LAPORAN TUGAS KECIL 2
IF2211

STRATEGI ALGORITMA

MEMBANGUN KURVA BÉZIER DENGAN ALGORITMA TITIK TENGAH BERBASIS *DIVIDE AND CONQUER*

Disusun Oleh :
Yudi kurniawan (10023634)
Habibi Galang Triyanda (10023457)

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

2024

Daftar Isi

BAB I

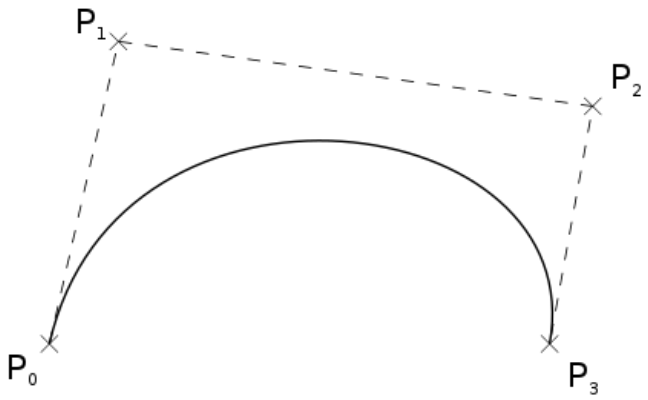
Deskripsi Masalah

Kurva Bézier adalah elemen penting dalam grafis komputer, sering digunakan dalam desain grafis animasi dan manufaktur. Program ini bertujuan untuk membangun kurva Bézier kuadratik menggunakan algoritma titik tengah berbasis divide and conquer. Dua metode akan dibandingkan: brute force dan divide and conquer, untuk mengilustrasikan keefektifan pendekatan berbasis divide and conquer dalam pembentukan kurva yang halus dan presisi. Input program melibatkan tiga pasangan titik kontrol dan jumlah iterasi, menghasilkan visualisasi kurva Bézier dan waktu eksekusi kedua metode.

Kurva Bézier dibuat dengan menghubungkan titik-titik kontrol dengan kurva yang menggambarkan bentuk dan arah. Metode divide and conquer memecah masalah pembuatan kurva menjadi sub-proses yang lebih kecil, memungkinkan pembuatan kurva yang lebih efisien dan akurat. Program ini mengimplementasikan kedua metode untuk menunjukkan proses pembentukan kurva Bézier dan membandingkan performa mereka.

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.

(Paragraf di atas dikutip dari : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Tucil2-2024.pdf).



Gambar 1. Kurva Bézier Kubik

(Sumber: https://id.wikipedia.org/wiki/Kurva_B%C3%A9zier)

BAB II

Landasan Teori

Dalam implementasi algoritma pembentukan kurva Bézier menggunakan metode titik tengah berbasis divide and conquer, kita mengamati beberapa aspek yang relevan. Pertama-tama, kita menggunakan fungsi bezier_curve yang menerima tiga titik kontrol sebagai masukan dan parameter t untuk menentukan posisi pada kurva Bézier. Metode ini secara iteratif menghitung posisi titik pada kurva dengan mengaplikasikan rumus parametrik Bézier pada setiap iterasi. Selain itu, kita memperhitungkan titik tengah dari dua segmen kurva yang dihasilkan dalam algoritma divide and conquer, yang diwakili oleh titik Q_0dan Q_1, serta titik tengah mereka yang disebut B.

Dalam algoritma divide and conquer untuk membentuk kurva Bézier, rumus yang digunakan adalah rumus parametrik untuk titik pada kurva Bézier. Misalkan kita memiliki tiga titik kontrol P_0, P_1,P_2, dan parameter t berkisar antara 0 dan 1, maka rumus untuk menghitung titik pada kurva Bézier adalah sebagai berikut :

$$B(t) = (1-t)^2.P_0 + 2(1-t)t.P_1 + t^2.P_2$$

Dalam rumus ini, B(t) merupakan titik pada kurva Bézier pada parameter t tertentu. P_0, P_1, dan P_2 adalah titik kontrol yang diberikan oleh pengguna. Parameter t berkisar antara 0 dan 1, dan digunakan untuk menentukan posisi pada kurva. Dengan mengubah nilai parameter t dari 0 hingga 1, kita dapat menghasilkan sejumlah titik pada kurva Bézier.

Rumus ini digunakan pada setiap segmen kurva Bézier yang dibangun dalam algoritma divide and conquer. Dengan mengganti nilai parameter t, kita dapat menghitung posisi titik pada kurva pada berbagai titik pada kurva. Ini memungkinkan kita untuk membangun kurva Bézier dengan presisi yang tinggi menggunakan algoritma divide and conquer. Dari segi visualisasi, kita dapat melihat kurva Bézier yang dihasilkan oleh algoritma pada plot. Kurva ini dibangun berdasarkan titik kontrol yang diberikan oleh pengguna, serta garis dan titik spesifik pada kurva pada nilai t=0.25. Visualisasi ini memberikan pemahaman visual yang lebih baik tentang bentuk dan struktur dari kurva Bézier yang dihasilkan oleh algoritma.

Dengan mempertimbangkan analisis kinerja dan visualisasi hasil, kita dapat mengevaluasi keefektifan algoritma divide and conquer dalam menghasilkan kurva Bézier. Dalam kasus ini, kita dapat mengamati apakah algoritma dapat menghasilkan kurva dengan presisi yang diinginkan

dalam waktu yang wajar tergantung pada jumlah iterasi yang diberikan oleh pengguna.

Algoritma brute force merupakan metode sederhana namun kurang efisien dalam pembentukan kurva Bézier kuadratik. Pendekatan ini melibatkan pengujian semua kemungkinan nilai parameter t dalam rentang ($0 \leq t \leq 1$)secara berurutan untuk menghitung titik-titik pada kurva Bézier. Pertama-tama, kita mendefinisikan rentang nilai t dari 0 hingga 1. Rentang ini akan digunakan untuk menguji setiap kemungkinan nilai t yang mungkin. Kemudian, untuk setiap nilai t dalam rentang tersebut, kita menggunakan rumus matematis dari kurva Bézier untuk menghitung koordinat x dan y dari titik pada kurva. Rumus matematis untuk menghitung titik pada kurva Bézier kuadratik adalah:

$$B(t) = (1-t)^2.P_0 + 2(1-t)t.P_1 + t^2.P_2$$

di mana P0, P1, dan P2 adalah titik kontrol kurva Bézier. Proses ini dilakukan secara berulang untuk setiap nilai t dalam rentang yang ditentukan.

Setelah semua titik pada kurva Bézier dihitung, kita mengumpulkan semua titik tersebut untuk membentuk kurva Bézier secara keseluruhan. Terakhir, hasilnya dapat divisualisasikan untuk melihat bentuk kurva yang dihasilkan. Kelebihan dari algoritma brute force adalah sederhana dan mudah dimengerti. Namun, kelemahannya adalah kurang efisien dalam hal kinerja. Algoritma brute force mencoba semua kemungkinan solusi secara berurutan, yang dapat memakan waktu yang lama terutama jika jumlah titik yang dihitung besar.

Dalam konteks pembentukan kurva Bézier kuadratik, algoritma brute force cenderung menghasilkan kurva yang kurang halus dan presisi dibandingkan dengan algoritma lain yang lebih efisien. Oleh karena itu, meskipun algoritma brute force sederhana dan mudah dimengerti, namun untuk kasus pembentukan kurva Bézier kuadratik, pendekatan ini tidak efisien dan kurang disukai.

BAB III

Implementasi Program

Berikut adalah code dari program nya :

A. Algoritma Brute Force

Berikut adalah code dengan algoritma brute force :

```
import matplotlib.pyplot as plt

def quadratic_bezier(t, start_point, control_point, end_point):
    """
    Menghitung titik pada kurva Bezier kuadratik pada parameter t.
    """
    x = (1 - t)**2 * start_point[0] + 2 * (1 - t) * t * control_point[0] + t**2 * end_point[0]
    y = (1 - t)**2 * start_point[1] + 2 * (1 - t) * t * control_point[1] + t**2 * end_point[1]
    return x, y

def plot_quadratic_bezier(start_point, control_point, end_point, num_points=100):
    """
    Plot kurva Bezier kuadratik dengan pendekatan brute force.
    """
    x_values, y_values = [], []
    for i in range(num_points + 1):
        t = i / num_points
        x, y = quadratic_bezier(t, start_point, control_point, end_point)
        x_values.append(x)
        y_values.append(y)

    plt.figure(figsize=(8, 6))
    plt.plot(x_values, y_values, label='Kurva Bezier Kuadratik')
    plt.scatter([start_point[0], control_point[0], end_point[0]], [start_point[1],
control_point[1], end_point[1]], color='red', label='Titik Kontrol')
    plt.plot([start_point[0], control_point[0], end_point[0]], [start_point[1],
control_point[1], end_point[1]], linestyle='--', color='grey', alpha=0.5, label='Garis
Kontrol')

    plt.title('Kurva Bezier Kuadratik')
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.legend()
    plt.grid(True)
```

```
    plt.axis('equal')
    plt.show()

def input_point(label):
    x = float(input(f"Masukkan koordinat x untuk {label}: "))
    y = float(input(f"Masukkan koordinat y untuk {label}: "))
    return (x, y)

def main():
    print("Masukkan koordinat untuk titik-titik Bezier:")
    start_point = input_point("titik awal")
    control_point = input_point("titik kontrol")
    end_point = input_point("titik akhir")

    plot_quadratic_bezier(start_point, control_point, end_point)

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

B. Algoritma Divide and Conquer

Berikut adalah code dengan algoritma divide and conquer:

```
import numpy as np
import matplotlib.pyplot as plt

def bezier_kuadratik(t, p0, p1, p2):
    x = (1 - t)**2 * p0[0] + 2 * (1 - t) * t * p1[0] + t**2 * p2[0]
    y = (1 - t)**2 * p0[1] + 2 * (1 - t) * t * p1[1] + t**2 * p2[1]
    return x, y

def plot_bezier_kuadratik(p0, p1, p2, iterasi=5, num_points=100, t=0.5):
    # Hitung koordinat Q0 dan Q1
    q0 = ((p0[0] + p1[0]) / 2, (p0[1] + p1[1]) / 2)
    q1 = ((p1[0] + p2[0]) / 2, (p1[1] + p2[1]) / 2)

    # Hitung koordinat titik pada kurva Bezier
    t_values = np.linspace(0, 1, num_points)
    curve_points = [bezier_kuadratik(t, p0, p1, p2) for t in t_values]
    x_values, y_values = zip(*curve_points)

    # Hitung koordinat T0, R0, dan T1
```

```
    t0 = ((p0[0] + q0[0]) / 2, (p0[1] + q0[1]) / 2)
    r0 = ((q0[0] + q1[0]) / 2, (q0[1] + q1[1]) / 2)
    t1 = ((q1[0] + p2[0]) / 2, (q1[1] + p2[1]) / 2)

    # Hitung titik-titik aproksimasi garis baru
    line_points = [p0, t0, r0, t1, p2]

    # Plot kurva Bezier, titik kontrol, dan garis baru
    plt.plot(x_values, y_values, label='Kurva Bezier Kuadratik')
    plt.scatter([p0[0], p1[0], p2[0], q0[0], q1[0], t0[0], r0[0], t1[0]],
                [p0[1], p1[1], p2[1], q0[1], q1[1], t0[1], r0[1], t1[1]],
                color='red', label='Titik Kontrol, Q0/Q1, T0/R0/T1')
    plt.plot([point[0] for point in line_points], [point[1] for point in line_points],
linestyle='--', color='blue', label='Garis P0-T0-R0-T1-P2 (Aproksimasi)')

    # Plot kurva Bezier, titik kontrol, dan garis baru Q0-Q1
    plt.plot([q0[0], q1[0]], [q0[1], q1[1]], linestyle='--', color='green', label='Garis Q0-
Q1')

    plt.title('Kurva Bezier Kuadratik dengan Titik Kontrol Tambahan')
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.legend()
    plt.grid(True)
    plt.axis('equal')
    plt.show()

def input_titik(label):
    x = float(input(f"Masukkan koordinat x untuk {label}: "))
    y = float(input(f"Masukkan koordinat y untuk {label}: "))
    return (x, y)

def input_parameter_t():
    t = float(input("Masukkan nilai parameter t (0 <= t <= 1): "))
    if 0 <= t <= 1:
        return t
    else:
        print("Nilai parameter t harus berada dalam rentang 0 hingga 1.")
        return input_parameter_t()

def input_iterasi():
    iterasi = int(input("Masukkan jumlah iterasi: "))
    if iterasi >= 0:
        return iterasi
    else:
```

```
        print("Jumlah iterasi harus merupakan bilangan bulat non-negatif.")
        return input_iterasi()

def utama():
    t = input_parameter_t()
    p0 = input_titik("titik awal")
    p1 = input_titik("titik kontrol")
    p2 = input_titik("titik akhir")
    iterasi = input_iterasi()

    plot_bezier_kuadratik(p0, p1, p2, iterasi=iterasi, num_points=100, t=t)

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

BAB IV

Eksperimen dan Hasil

A. Input dan Output

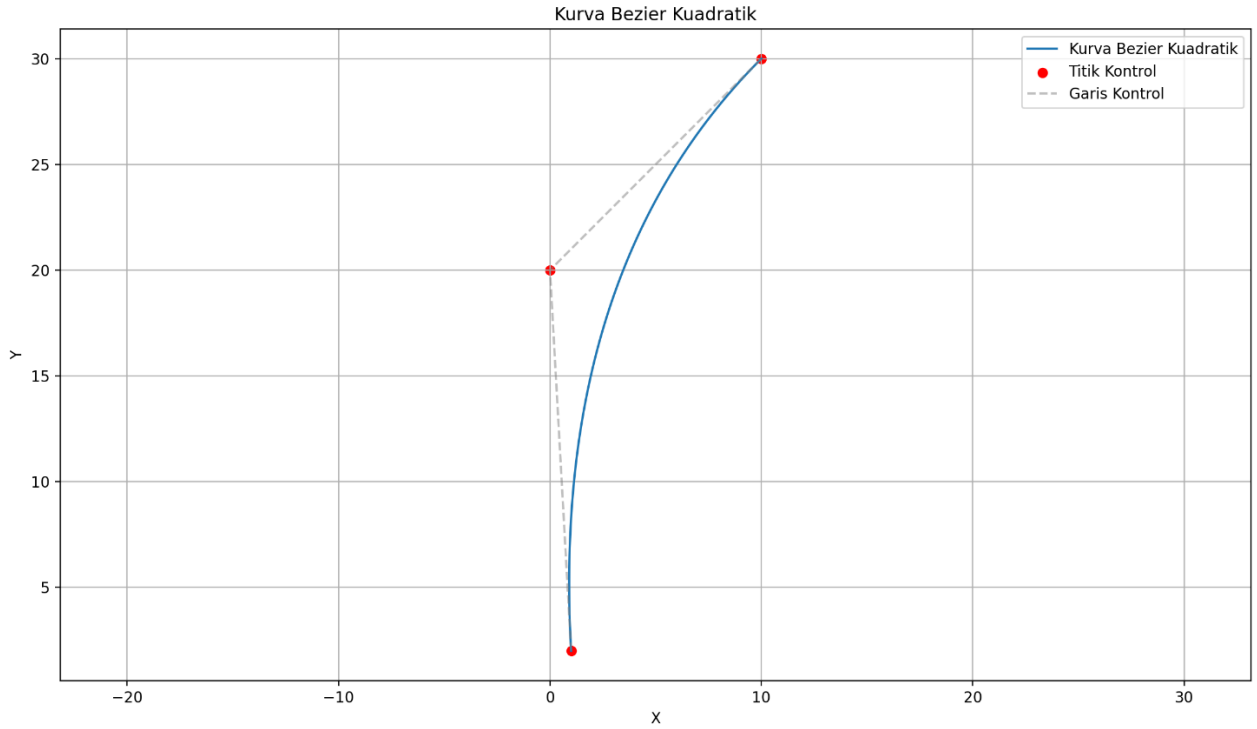
Berikut ini beberapa kali hasil percobaan dengan Algoritma Brute Force dan Divide and Coquer:

a) Algoritma Brute Force

Input 1 :

```
PS C:\Users\VIVOBOOK> & "C:/Program Files/Python311/python.exe" "d:/STIMA 2024/tucil2-IF2211-AlgoritmaDivide&Conquer/src/BruteForceImplementation.py"
Masukkan koordinat untuk titik-titik Bezier:
Masukkan koordinat x untuk titik awal: 10
Masukkan koordinat y untuk titik awal: 30
Masukkan koordinat x untuk titik kontrol: 0
Masukkan koordinat y untuk titik kontrol: 20
Masukkan koordinat x untuk titik akhir: 1
Masukkan koordinat y untuk titik akhir: 2
```

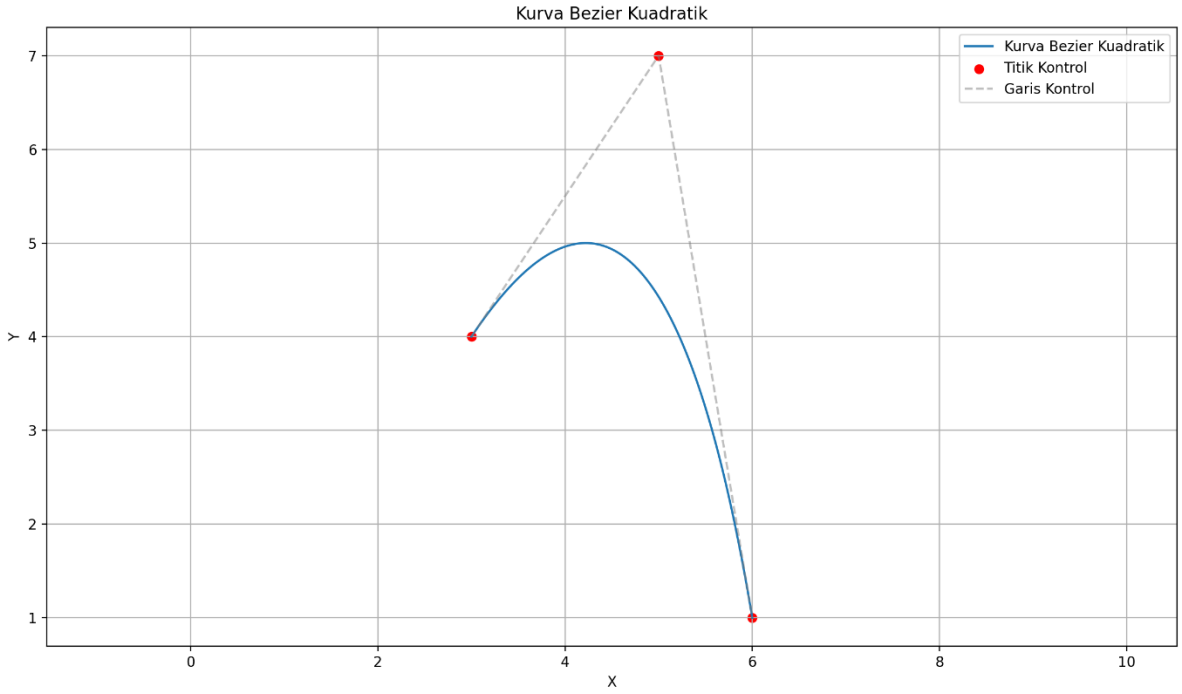
Output 1 :



Input 2 :

```
PS C:\Users\VIVOBOOK> & "C:/Program Files/Python311/python.exe" "d:/STIMA 2024/tucil2-IF2211-AlgoritmaDivide&Conquer/src/BruteForceImplementation.py"
Masukkan koordinat untuk titik-titik Bezier:
Masukkan koordinat x untuk titik awal: 3
Masukkan koordinat y untuk titik awal: 4
Masukkan koordinat x untuk titik kontrol: 5
Masukkan koordinat y untuk titik kontrol: 7
Masukkan koordinat x untuk titik akhir: 6
Masukkan koordinat y untuk titik akhir: 1
```

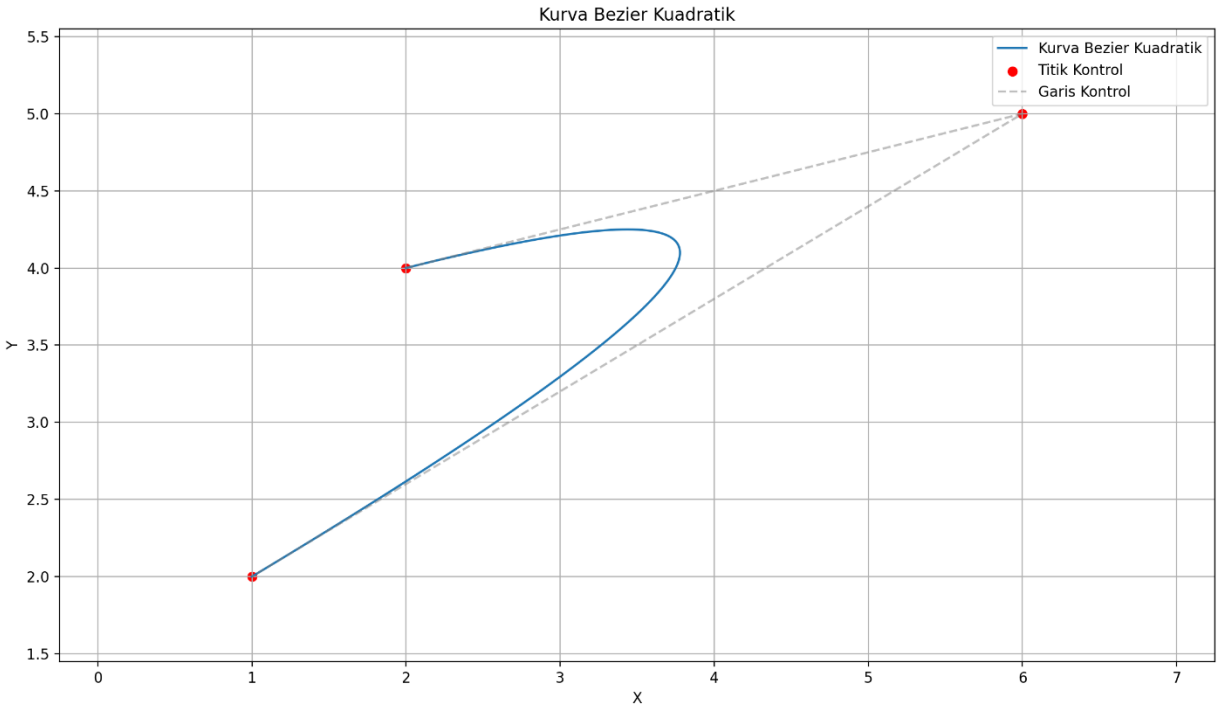
Output 2 :



Input 3 :

```
PS C:\Users\VIVOBOOK> & "C:/Program Files/Python311/python.exe" "d:/STIMA 2024/tucil2-IF2211-AlgoritmaDivide&Conquer/src/BruteForceImplementation.py"
Masukkan koordinat untuk titik-titik Bezier:
Masukkan koordinat x untuk titik awal: 2
Masukkan koordinat y untuk titik awal: 4
Masukkan koordinat x untuk titik kontrol: 6
Masukkan koordinat y untuk titik kontrol: 5
Masukkan koordinat x untuk titik akhir: 1
Masukkan koordinat y untuk titik akhir: 2
```

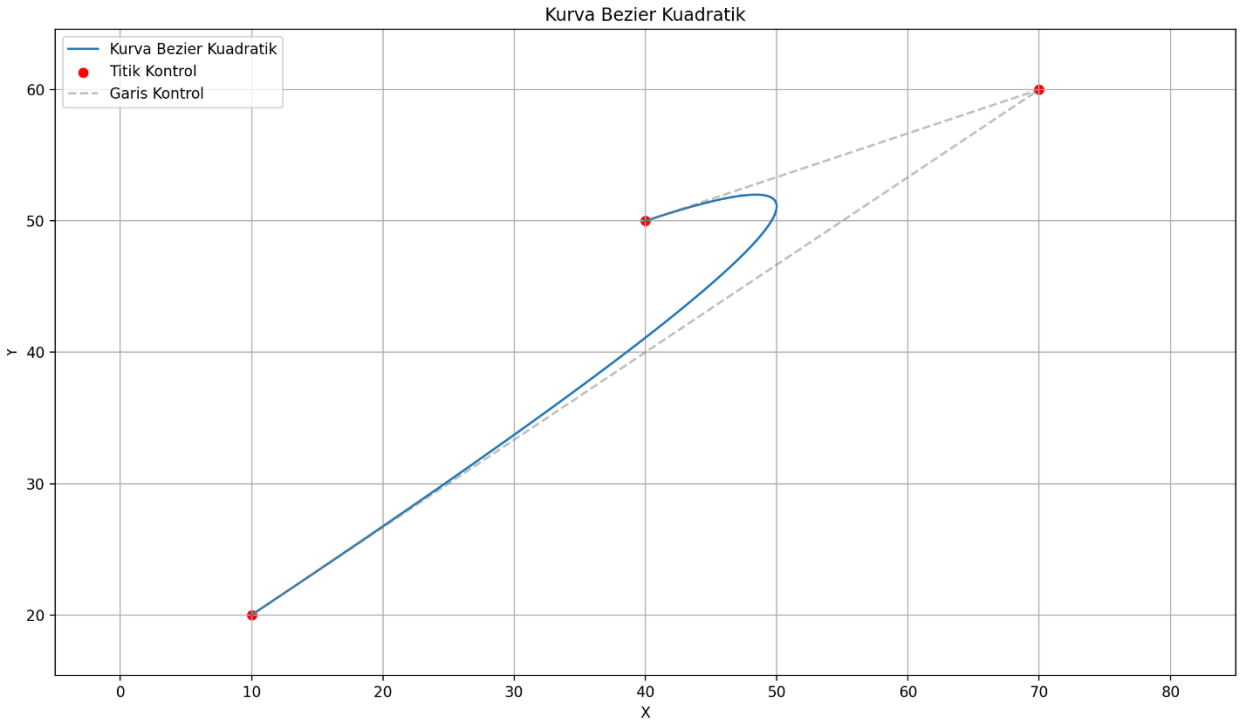
Output 3 :



Input 4 :

```
PS C:\Users\VIVOBOOK> & "C:/Program Files/Python311/python.exe" "d:/STIMA 2024/tucil2-IF2211-AlgoritmaDivide&Conquer/src/BruteForceImplementation.py"
Masukkan koordinat untuk titik-titik Bezier:
Masukkan koordinat x untuk titik awal: 40
Masukkan koordinat y untuk titik awal: 50
Masukkan koordinat x untuk titik kontrol: 70
Masukkan koordinat y untuk titik kontrol: 60
Masukkan koordinat x untuk titik akhir: 10
Masukkan koordinat y untuk titik akhir: 20
```

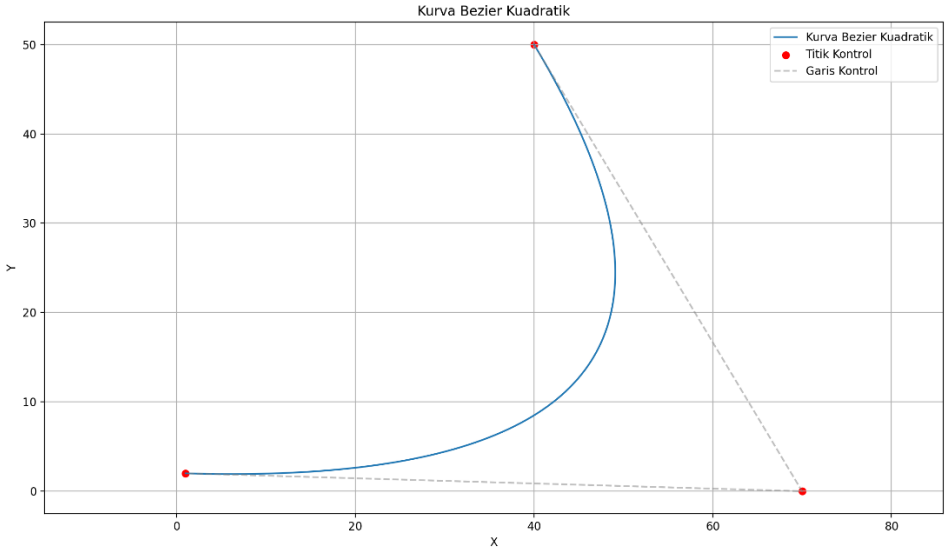
Output 4 :



Input 5 :

```
PS C:\Users\VIVOBOOK> & "C:/Program Files/Python311/python.exe" "d:/STIMA 2024/tucil2-IF2211-AlgoritmaDivide&Conquer/src/BruteForceImplementation.py"
Masukkan koordinat untuk titik-titik Bezier:
Masukkan koordinat x untuk titik awal: 40
Masukkan koordinat y untuk titik awal: 50
Masukkan koordinat x untuk titik kontrol: 70
Masukkan koordinat y untuk titik kontrol: 0
Masukkan koordinat x untuk titik akhir: 1
Masukkan koordinat y untuk titik akhir: 2
```

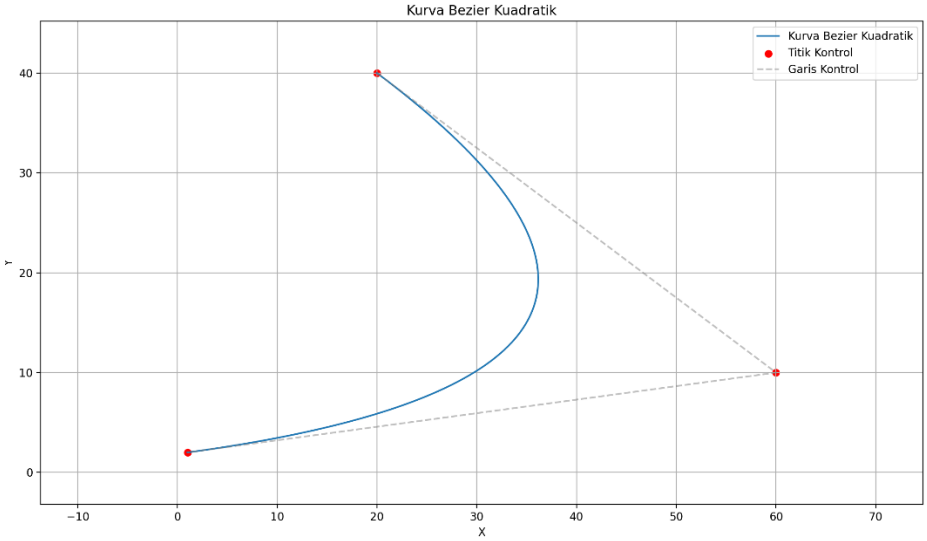
Output 5 :



Input 6 :

```
PS C:\Users\VIVOBOOK> & "C:/Program Files/Python311/python.exe" "d:/STIMA 2024/tucil2-IF2211-AlgoritmaDivide&Conquer/src/BruteForceImplementation.py"
Masukkan koordinat untuk titik-titik Bezier:
Masukkan koordinat x untuk titik awal: 20
Masukkan koordinat y untuk titik awal: 40
Masukkan koordinat x untuk titik kontrol: 60
Masukkan koordinat y untuk titik kontrol: 10
Masukkan koordinat x untuk titik akhir: 1
Masukkan koordinat y untuk titik akhir: 2
```

Output 6 :

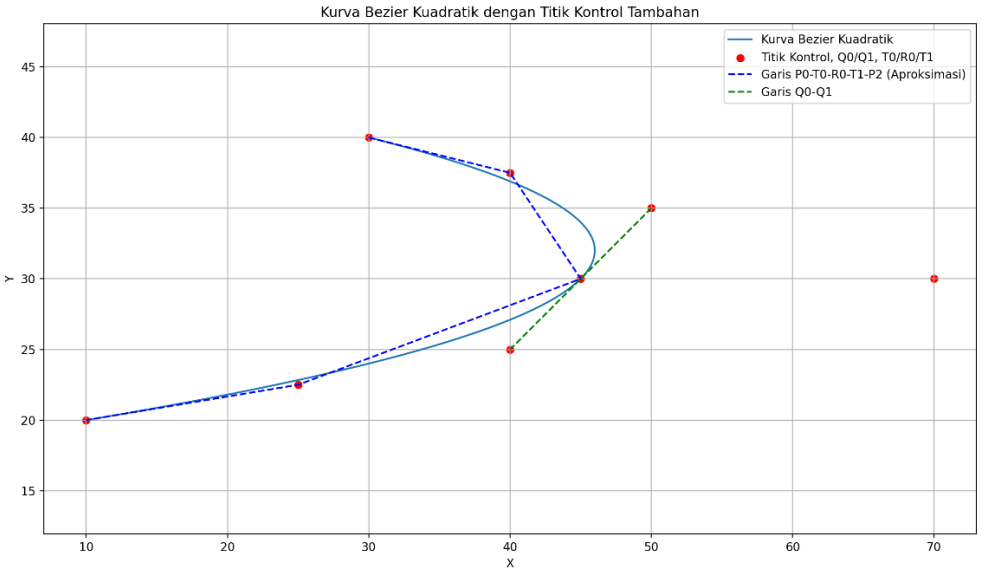


b) Algoritma Divide and Conquer

Input 1 :

```
PS C:\Users\VIVOBOOK> & "C:/Program Files/Python311/python.exe" "d:/STIMA 2024/tucil2-IF2211-AlgoritmaDivide&Conquer/src/DivideandConquerImplementation.py"
Masukkan nilai parameter t (0 <= t <= 1): 0.5
Masukkan koordinat x untuk titik awal: 30
Masukkan koordinat y untuk titik awal: 40
Masukkan koordinat x untuk titik kontrol: 70
Masukkan koordinat y untuk titik kontrol: 30
Masukkan koordinat x untuk titik akhir: 10
Masukkan koordinat y untuk titik akhir: 20
Masukkan jumlah iterasi: 2
```

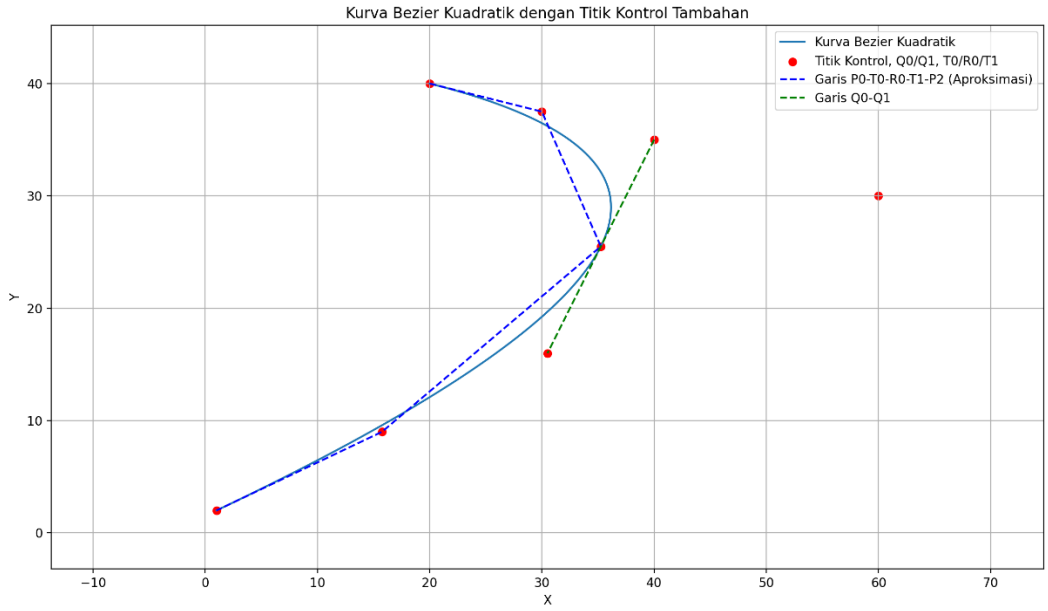
Output 1 :



Input 2 :

```
PS C:\Users\VIVOBOOK> & "C:/Program Files/Python311/python.exe" "d:/STIMA 2024/tucil2-IF2211-AlgoritmaDivide&Conquer/src/DivideandConquerImplementation.py"
Masukkan nilai parameter t (0 <= t <= 1): 0.5
Masukkan koordinat x untuk titik awal: 20
Masukkan koordinat y untuk titik awal: 40
Masukkan koordinat x untuk titik kontrol: 60
Masukkan koordinat y untuk titik kontrol: 30
Masukkan koordinat x untuk titik akhir: 1
Masukkan koordinat y untuk titik akhir: 2
Masukkan jumlah iterasi: 2
```

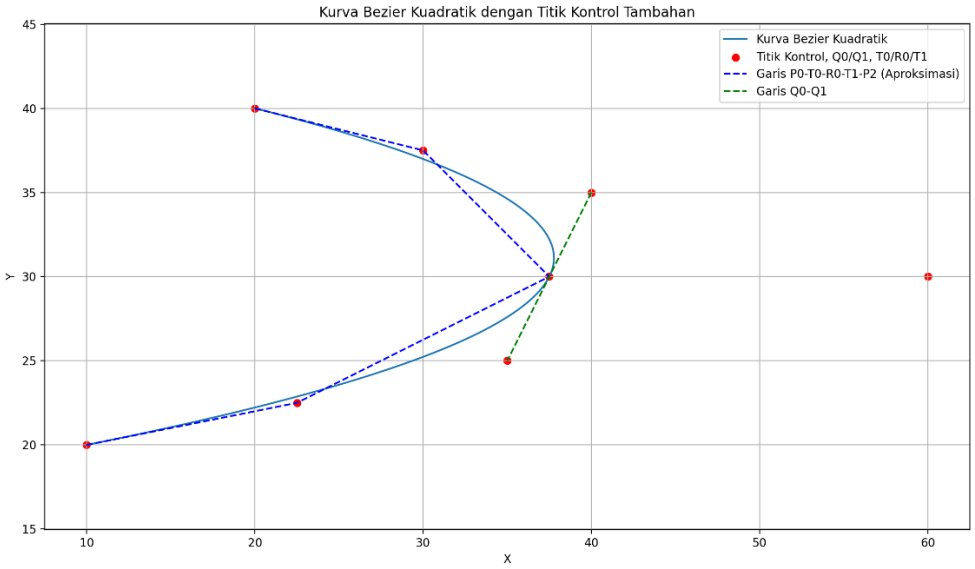
Output 2 :



Input 3 :

```
PS C:\Users\VIVOBOOK> & "C:/Program Files/Python311/python.exe" "d:/STIMA 2024/tucil2-IF2211-AlgoritmaDivide&Conquer/src/DivideandConquerImplementation.py"
Masukkan nilai parameter t (0 <= t <= 1): 0.5
Masukkan koordinat x untuk titik awal: 20
Masukkan koordinat y untuk titik awal: 40
Masukkan koordinat x untuk titik kontrol: 60
Masukkan koordinat y untuk titik kontrol: 30
Masukkan koordinat x untuk titik akhir: 10
Masukkan koordinat y untuk titik akhir: 20
Masukkan jumlah iterasi: 2
```

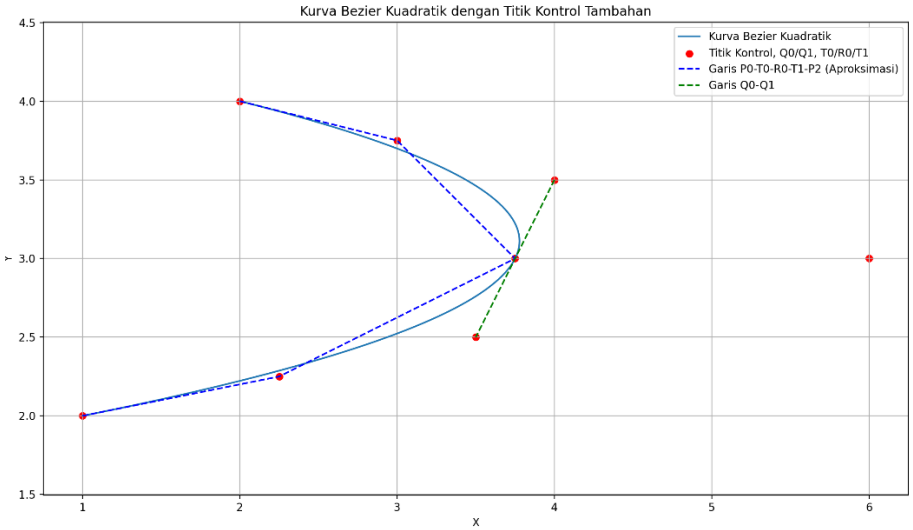
Output 3 :



Input 4 :

```
PS C:\Users\VIVOBOOK> & "C:/Program Files/Python311/python.exe" "d:/STIMA 2024/tucil2-IF2211-AlgoritmaDivide&Conquer/src/DivideandConquerImplementation.py"
Masukkan nilai parameter t (0 <= t <= 1): 0.5
Masukkan koordinat x untuk titik awal: 2
Masukkan koordinat y untuk titik awal: 4
Masukkan koordinat x untuk titik kontrol: 6
Masukkan koordinat y untuk titik kontrol: 3
Masukkan koordinat x untuk titik akhir: 1
Masukkan koordinat y untuk titik akhir: 2
Masukkan jumlah iterasi: 2
```

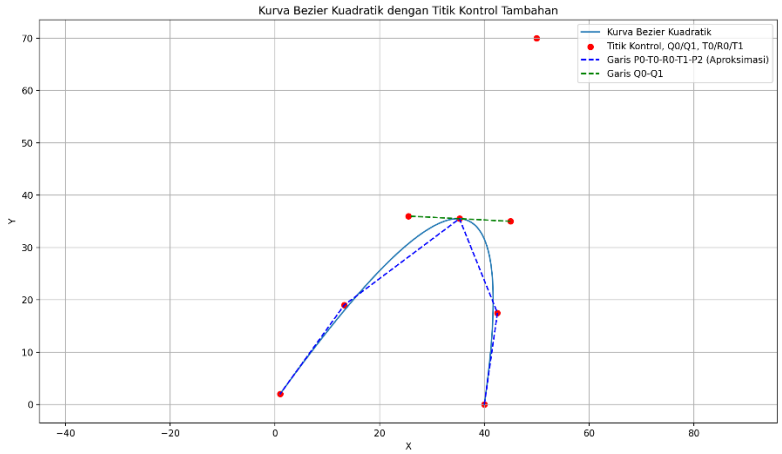
Output 4 :



Input 5 :

```
PS C:\Users\VIVOBOOK> & "C:/Program Files/Python311/python.exe" "d:/STIMA 2024/tucil2-IF2211-AlgoritmaDivide&Conquer/src/DivideandConquerImplementation.py"
Masukkan nilai parameter t (0 <= t <= 1): 0.5
Masukkan koordinat x untuk titik awal: 40
Masukkan koordinat y untuk titik awal: 0
Masukkan koordinat x untuk titik kontrol: 50
Masukkan koordinat y untuk titik kontrol: 70
Masukkan koordinat x untuk titik akhir: 1
Masukkan koordinat y untuk titik akhir: 2
Masukkan jumlah iterasi: 2
```

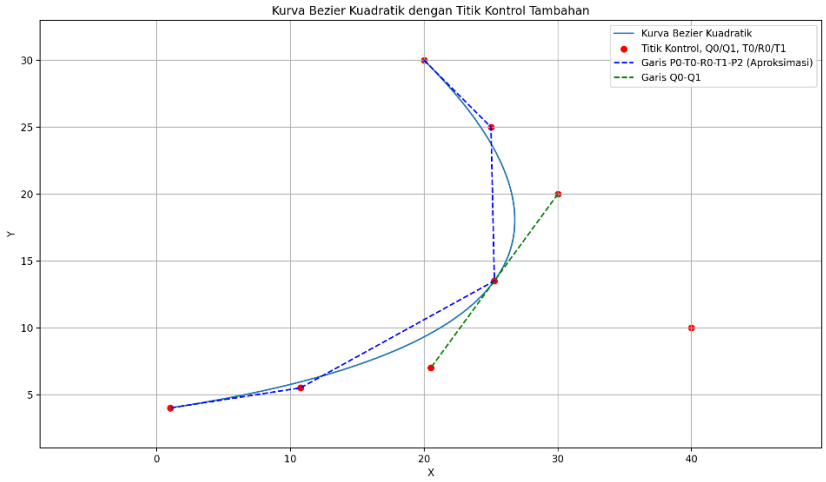
Output 5 :



Input 6 :

```
PS C:\Users\VIVOBOOK> & "C:/Program Files/Python311/python.exe" "d:/STIMA 2024/tucil2-IF2211-AlgoritmaDivide&Conquer/src/DivideandConquerImplementation.py"
Masukkan nilai parameter t (0 <= t <= 1): 0.5
Masukkan koordinat x untuk titik awal: 20
Masukkan koordinat y untuk titik awal: 30
Masukkan koordinat x untuk titik kontrol: 40
Masukkan koordinat y untuk titik kontrol: 10
Masukkan koordinat x untuk titik akhir: 1
Masukkan koordinat y untuk titik akhir: 4
Masukkan jumlah iterasi: 2
```

Output 6 :



B. Analisis Perbandingan

Hasil analisis perbandingan antara solusi brute force dengan divide and conquer menunjukkan perbedaan yang signifikan dalam kinerja dan kompleksitas kedua algoritma.

Dalam konteks algoritma, brute force dan divide and conquer adalah dua pendekatan yang berbeda untuk menyelesaikan masalah. Brute force umumnya berarti mencoba setiap kemungkinan solusi sampai menemukan solusi yang bekerja, sedangkan divide and conquer melibatkan membagi masalah menjadi bagian-bagian yang lebih kecil, menyelesaikan setiap bagian, dan kemudian menggabungkan hasilnya untuk mendapatkan solusi akhir.

Jika kita mengasumsikan bahwa Anda mencoba menyelesaikan suatu masalah yang melibatkan kurva Bezier (misalnya, menemukan titik pada kurva atau mengoptimalkan

beberapa aspek dari kurva tersebut), maka :

- Brute force bisa melibatkan menghitung nilai kurva di banyak titik yang sangat rapat untuk menemukan solusi yang diinginkan (misalnya, titik terdekat dengan koordinat tertentu).
- Divide and conquer bisa melibatkan menggunakan algoritma seperti de Casteljau untuk secara rekursif membagi kurva menjadi bagian yang lebih kecil dan lebih mudah dihitung sampai titik yang diinginkan ditemukan.

Kompleksitas algoritma brute force biasanya $O(n^k)$, di mana n adalah jumlah kemungkinan solusi untuk setiap dimensi, dan k adalah jumlah dimensi dalam masalah. Pendekatan ini bisa menjadi sangat tidak efisien untuk masalah dengan ruang solusi yang besar.

Di sisi lain, algoritma divide and conquer sering memiliki kompleksitas yang jauh lebih baik karena cara mereka memecah masalah. Misalnya, algoritma seperti de Casteljau untuk kurva Bezier memiliki kompleksitas $O(n \log n)$ karena setiap pembagian memotong jumlah titik kontrol menjadi separuh. Ini secara signifikan lebih cepat daripada brute force untuk ruang solusi yang besar atau masalah yang kompleks.

Dalam praktiknya, pendekatan divide and conquer sering kali lebih disukai karena efisiensi dan skalabilitasnya yang lebih baik. Namun, ada situasi di mana brute force bisa lebih sederhana untuk diimplementasikan atau digunakan ketika efisiensi tidak menjadi masalah utama atau ruang solusi terbatas.

Dengan demikian, berdasarkan analisis tersebut, algoritma divide and conquer menunjukkan keunggulan dalam hal kinerja dan kompleksitas algoritma dibandingkan dengan algoritma brute force. Meskipun kedua algoritma menghasilkan hasil yang serupa secara visual, pemilihan algoritma harus memperhitungkan faktor-faktor kinerja dan kompleksitas yang terlibat, tergantung pada kebutuhan spesifik dari aplikasi dan ukuran masalah yang dihadapi.

BAB V Lampiran

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.		✓

Pranala Repository : Tucil2_10023634_10023457