

LAPORAN TUGAS KECIL 2
IF2211 STRATEGI ALGORITMA

**Membangun Kurva Bézier dengan Algoritma Titik Tengah
berbasis Divide and Conquer**



Oleh:

Chelvadinda

13522154

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023/2024

DAFTAR ISI

DAFTAR ISI	2
BAB I ALGORITMA	3
1.1 Algoritma Brute Force	3
1.2 Algoritma divide and conquer	4
BAB II SOURCE CODE	5
BAB III HASIL EKSEKUSI PROGRAM	10
BAB IV LAMPIRAN	14
4.1 Pranala Repository Github	14
4.2 Tabel Ketercapaian Program	14

BAB I

ALGORITMA

1.1 Algoritma Brute Force

Algoritma brute force adalah pendekatan sederhana yang mencoba semua kemungkinan solusi secara berurutan untuk menemukan solusi optimal. Dalam konteks pembentukan kurva Bezier, algoritma brute force menghitung titik-titik pada kurva dengan cara mencoba semua kombinasi titik kontrol dan menghitung posisi titik pada kurva untuk setiap nilai parameter t dalam rentang $[0, 1]$.

Langkah-langkah algoritma brute force untuk membentuk kurva Bezier dapat dijelaskan sebagai berikut:

1. Hitung titik-titik pada kurva Bezier untuk setiap nilai parameter t dalam rentang $[0, 1]$

Pertama, kita tentukan jumlah titik kontrol dari kurva Bezier, yang akan menentukan orde kurva tersebut. Kemudian, kita tentukan rentang nilai t dalam interval $[0, 1]$, yang akan menentukan seberapa dekat titik-titik yang dihasilkan berada pada kurva. Untuk setiap nilai t dalam rentang tersebut, kita akan menghitung posisi titik pada kurva menggunakan rumus umum kurva Bezier.

2. Untuk setiap nilai t , hitung koordinat x dan y dari titik pada kurva menggunakan rumus umum kurva Bezier

Setiap titik pada kurva Bezier dapat dihitung menggunakan rumus umum kurva Bezier yang bergantung pada nilai parameter t dan posisi titik kontrol. Kita akan melakukan perhitungan ini untuk setiap nilai t dalam rentang $[0, 1]$, sehingga mendapatkan serangkaian titik yang akan membentuk kurva Bezier. Dengan melakukan perhitungan untuk setiap nilai t dalam rentang $[0, 1]$, algoritma brute force akan menghasilkan serangkaian titik yang membentuk kurva Bezier. Namun, karena algoritma ini mencoba semua kemungkinan kombinasi titik kontrol, dapat menjadi tidak efisien terutama untuk jumlah titik kontrol yang besar.

Implementasi algoritma brute force dalam bahasa Python melibatkan perhitungan untuk setiap nilai t dalam rentang $[0, 1]$, menghitung posisi titik pada kurva Bezier menggunakan rumus umum, dan mengumpulkan titik-titik tersebut dalam sebuah daftar (list).

1.2 Algoritma *divide and conquer*

Analisis dan implementasi algoritma divide and conquer untuk membentuk kurva Bezier melibatkan membagi masalah menjadi bagian-bagian yang lebih kecil, menyelesaikan setiap bagian secara terpisah, dan kemudian menggabungkan solusi-solusi tersebut untuk membentuk solusi akhir. Berikut adalah langkah-langkahnya:

1. Memisahkan Masalah: Langkah pertama dalam algoritma divide and conquer adalah memisahkan masalah besar menjadi bagian-bagian yang lebih kecil. Dalam konteks pembentukan kurva Bezier, kita dapat membagi titik-titik kontrol menjadi dua kelompok.
2. Rekursi: Setelah memisahkan masalah menjadi bagian-bagian yang lebih kecil, langkah berikutnya adalah menyelesaikan setiap bagian secara terpisah. Dalam hal ini, kita akan menggunakan rekursi untuk menghitung kurva Bezier untuk setiap bagian titik kontrol.
3. Gabungkan Solusi: Setelah mendapatkan kurva Bezier untuk setiap bagian, langkah terakhir adalah menggabungkan solusi-solusi tersebut untuk membentuk solusi akhir. Ini dapat dilakukan dengan menggabungkan kurva-kurva menjadi satu kesatuan yang mulus.

Dengan menggunakan pendekatan divide and conquer, kita dapat mengurangi kompleksitas permasalahan secara signifikan dengan memecahnya menjadi bagian-bagian yang lebih kecil dan lebih mudah dipecahkan. Ini memungkinkan kita untuk mengatasi masalah pembentukan kurva Bezier dengan lebih efisien daripada menggunakan pendekatan brute force.

BAB II SOURCE CODE

2.1 bruteForce_curveBezier.py

Source Code	Description
<pre> import matplotlib.pyplot as plt import numpy as np import time def get_input(): p0 = tuple(map(float, input("Koordinat titik p0 (x y): ").split())) p1 = tuple(map(float, input("Koordinat titik p1 (x y): ").split())) p2 = tuple(map(float, input("Koordinat titik p2 (x y): ").split())) iterasi = int(input("Banyak iterasi yang dilakukan: ")) n = iterasi for i in range(n): if i == 0: iterasi = 3 else: iterasi = iterasi * 2 - 1 return p0, p1, p2, iterasi </pre>	<p>Fungsi get_input()</p> <ul style="list-style-type: none"> - Fungsi ini bertugas untuk mengambil input dari pengguna berupa koordinat titik (x, y) dan jumlah iterasi yang diinginkan. - Input yang dimasukkan akan diproses menjadi tuple-tuple titik (x, y) dan jumlah iterasi. - Fungsi ini mengembalikan tuple berisi koordinat titik p0, p1, p2, dan jumlah iterasi.
<pre> # BRUTE FORCE # def brute_force_bezier(points, iterasi): result = [] curvepoints = [] if iterasi > 1: for i in range(iterasi): t = i / (iterasi-1) x = (1 - t) ** 2 * points[0][0] + 2 * (1 - t) * t * points[1][0] + t ** 2 * points[2][0] y = (1 - t) ** 2 * points[0][1] + 2 * (1 - t) * t * points[1][1] + t ** 2 * points[2][1] result.append([x, y]) curvepoints.append([x, y]) else: result.extend(points) curvepoints.extend(points) return result, curvepoints </pre>	<p>Fungsi brute_force_bezier(points, iterasi)</p> <ul style="list-style-type: none"> - Fungsi ini mengimplementasikan algoritma brute force untuk menghitung kurva Bezier. - Input fungsi ini adalah titik-titik kontrol (points) dan jumlah iterasi. - Fungsi ini mengembalikan dua list: list hasil (result) yang berisi titik-titik pada kurva Bezier, dan list (curvepoints) yang berisi titik-titik kontrol serta hasil.

```
def plot_curve(result, title, color, label):
    x = [result[0] for result in result]
    y = [result[1] for result in result]
    plt.plot(x, y, color, label=label)
    plt.scatter(x, y, c=color, s=10, alpha=0.5)

def plot_control_polygon(points, color):
    x = [point[0] for point in points]
    y = [point[1] for point in points]
    plt.plot(x, y, linestyle='--', color=color, label='Control Polygon')

def plot_control_points(points, color):
    x = [point[0] for point in points]
    y = [point[1] for point in points]
    plt.scatter(x, y, c=color, label='Control Points', s=50)
```

Fungsi `plot_curve(result, title, color, label)`

- Fungsi ini digunakan untuk memplot kurva hasil dari algoritma brute force.
- Input fungsi ini adalah list `result` yang berisi titik-titik pada kurva, judul plot (`title`), warna kurva (`color`), dan label untuk kurva (`label`).

Fungsi `plot_control_polygon(points, color)`

- Fungsi ini digunakan untuk memplot poligon kontrol yang terbentuk dari titik-titik kontrol.
- Input fungsi ini adalah list `points` yang berisi titik-titik kontrol, dan warna poligon (`color`).

Fungsi `plot_control_points(points, color)`

- Fungsi ini digunakan untuk memplot titik-titik kontrol.
- Input fungsi ini adalah list `points` yang berisi titik-titik kontrol, dan warna titik kontrol (`color`).

```
def main():
    p0, p1, p2, iterasi = get_input()
    # Start time
    start_time = time.time()

    # Brute Force
    points = [p0, p1, p2]
    t = np.linspace(0, 1)
    result, curvepoints = brute_force_bezier(points, iterasi)

    # End time
    end_time = time.time()
```

Fungsi `main()`

- Fungsi utama yang menjalankan alur utama program.
- Memanggil fungsi `get_input()` untuk mendapatkan input pengguna.
- Menghitung kurva Bezier menggunakan fungsi `brute_force_bezier()` dan memplotnya.
- Memplot poligon kontrol dan titik-titik kontrol.
- Menghitung dan mencetak waktu eksekusi.
- Memastikan bahwa program dijalankan hanya jika file ini dieksekusi

<pre> # Plot Bezier Curve plot_curve(result, 'Bezier Curves', 'green', 'Brute Force') # Plot Control Polygon plot_control_polygon(points, 'black') # Plot Control Points plot_control_points(points, 'red') plt.legend() plt.show() # Execution time execution_time = (end_time - start_time) * 1000 print("Execution time:", execution_time, "ms") if __name__ == "__main__": main() </pre>	<p>secara langsung, bukan diimpor sebagai modul.</p> <ul style="list-style-type: none"> - Memanggil fungsi main() untuk menjalankan program utama. - Semua fungsi tersebut bekerja sama untuk menghitung dan memvisualisasikan kurva Bezier serta titik-titik kontrolnya, sambil mengukur waktu eksekusi.
--	---

2.2 dnc_curveBezier.py

Source Code	Description
<pre> import matplotlib.pyplot as plt import numpy as np import time curve_points = [] control_points = [] mid_points = [] def get_input(): while True: try: p0 = tuple(map(float, input("Koordinat titik p0 (x y): ").split())) p1 = tuple(map(float, input("Koordinat titik p1 (x y): ").split())) p2 = tuple(map(float, input("Koordinat titik p2 (x y): ").split())) iteration = int(input("Banyak iterasi yang dilakukan: ")) if iteration <= 0: print("Banyak iterasi harus lebih besar dari 0. Silakan coba lagi.") continue return p0, p1, p2, iteration except ValueError: print("Input tidak valid. Pastikan Anda memasukkan angka.") except KeyboardInterrupt: print("\nProgram berhenti.") exit() return p0, p1, p2, iteration </pre>	<p>Fungsi get_input()</p> <ul style="list-style-type: none"> - Fungsi ini bertanggung jawab untuk meminta input dari pengguna berupa koordinat titik kontrol (p0, p1, p2) dan jumlah iterasi. - Input yang dimasukkan akan diproses, dan jika input tidak valid (misalnya, bukan angka), pesan kesalahan akan ditampilkan. - Fungsi ini akan terus meminta input hingga input yang valid diterima. - Setelah mendapatkan input yang valid, fungsi ini mengembalikan tuple yang berisi koordinat titik kontrol dan jumlah iterasi.

```
# divide and conquer #
def result(p0, p1, p2, iteration):
    curve_points.append(p0)
    divide_conquer(p0, p1, p2, 0, iteration)
    curve_points.append(p2)]

def divide_conquer(p0, p1, p2, current_Iteration, iteration):
    if current_Iteration < iteration:
        mid1 = [(p0[0] + p1[0]) / 2, (p0[1] + p1[1]) / 2]
        mid2 = [(p1[0] + p2[0]) / 2, (p1[1] + p2[1]) / 2]
        mid = [(mid1[0] + mid2[0]) / 2, (mid1[1] + mid2[1]) / 2]
        mid_points.append(mid1)
        mid_points.append(mid2)
        mid_points.append(mid)

        current_Iteration += 1
        # kiri
        divide_conquer(p0, mid1, mid, current_Iteration, iteration)
        curve_points.append(mid)
        # kanan
        divide_conquer(mid, mid2, p2, current_Iteration, iteration)
```

Fungsi result(p0, p1, p2, iteration)

- Fungsi ini bertugas untuk menginisialisasi proses rekursif pemisahan dan penaklukan (Divide and Conquer) untuk menghasilkan kurva Bezier.
- Titik p0 dan p2 ditambahkan ke daftar titik kurva (curve_points).
- Fungsi ini memanggil fungsi divide_conquer() untuk memulai proses pemisahan dan penaklukan.

Fungsi divide_conquer(p0, p1, p2, current_Iteration, iteration)

- Fungsi rekursif ini membagi segmen kurva yang diberikan menjadi dua segmen, kemudian mengulangi proses untuk kedua segmen tersebut.
- Pemisahan dilakukan hingga mencapai iterasi maksimum yang diinginkan.
- Titik tengah (midpoint) dari segmen ditempatkan ke dalam daftar titik tengah (mid_points) untuk visualisasi.
- Proses ini terus berulang hingga mencapai iterasi maksimum.

```
def plot_curve(curve_points, title, color, label):
    x = [curvePoint[0] for curvePoint in curve_points]
    y = [curvePoint[1] for curvePoint in curve_points]
    plt.plot(x, y, color, label=label)
    plt.scatter(x, y, c=color, s=10, alpha=0.5)

def plot_control_polygon(points, color):
    x = [point[0] for point in points]
    y = [point[1] for point in points]
    plt.plot(x, y, linestyle='--', color=color, label='Control Polygon')

def plot_control_points(points, color):
    x = [point[0] for point in points]
    y = [point[1] for point in points]
    plt.scatter(x, y, c=color, label='Control Points', s=50)
```

Fungsi plot_curve(curve_points, title, color, label)

- Fungsi ini bertugas untuk memplot kurva Bezier berdasarkan titik-titik yang diperoleh dari proses Divide and Conquer.
- Kurva Bezier dipetakan ke dalam grafik menggunakan matplotlib.
- Input fungsi ini adalah daftar titik-titik kurva (curve_points), judul plot (title), warna kurva (color), dan label untuk kurva (label).

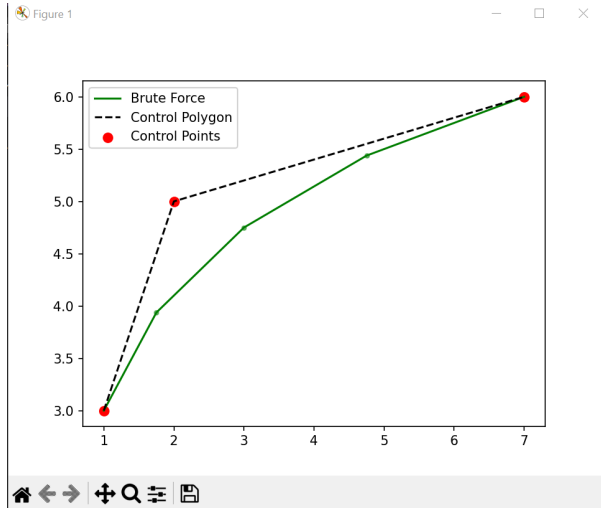
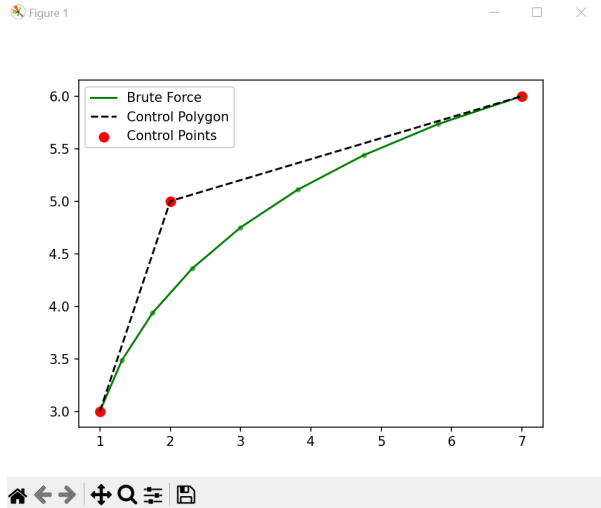
Fungsi

	<p>plot_control_polygon(points, color)</p> <ul style="list-style-type: none"> - Fungsi ini digunakan untuk memplot poligon kontrol (control polygon) yang terbentuk dari titik-titik kontrol. - Inputnya adalah daftar titik-titik kontrol (points) dan warna poligon kontrol (color). <p>Fungsi plot_control_points(points, color)</p> <ul style="list-style-type: none"> - Fungsi ini digunakan untuk memplot titik-titik kontrol. - Inputnya adalah daftar titik-titik kontrol (points) dan warna titik kontrol (color).
<pre>def main(): p0, p1, p2, iteration = get_input() # Start time start_time = time.time() points = [p0,p1,p2] # Devide Conquer result(p0, p1, p2, iteration) end_time = time.time() # end time setelah dnc # # Plot Bezier Curves plot_curve(curve_points, 'Bezier Curves', 'orange', 'Divide and Conquer') # Plot Control Polygon plot_control_polygon(points, 'black') # Plot Control Points plot_control_points(points, 'red') plt.legend() plt.show() # waktu eksekusi execution_time = (end_time - start_time) * 1000 print("Execution time:", execution_time, "ms") if __name__ == "__main__": main()</pre>	<p>Fungsi main()</p> <ul style="list-style-type: none"> - Fungsi utama yang menjalankan alur utama program. - Meminta input dari pengguna, menghitung kurva Bezier menggunakan metode Divide and Conquer, dan memplot hasilnya. - Mengukur dan mencetak waktu eksekusi. - Memastikan bahwa program dijalankan hanya jika file ini dieksekusi secara langsung, bukan diimpor sebagai modul. - Memanggil fungsi main() untuk menjalankan program utama.

BAB III

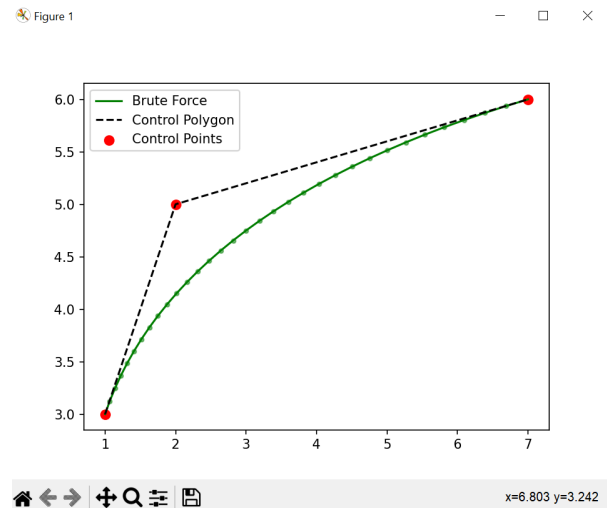
HASIL EKSEKUSI PROGRAM

3.1 bruteForce_curveBezier.py

Input	Output
<p>Koordinat titik p0 (x y): 1 3 Koordinat titik p1 (x y): 2 5 Koordinat titik p2 (x y): 7 6 Banyak iterasi yang dilakukan: 2</p> <pre>Koordinat titik p0 (x y): 1 3 Koordinat titik p1 (x y): 2 5 Koordinat titik p2 (x y): 7 6 Banyak iterasi yang dilakukan: 2 Execution time: 0.0 ms</pre>	 <p>Execution time: 0.0 ms</p>
<p>Koordinat titik p0 (x y): 1 3 Koordinat titik p1 (x y): 2 5 Koordinat titik p2 (x y): 7 6 Banyak iterasi yang dilakukan: 3</p> <pre>Koordinat titik p0 (x y): 1 3 Koordinat titik p1 (x y): 2 5 Koordinat titik p2 (x y): 7 6 Banyak iterasi yang dilakukan: 3 Execution time: 0.0 ms</pre>	 <p>Execution time: 0.0 ms</p>

Koordinat titik p0 (x y): 1 3
 Koordinat titik p1 (x y): 2 5
 Koordinat titik p2 (x y): 7 6
 Banyak iterasi yang dilakukan: 5

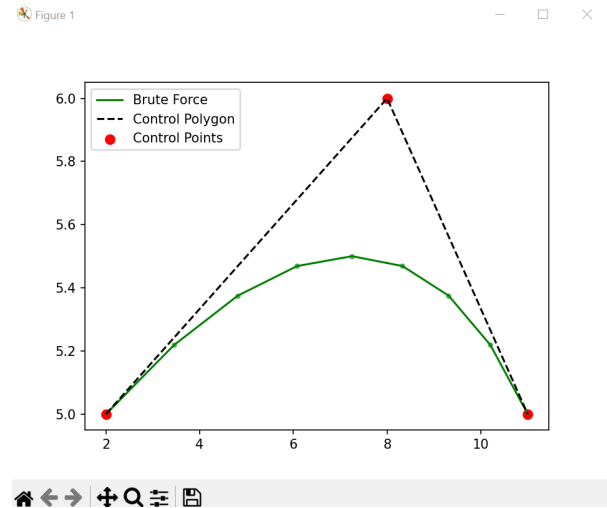
```
Koordinat titik p0 (x y): 1 3
Koordinat titik p1 (x y): 2 5
Koordinat titik p2 (x y): 7 6
Banyak iterasi yang dilakukan: 5
Execution time: 0.0 ms
```



Execution time: 0.0 ms

Koordinat titik p0 (x y): 2 5
 Koordinat titik p1 (x y): 8 6
 Koordinat titik p2 (x y): 11 5
 Banyak iterasi yang dilakukan: 3

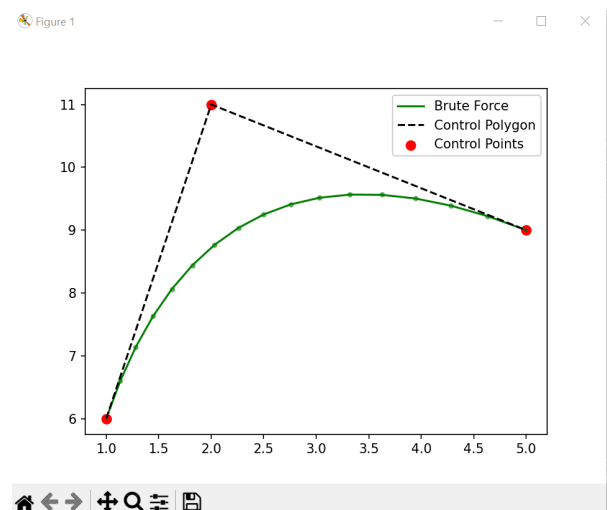
```
Koordinat titik p0 (x y): 2 5
Koordinat titik p1 (x y): 8 6
Koordinat titik p2 (x y): 11 5
Banyak iterasi yang dilakukan: 3
Execution time: 0.0 ms
```



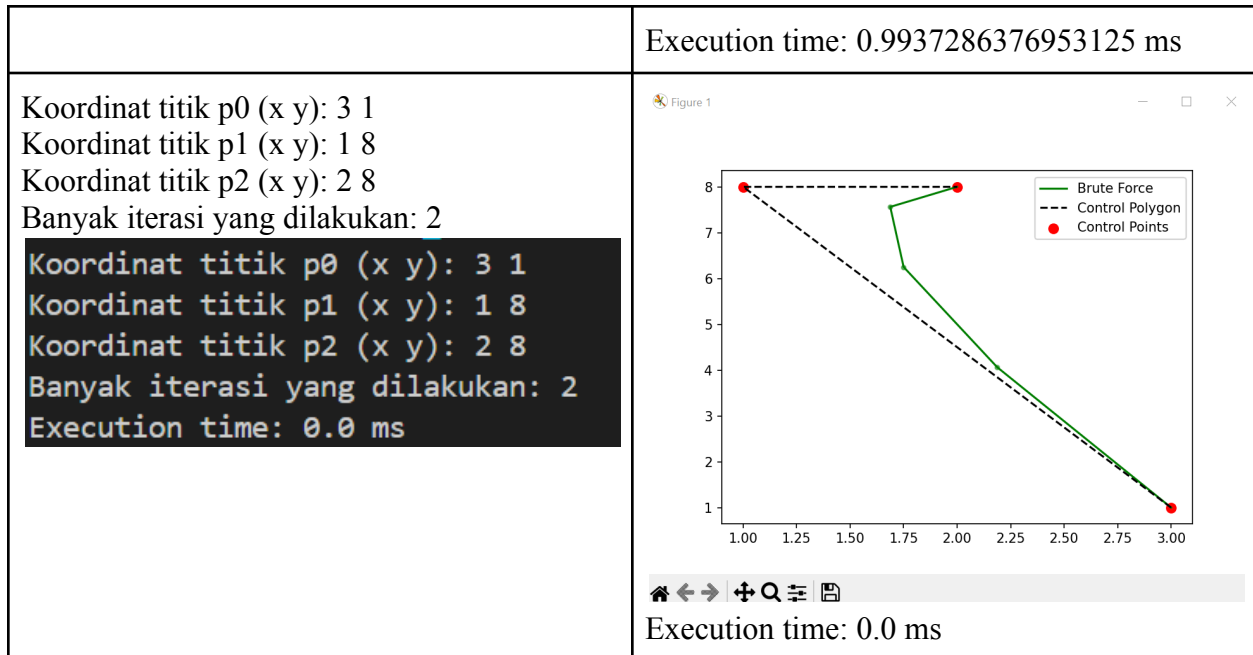
Execution time: 0.0 ms

Koordinat titik p0 (x y): 1 6
 Koordinat titik p1 (x y): 2 11
 Koordinat titik p2 (x y): 5 9
 Banyak iterasi yang dilakukan: 4

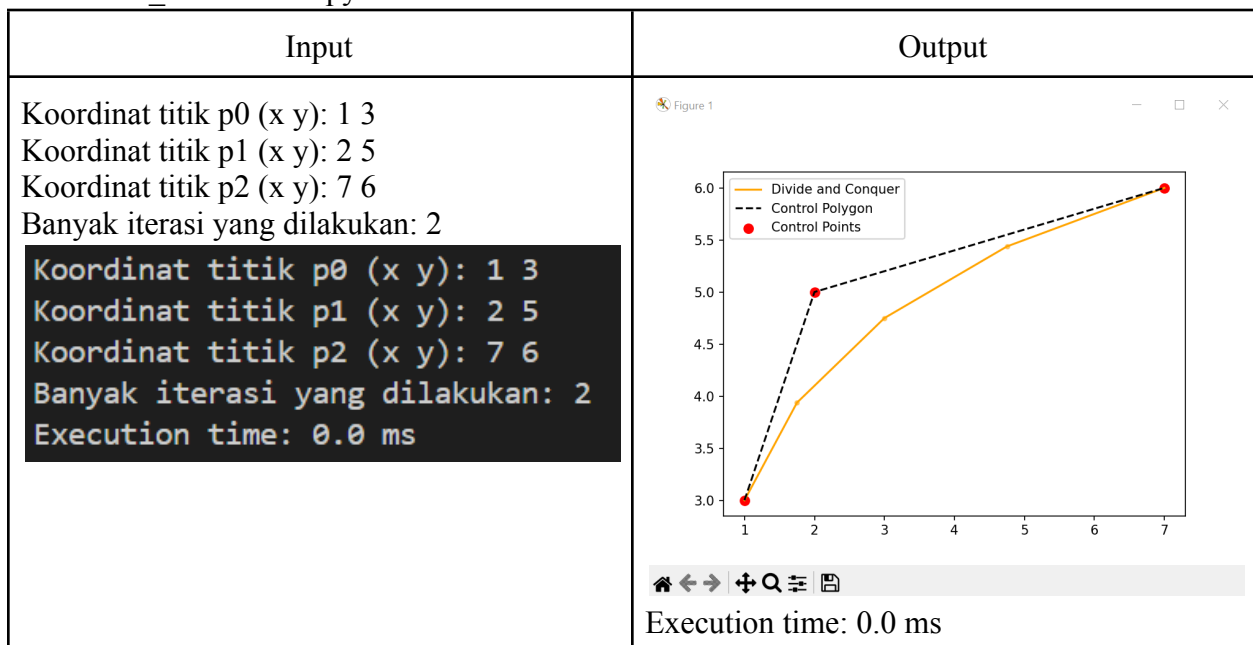
```
Koordinat titik p0 (x y): 1 6
Koordinat titik p1 (x y): 2 11
Koordinat titik p2 (x y): 5 9
Banyak iterasi yang dilakukan: 4
Execution time: 0.9937286376953125 ms
```



Execution time: 0.9937286376953125 ms

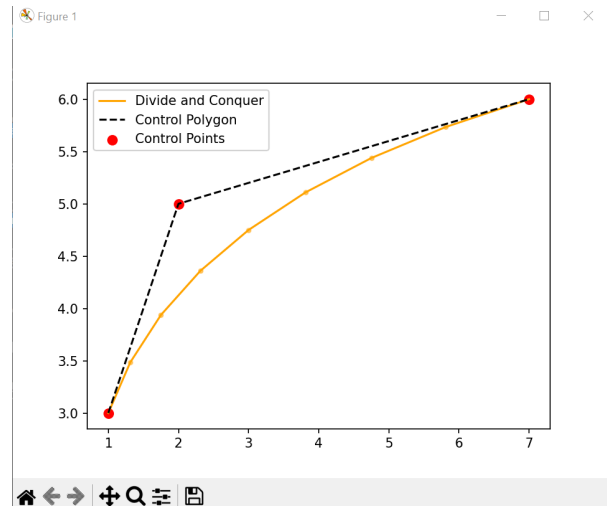


3.2 dnc_curveBezier.py



Koordinat titik p0 (x y): 1 3
 Koordinat titik p1 (x y): 2 5
 Koordinat titik p2 (x y): 7 6
 Banyak iterasi yang dilakukan: 3

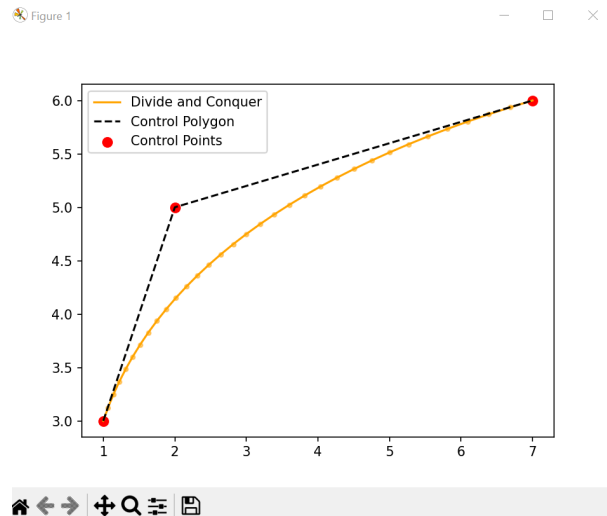
```
Koordinat titik p0 (x y): 1 3
Koordinat titik p1 (x y): 2 5
Koordinat titik p2 (x y): 7 6
Banyak iterasi yang dilakukan: 3
Execution time: 0.0 ms
```



Execution time: 0.0 ms

Koordinat titik p0 (x y): 1 3
 Koordinat titik p1 (x y): 2 5
 Koordinat titik p2 (x y): 7 6
 Banyak iterasi yang dilakukan: 5

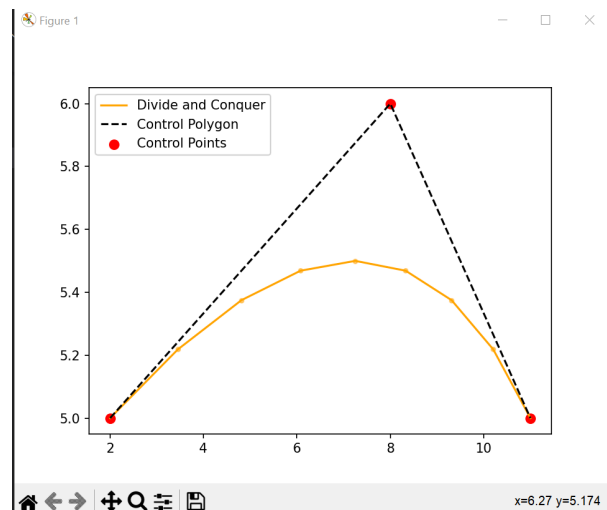
```
Koordinat titik p0 (x y): 1 3
Koordinat titik p1 (x y): 2 5
Koordinat titik p2 (x y): 7 6
Banyak iterasi yang dilakukan: 5
Execution time: 0.0 ms
```



Execution time: 0.0 ms

Koordinat titik p0 (x y): 2 5
 Koordinat titik p1 (x y): 8 6
 Koordinat titik p2 (x y): 11 5
 Banyak iterasi yang dilakukan: 3

```
Koordinat titik p0 (x y): 2 5
Koordinat titik p1 (x y): 8 6
Koordinat titik p2 (x y): 11 5
Banyak iterasi yang dilakukan: 3
Execution time: 0.0 ms
```



x=6.27 y=5.174

	Execution time: 0.0 ms
<p>Koordinat titik p0 (x y): 1 6 Koordinat titik p1 (x y): 2 11 Koordinat titik p2 (x y): 5 9 Banyak iterasi yang dilakukan: 4</p> <pre>Koordinat titik p0 (x y): 1 6 Koordinat titik p1 (x y): 2 11 Koordinat titik p2 (x y): 5 9 Banyak iterasi yang dilakukan: 4 Execution time: 0.0 ms</pre>	<p>Figure 1</p> <p>Execution time: 0.0 ms</p>
<p>Koordinat titik p0 (x y): 3 1 Koordinat titik p1 (x y): 1 8 Koordinat titik p2 (x y): 2 8 Banyak iterasi yang dilakukan: 2</p> <pre>Koordinat titik p0 (x y): 3 1 Koordinat titik p1 (x y): 1 8 Koordinat titik p2 (x y): 2 8 Banyak iterasi yang dilakukan: 2 Execution time: 0.0 ms</pre>	<p>Figure 1</p> <p>Execution time: 0.0 ms</p>

BAB IV

LAMPIRAN

- 4.1 Pranala Repository Github
https://github.com/chelvadinda/Tucil2_13522154
- 4.2 Tabel Ketercapaian Program

Tabel 4.2.1. Ketercapaian Program

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