkan titik 5: 10 10
Masukkan titik 6: 15 10
Masukkan jumlah iterasi: 8
```

BAB VI
LAMPIRAN

Poin	Ya	Tidak
1. Program berhasil dijalankan	✅	
2. Program dapat menghasilkan visualisasi kurva Bezier	✅	
3. Solusi yang diberikan program optimal	✅	
4. [Bonus] Program dapat membuat kurva untuk n titik kontrol	✅	
5. **[Bonus]** Program dapat melakukan proses visualisasi kurva	✅	