LAPORAN TUGAS KECIL 2

IF2211 - STRATEGI ALGORITMA

“Membangun Kurva Bézier dengan Algoritma Titik Tengah berbasis Divide and Conquer”

Dosen:

Ir. Rila Mandala, M.Eng., Ph.D.

Monterico Adrian, S.T., M.T.

Kelompok 84:

M. Zaidan Sa’dun R. (13522146)

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER 2 TAHUN 2023/2024

DAFTAR ISI

BAB I Algoritma *Brute Force* dan *Divide and Conquer*

Pada pembuatan kurva Bézier, algoritma brute force dan algoritma divide and conquer (pembagian dan penaklukan) adalah dua pendekatan yang berbeda. Mari kita analisis dan bandingkan keduanya

1.1 Algoritma Brute Force

Algoritma brute force adalah pendekatan langsung di mana semua kemungkinan solusi diuji secara langsung. Untuk membuat kurva Bézier dengan metode brute force, kita akan mencoba setiap kombinasi titik kontrol untuk menentukan titik pada kurva. Langkah-langkah umum algoritma brute force untuk membuat kurva Bézier adalah sebagai berikut:

1. Ambil setiap kombinasi titik kontrol yang memungkinkan.
2. Hitung titik-titik pada kurva Bézier menggunakan persamaan Bézier.
3. Bandingkan hasilnya dan ambil yang terbaik berdasarkan kriteria tertentu (misalnya, jarak antara titik kurva dan titik target).

Keuntungan dari algoritma brute force adalah kejelasan dan kesederhanaannya. Namun, kompleksitas waktu algoritma ini meningkat secara eksponensial dengan jumlah titik kontrol. Oleh karena itu, kinerjanya mungkin tidak efisien untuk jumlah titik kontrol yang besar.

1.2 Algoritma Divide and Conquer

Algoritma divide and conquer memecah masalah menjadi submasalah yang lebih kecil, menyelesaikan submasalah-submasalah tersebut, dan kemudian menggabungkan solusi-solusi submasalah tersebut untuk mendapatkan solusi akhir. Untuk membuat kurva Bézier dengan algoritma divide and conquer, kita dapat membagi titik kontrol menjadi dua set dan membuat dua kurva Bézier terpisah, kemudian menggabungkan kurva-kurva tersebut untuk mendapatkan kurva Bézier akhir.

Langkah-langkah umum algoritma divide and conquer untuk membuat kurva Bézier adalah sebagai berikut:

1. Bagi set titik kontrol menjadi dua bagian.
2. Buat kurva Bézier untuk setiap bagian.
3. Gabungkan dua kurva Bézier tersebut untuk mendapatkan kurva Bézier akhir.

Keuntungan dari algoritma divide and conquer adalah kompleksitas waktu yang lebih baik dibandingkan dengan brute force untuk jumlah titik kontrol yang besar. Namun, algoritma ini memerlukan pemahaman yang lebih dalam tentang struktur masalah dan pengelompokan yang tepat dari titik kontrol.

1.3 Perbandingan

1. Kompleksitas Waktu

Brute force memiliki kompleksitas waktu eksponensial, sementara algoritma divide and conquer cenderung memiliki kompleksitas waktu yang lebih baik, terutama untuk jumlah titik kontrol yang besar.

2. Kesulitan Implementasi

Brute force relatif lebih mudah diimplementasikan karena sifatnya yang langsung, sementara algoritma divide and conquer memerlukan pemahaman yang lebih dalam tentang masalah dan struktur.

3. Efisiensi

Algoritma divide and conquer umumnya lebih efisien daripada brute force, terutama untuk ukuran input yang besar.

Dalam praktiknya, algoritma divide and conquer seringkali dipilih karena kinerjanya yang lebih baik. Namun, penting untuk mempertimbangkan kebutuhan spesifik dan kompleksitas masalah ketika memilih antara kedua pendekatan ini.

BAB II Source Program

Pada program kurva bezier ini saya menggunakan bahasa python, berikut code nya

File BruteForce.py (berisi algoritma brute force kurva bezier)

```
import matplotlib.pyplot as plt
import numpy as np
import time
from matplotlib.animation import FuncAnimation

class BezierCurve:
    def __init__(self, num_points):
        self.num_points = num_points
        self.user_contpt = self.get_control_points()
        self.time = 0


    def get_control_points(self):
        points = []
        for i in range(self.num_points):
            x, y = map(float, input(f"Point {i+1} (x,y): ").split(','))
            points.append([x, y])
        return np.array(points)

    def calculate_bezier_point(self, t, koordinat):
        start = time.time()
        n = len(koordinat) - 1
        point = np.zeros(2)
        for i in range(n+1):
            # Calculate binomial coefficient
            if i < 0 or i > n:
                binom_coef = 0
```

```
            elif i == 0 or i == n:
                binom_coef = 1
            else:
                k = min(i, n - i)
                binom_coef = 1
                for j in range(k):
                    binom_coef = binom_coef * (n - j) // (j + 1)
            point += binom_coef * (t ** i) * ((1 - t) ** (n - i)) * koordinat[i]
        self.time += time.time() - start
        return point

    def gettime(self):
        return self.time

    def animate(self, i, it):
        plt.cla()

        curve_points_user = []

        for j in range(i + 2):
            t = j / (i + 1)
            curve_points_user.append(self.calculate_bezier_point(t, self.user_contpt))

        curve_points_user = np.array(curve_points_user)
        plt.plot(curve_points_user[:, 0], curve_points_user[:, 1], 'b-', label="Bezier Curve
(Brute Force)")
        plt.scatter(curve_points_user[:, 0], curve_points_user[:, 1], color='blue')
        plt.plot(self.user_contpt[:, 0], self.user_contpt[:, 1],'ro-', label="Control Points")

        # Check if all iterations have been completed
        if i < it:
            plt.title("Bezier Curve with Brute Force Method (Process)")
        else:
            plt.title("Bezier Curve with Brute Force Method (Completed)")

        plt.xlabel('x')
        plt.ylabel('y')
        plt.legend()
        plt.grid(True)

def main():
    num_points = int(input("Masukkan n jumlah titik: "))
    bezier = BezierCurve(num_points)
    it = ((int(input("Masukkan jumlah iterasi: "))) * (num_points-1))
    anim = FuncAnimation(plt.gcf(), lambda i: bezier.animate(i, it), frames=it+2,
repeat=False)
    plt.show()
    print(f"Time: {bezier.gettime()}s")
```

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

File DNC.py (berisi algoritma divide n conquer ku)

```
import matplotlib.pyplot as plt
import numpy as np
import time
from matplotlib.animation import FuncAnimation

class BezierCurve:
    def __init__(self, ctrl_points, iterations):
        self.ctrl_points = ctrl_points
        self.iterations = iterations
        self.bezier_points = []
        self.time = 0
        self.generate_bezier()

    def midpoint(self, point1, point2):
        return ((point1[0] + point2[0]) / 2, (point1[1] + point2[1]) / 2)

    def interpolate_bezier_points(self, ctrl1, ctrl2, ctrl3, current_iteration):
        if current_iteration < self.iterations:
            mid1 = self.midpoint(ctrl1, ctrl2)
            mid2 = self.midpoint(ctrl2, ctrl3)
            mid3 = self.midpoint(mid1, mid2)

            current_iteration += 1
            self.interpolate_bezier_points(ctrl1, mid1, mid3, current_iteration)
            self.bezier_points.append(mid3)
            self.interpolate_bezier_points(mid3, mid2, ctrl3, current_iteration)

    def generate_bezier(self):
        start = time.time()
        ctrl1, ctrl2, ctrl3 = self.ctrl_points
        self.bezier_points = [ctrl1]
        self.interpolate_bezier_points(ctrl1, ctrl2, ctrl3, 0)
        self.bezier_points.append(ctrl3)
        self.time += time.time() - start

    def gettime(self):
        return self.time

def get_input():
    print("Masukkan 3 titik point:")
    ctrl_points = [tuple(map(float, input(f"Point {i+1} (x,y): ").split(','))) for i in range(3)]
```

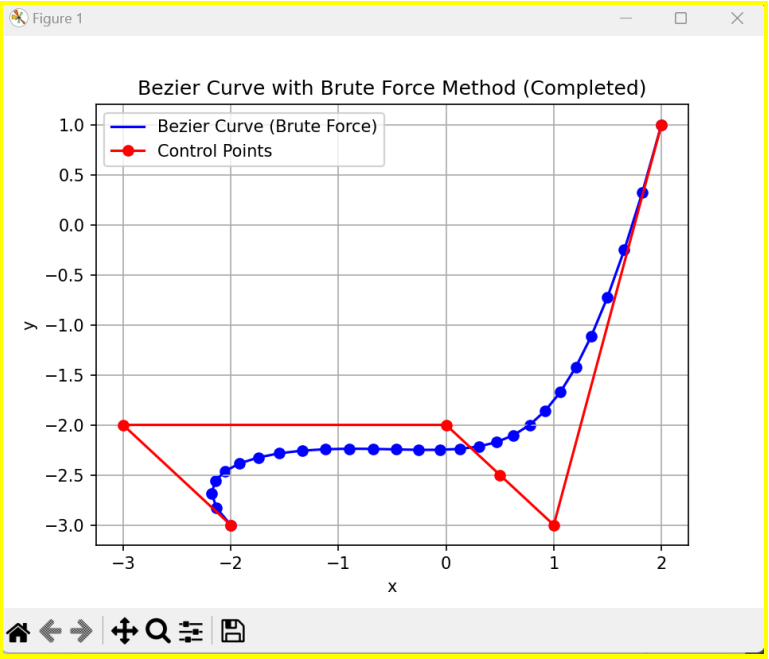
```
    iterations = int(input("Masukkan jumlah iterasi: "))
    return ctrl_points, iterations

def animate(i):
    plt.cla()
    bezier_curve = BezierCurve(ctrl_points, i)
    points = np.array(bezier_curve.bezier_points)
    ctrl_points_array = np.array(ctrl_points)
    plt.plot(points[:, 0], points[:, 1], 'b-', label='Bezier Curve')
    plt.scatter(points[:, 0], points[:, 1], color='blue')
    plt.plot(ctrl_points_array[:, 0], ctrl_points_array[:, 1], 'ro-', label='Control Points')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.legend()
    plt.title('Bezier Curve with Divide and Conquer Method ke-{0}'.format(i))
    plt.grid(True)

def main():
    global ctrl_points
    ctrl_points, iterations = get_input()

    anim = FuncAnimation(plt.gcf(), animate, frames=iterations+1, repeat=False)


    plt.show()
    print(f"Time: {BezierCurve(ctrl_points, iterations).gettime()} seconds")

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

BAB III Input & Output

INPUT BRUTE FORCE

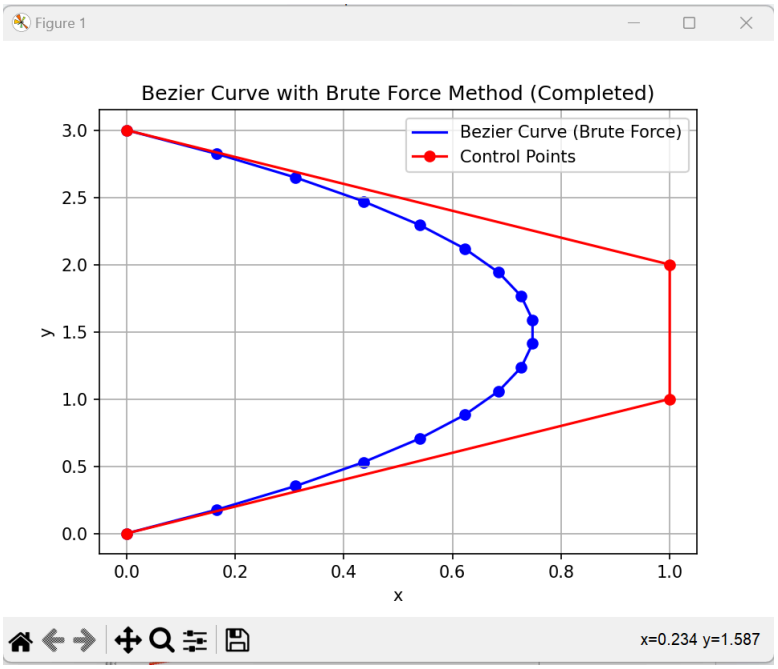
INPUT 1 **OUTPUT 1**

```
Masukkan n jumlah titik: 6
Point 1 (x,y): -2,-3
Point 2 (x,y): -3,-2
Point 3 (x,y): 0,-2
Point 4 (x,y): 0.5,-2.5
Point 5 (x,y): 1,-3
Point 6 (x,y): 2,1
Masukkan jumlah iterasi: 5
Time: 0.01387023925781255
```



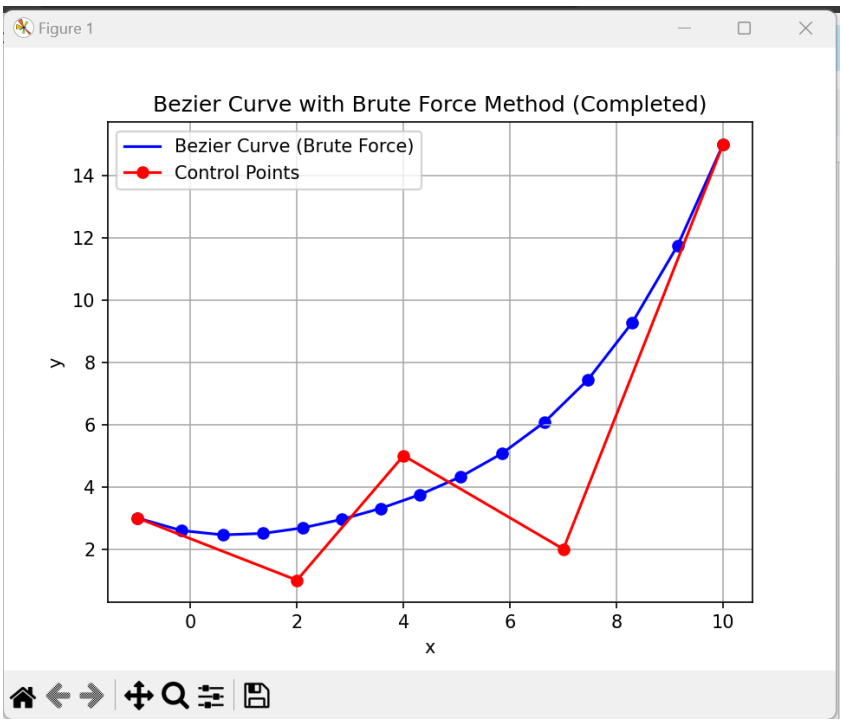
INPUT 2 **OUTPUT 2**

```
Masukkan n jumlah titik: 4
Point 1 (x,y): 0,0
Point 2 (x,y): 1,1
Point 3 (x,y): 1,2
Point 4 (x,y): 0,3
Masukkan jumlah iterasi: 5
Time: 0.0070548057556152345s
```



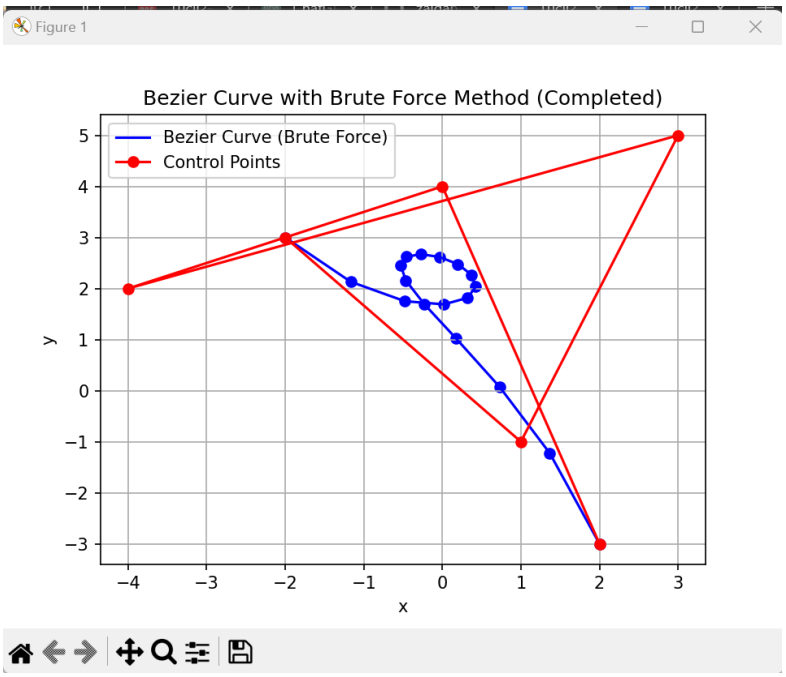
INPUT 3 **OUTPUT 3**

```
Masukkan n jumlah titik: 5
Point 1 (x,y): -1,3
Point 2 (x,y): 2,1
Point 3 (x,y): 4,5
Point 4 (x,y): 7,2
Point 5 (x,y): 10,15
Masukkan jumlah iterasi: 3
Time: 0.0020444393157958984s
```



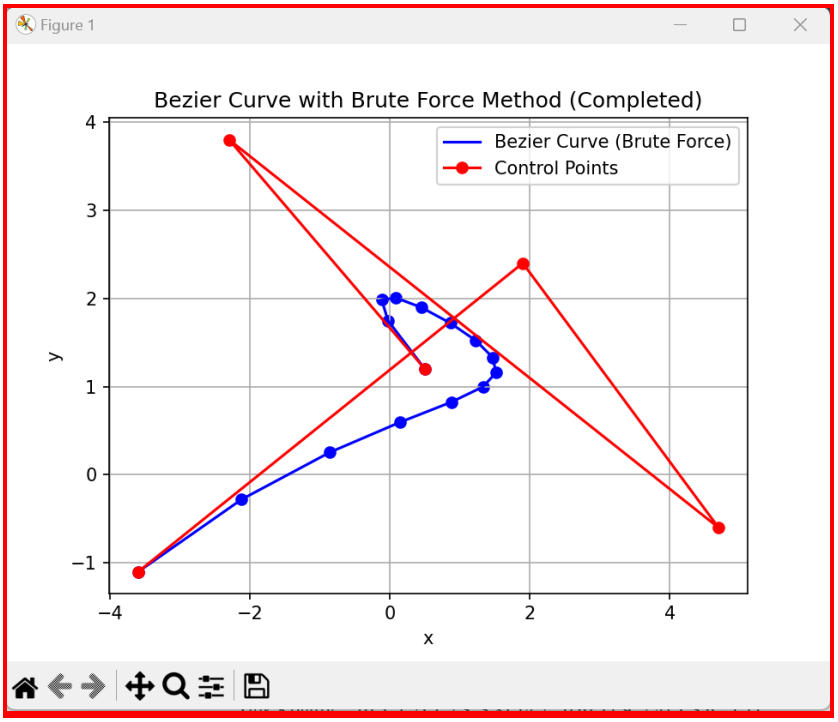
INPUT 4 **OUTPUT 4**

```
Masukkan n jumlah titik: 6
Point 1 (x,y): -2,3
Point 2 (x,y): 1,-1
Point 3 (x,y): 3,5
Point 4 (x,y): -4,2
Point 5 (x,y): 0,4
Point 6 (x,y): 2,-3
Masukkan jumlah iterasi: 3
Time: 0.008055925369262695s
```



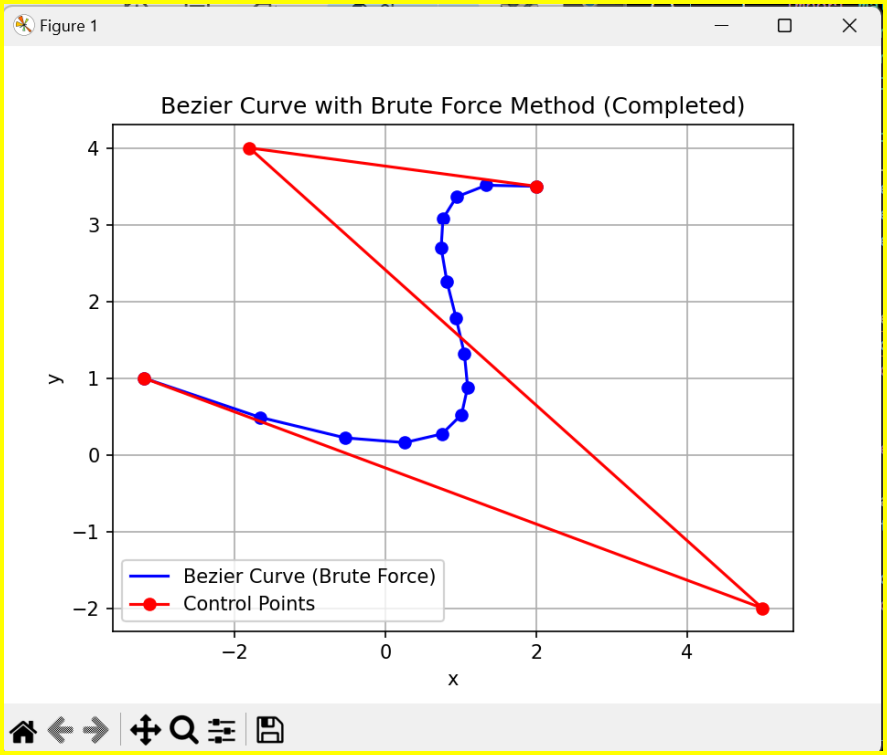
INPUT 5 **OUTPUT 5**

```
Masukkan n jumlah titik: 5
Point 1 (x,y): 0.5,1.2
Point 2 (x,y): -2.3,3.8
Point 3 (x,y): 4.7,-0.6
Point 4 (x,y): 1.9,2.4
Point 5 (x,y): -3.6,-1.1
Masukkan jumlah iterasi: 3
Time: 0.0030295848846435547s
```



INPUT 6 **OUTPUT 6**

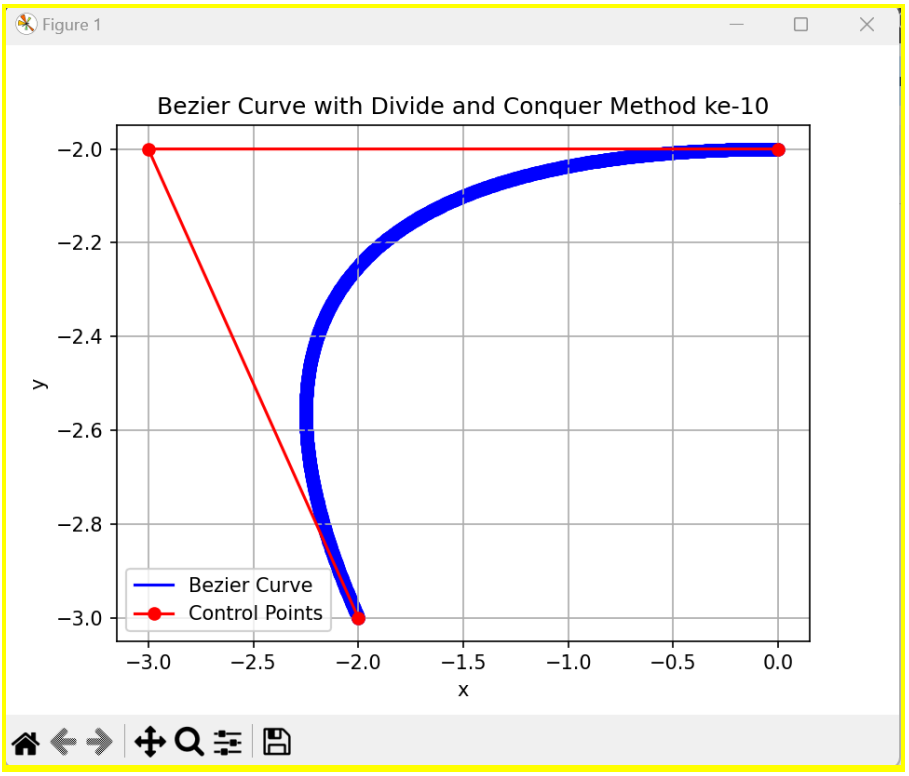
```
Masukkan n jumlah titik: 4
Point 1 (x,y): 2,3.5
Point 2 (x,y): -1.8,4
Point 3 (x,y): 5,-2
Point 4 (x,y): -3.2,1
Masukkan jumlah iterasi: 4
Time: 0.0019929409027099961s
```



INPUT DIVIDE AND CONQUER

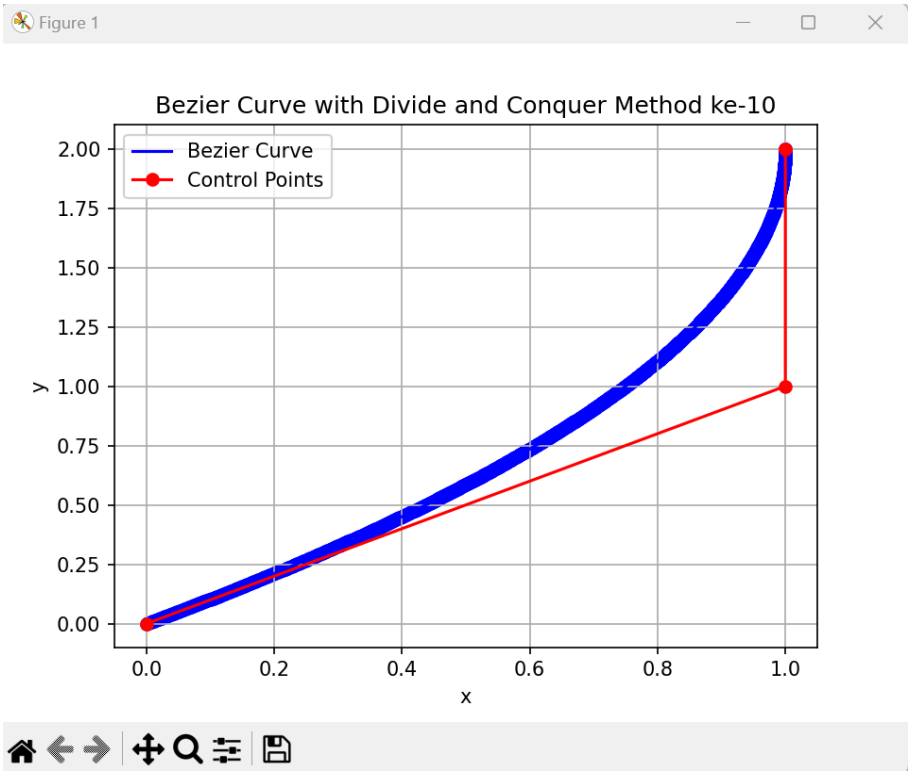
INPUT 1 **OUTPUT 1**

```
Masukkan 3 titik point:
Point 1 (x,y): -2,-3
Point 2 (x,y): -3,-2
Point 3 (x,y): 0,-2
Masukkan jumlah iterasi: 10
Time: 0.0022096633911132812 seconds
```



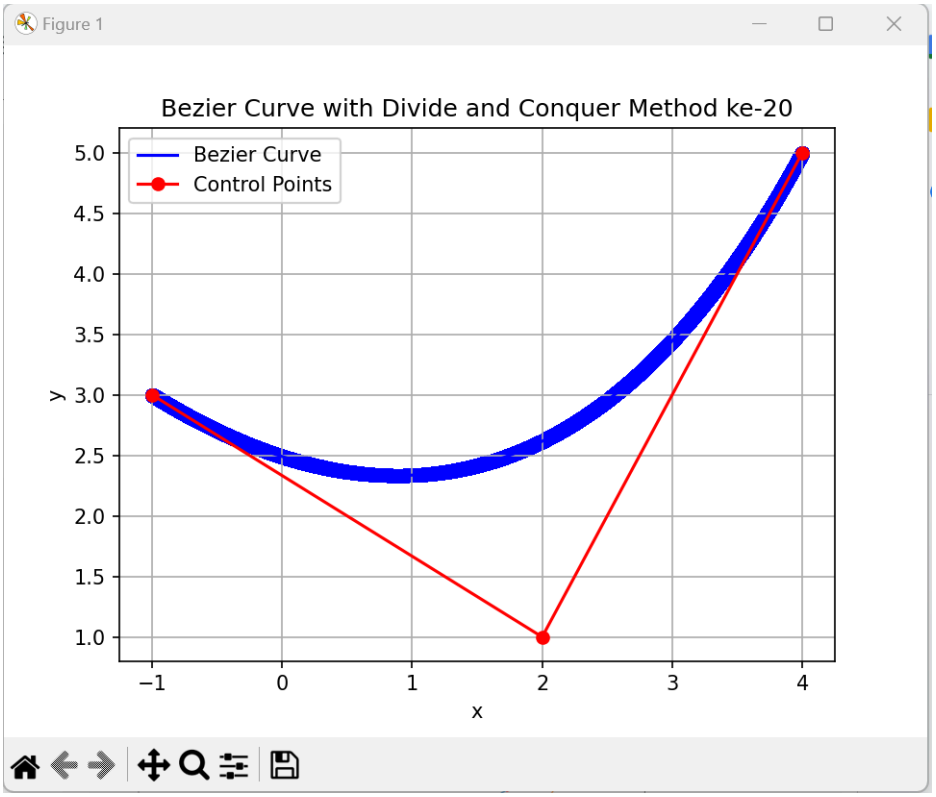
INPUT 2 **OUTPUT 2**

```
Masukkan 3 titik point:
Point 1 (x,y): 0,0
Point 2 (x,y): 1,1
Point 3 (x,y): 1,2
Masukkan jumlah iterasi: 10
Time: 0.00300312042236328l2 seconds
```



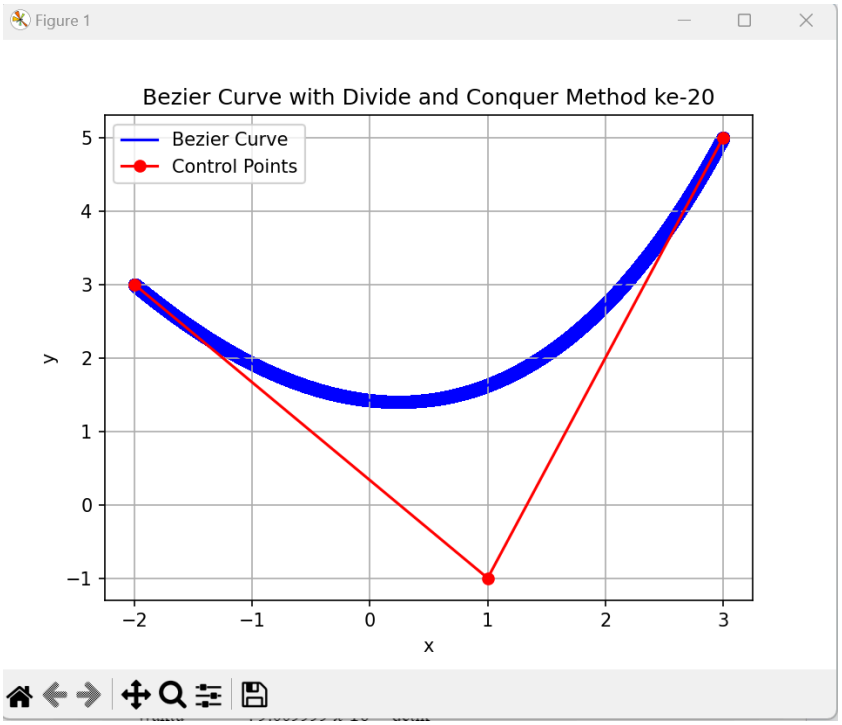
INPUT 3 **OUTPUT 3**

```
Masukkan 3 titik point:
Point 1 (x,y): -1,3
Point 2 (x,y): 2,1
Point 3 (x,y): 4,5
Masukkan jumlah iterasi: 20
Time: 1.2200653553009033 seconds
```



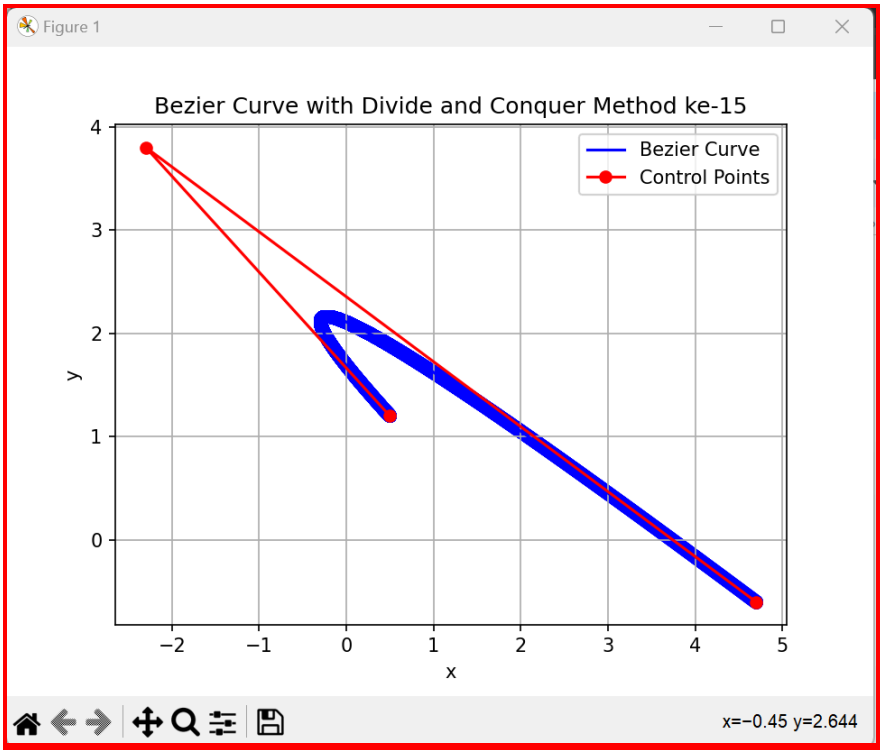
INPUT 4 **OUTPUT 4**

```
Masukkan 3 titik point:
Point 1 (x,y): -2,3
Point 2 (x,y): 1,-1
Point 3 (x,y): 3,5
Masukkan jumlah iterasi: 20
Time: 1.3234772682189941 seconds
```



INPUT 5 **OUTPUT 5**

```
Masukkan 3 titik point:
Point 1 (x,y): 0.5,1.2
Point 2 (x,y): -2.3,3.8
Point 3 (x,y): 4.7,-0.6
Masukkan jumlah iterasi: 15
Time: 0.051201581954956055 seconds
```



INPUT 6 **OUTPUT 6**

```
Masukkan 3 titik point:
Point 1 (x,y): 2,3.5
Point 2 (x,y): -1.8,4
Point 3 (x,y): 5,-2
Masukkan jumlah iterasi: 18
Time: 0.28989720344543457 seconds
```

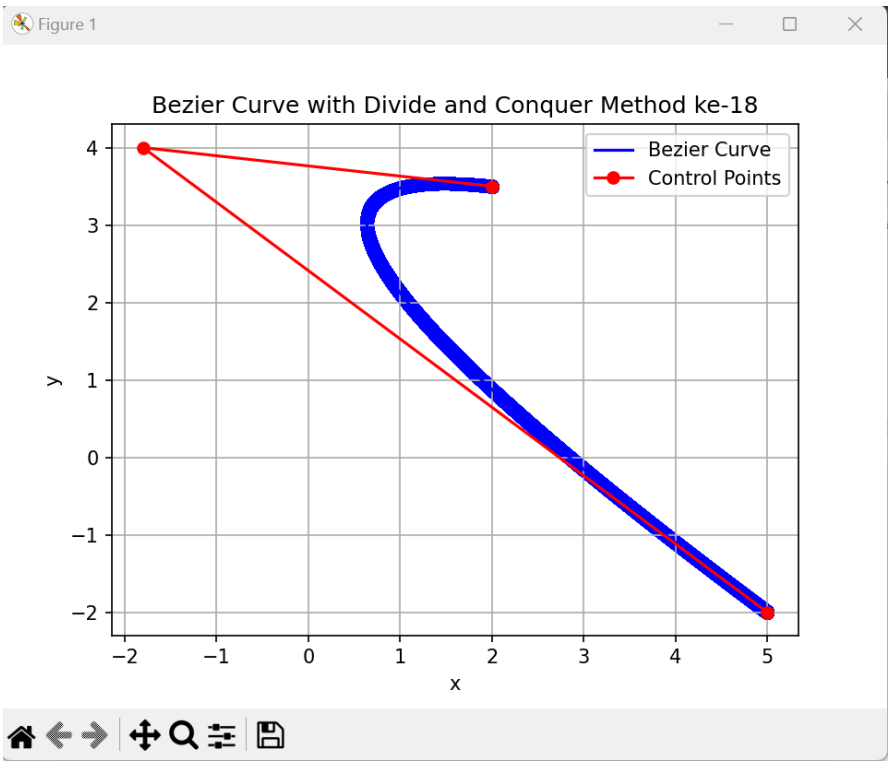


Figure 1
Bezier Curve with Divide and Conquer Method ke-18
Bezier Curve
Control Points
y
x
4
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4
5

BAB IV Analisis Hasil

Waktu antara kedua algoritma ini tergantung pada ukuran input (jumlah titik kontrol) dan efisiensi implementasi masing-masing algoritma. Namun, secara umum, algoritma divide and conquer cenderung lebih cepat daripada algoritma brute force untuk jumlah titik kontrol yang besar.

Alasan utama adalah kompleksitas waktu dari masing-masing algoritma:

1. Algoritma Brute Force

   Kompleksitas waktu algoritma brute force untuk membuat kurva Bézier adalah O(2^n), dimana n adalah jumlah titik kontrol. Ini berarti waktu eksekusi algoritma meningkat secara eksponensial seiring dengan penambahan titik kontrol.

2. Algoritma Divide and Conquer

   Kompleksitas waktu algoritma divide and conquer untuk membuat kurva Bézier cenderung lebih rendah. Meskipun kompleksitas pasti tergantung pada implementasi spesifik, algoritma ini biasanya memiliki kompleksitas waktu O(n log n) atau O(n^2), tergantung pada bagaimana algoritma digariskan.

Jadi, untuk jumlah titik kontrol yang besar, algoritma divide and conquer biasanya akan lebih cepat dibandingkan dengan algoritma brute force. Namun, untuk jumlah titik kontrol yang sangat kecil, perbedaan waktu mungkin tidak signifikan, dan algoritma brute force mungkin terasa lebih cepat karena kesederhanaannya.

BAB V Implementasi Bonus

5.1 Bonus 1

Saya mengimplementasikan bonus pertama generalisasi algoritma untuk menerima n buah titik kontrol. Generalisasi algoritma kami lakukan dengan mengganti rumus kurva Bezier kuadratik dengan rumus general kurva Bezier, yaitu:

untuk algoritma brute force saja

5.2 Bonus 2

Saya mengimplementasikan bonus kedua yaitu visualisasi proses pembentukan kurva menggunakan library matplotlib. Proses visualisasi pembentukan dibagi menjadi beberapa tahapan:

1. Digambarkan titik kontrol dan garis penghubungnya sesuai urutan masukan titik kontrol.
2. Kurva digambarkan secara bertahap dengan increment 1 titik Bezier.

PRANALA

Link GITHUB: zaidanav/Tucil2_13522146 (github.com)

Tabel kebenaran:

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