Laporan Tugas Kecil 2
IF2211 Strategi Algoritma
Kurva Bézier
Semester II Tahun 2023/2024

Disusun Oleh:

Muhammad Althariq Fairuz 13522027
Moh Fairuz Alauddin Yahya 13522057

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

DAFTAR ISI

BAB 1

Deskripsi Tugas

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.

BAB 2

Landasan Teori

2.1 Algoritma Brute Force

Algoritma Brute Force adalah algoritma yang menggunakan pendekatan yang lempang (straightforward) untuk memecahkan suatu persoalan.

2.2 Algoritma Divide and Conquer

Algoritma Divide and Conquer adalah pendekatan pemecahan masalah yang melibatkan pembagian persoalan menjadi beberapa sub-persoalan yang mirip dengan persoalan awal tetapi berukuran lebih kecil. Idealnya, setiap sub-persoalan memiliki ukuran yang hampir sama. Langkah selanjutnya adalah menyelesaikan setiap sub-persoalan, baik secara langsung jika ukurannya sudah cukup kecil atau secara rekursif jika masih besar. Akhirnya, solusi dari setiap sub-persoalan digabungkan untuk membentuk solusi bagi persoalan awal.

BAB 3

Implementasi Program

3.1 Implementasi Algoritma Brute Force

Algoritma brute force diimplementasikan ini sebagai pembanding dengan algoritma divide and conquer yang menjadi topik utama pada tugas kecil kali ini. Implementasi algoritma brute force pada tugas kecil kali ini memiliki 2 fungsi utama:

1. **Fungsi calculate_pascal_triangle_row**

   Fungsi ini menghitung dan mengembalikan baris tertentu dalam Segitiga Pascal berdasarkan nomor baris yang diberikan. Segitiga Pascal digunakan untuk menghitung koefisien binomial, yang digunakan dalam rumus Bezier.

2. **Fungsi create_bezier_curve**

   Fungsi ini mengaproksimasi kurva Bezier dengan iterasi brute force. Berikut adalah langkah-langkahnya:

   - Menginisialisasi daftar kosong untuk titik-titik kurva.
   - Menghitung koefisien binomial dengan memanggil calculate_pascal_triangle_row.
   - Mengiterasi dari 0 hingga jumlah iterasi. Untuk setiap iterasi:
     - Menghitung parameter t yang merupakan rasio iterasi saat ini terhadap total iterasi.
     - Menginisialisasi x dan y menjadi 0. Ini akan menjadi koordinat titik pada kurva.
     - Mengiterasi melalui semua titik kontrol. Untuk setiap titik kontrol:
       - Menghitung koefisien yang merupakan hasil kali koefisien binomial, (1 - t) ^ {(n - j)} dan t ^ j, di mana n adalah jumlah total titik kontrol dan j adalah indeks titik kontrol saat ini.
       - Menambahkan hasil kali koefisien dan koordinat x dan y titik kontrol ke x dan y.
     - Menambahkan titik dengan koordinat x dan y ke daftar titik kurva.
   - Menetapkan titik-titik kurva ke self.curve_points.

3.2 Implementasi Algoritma Divide and Conquer

Berikut adalah fungsi utama dalam algoritma divide and conquer:

1. **Fungsi midpoint**

   Fungsi ini menghitung titik tengah antara dua titik. Titik tengah dihitung dengan mencari rata-rata koordinat x dan y dari dua titik tersebut.

2. **Fungsi populate_bezier_points**

   Fungsi ini mengisi titik-titik pada kurva Bezier berdasarkan titik-titik kontrol dan iterasi saat ini. Fungsi ini bekerja sebagai berikut:

   - Jika iterasi saat ini kurang dari jumlah iterasi yang ditentukan, fungsi ini akan membuat dua daftar baru untuk titik-titik kontrol kiri dan kanan.
   - Fungsi ini kemudian menghitung titik-titik tengah antara setiap pasangan titik kontrol berurutan sampai hanya tersisa satu titik tengah.
   - Titik-titik kontrol baru ditambahkan ke daftar titik-titik kontrol kiri dan kanan.
   - Fungsi ini kemudian dipanggil secara rekursif untuk titik-titik kontrol kiri dan kanan, dan titik-titik tengah ditambahkan ke daftar titik-titik Bezier.

3. **Fungsi calculate_dnc_bezier_points**

   Fungsi ini menghitung titik-titik pada kurva Bezier menggunakan algoritma divide and conquer. Fungsi ini bekerja sebagai berikut:

   - Jika ada kurang dari 3 titik kontrol, fungsi ini akan mengembalikan titik-titik kontrol tersebut.
   - Jika tidak, fungsi ini akan menginisialisasi daftar titik-titik Bezier dengan titik kontrol pertama, memanggil fungsi populate_bezier_points untuk mengisi titik-titik Bezier, dan kemudian menambahkan titik kontrol terakhir ke daftar titik-titik Bezier.

4. **Fungsi generate_bezier_dnc**

Fungsi ini menghasilkan titik-titik kurva Bezier berdasarkan titik kontrol dan iterasi. Fungsi ini menghasilkan titik-titik kurva Bezier untuk setiap iterasi dari 1 hingga jumlah iterasi.

Secara umum, alur programnya adalah sebagai berikut:

1. Fungsi generate_bezier_dnc dipanggil dengan titik kontrol dan jumlah iterasi sebagai argumen. Fungsi ini akan menghasilkan titik-titik kurva Bezier untuk setiap iterasi dari 1 hingga k . Untuk setiap iterasi, fungsi calculate_dnc_bezier_points dipanggil.

2. Fungsi calculate_dnc_bezier_points menginisialisasi daftar titik Bezier dengan titik kontrol pertama, memanggil populate_bezier_points untuk mengisi titik Bezier, dan menambahkan titik kontrol terakhir ke daftar titik Bezier. Jika ada kurang dari 3 titik kontrol, titik kontrol itu sendiri dikembalikan.

3. Fungsi populate_bezier_points mengisi titik-titik Bezier berdasarkan titik kontrol dan iterasi saat ini. Fungsi ini membagi titik kontrol menjadi dua set, menghitung titik tengah, dan mengulangi proses ini secara rekursif sampai mencapai tingkat iterasi yang diinginkan. Setelah itu, semua titik tengah ini digabungkan untuk membentuk kurva Bezier.

4. Setelah semua iterasi selesai, generate_bezier_dnc mengembalikan daftar titik-titik kurva Bezier untuk setiap iterasi.

Secara garis besar, ide dari algoritma divide and conquer pada tugas ini adalah:

1. **Inisialisasi**

   Algoritma ini dimulai dengan menerima titik-titik kontrol sebagai input. Titik-titik kontrol ini adalah titik-titik yang akan digunakan untuk membentuk kurva Bezier.

2. **Pembagian (Divide)**

   Algoritma kemudian membagi sekumpulan titik kontrol menjadi dua subset, yaitu titik-titik kiri dan titik-titik kanan. Pembagian ini dilakukan dengan mengambil titik tengah dari setiap pasangan titik kontrol berturut-turut hingga menyisakan satu titik tengah tunggal. Titik tengah ini kemudian digunakan sebagai pemisah antara subset kiri dan kanan.

3. **Rekursi**

Setelah pembagian, algoritma memanggil dirinya sendiri secara rekursif untuk menghitung titik-titik kurva Bezier dari subset kiri dan kanan. Proses ini terus berlanjut hingga mencapai tingkat iterasi yang ditentukan oleh pengguna.

4. **Penggabungan (Combine)**

   Setelah semua iterasi selesai, hasil dari subset kiri dan kanan digabungkan untuk membentuk kurva Bezier secara keseluruhan. Penggabungan ini dilakukan dengan mempertimbangkan urutan dan proporsi dari masing-masing titik kontrol pada subset kiri dan kanan.

Dengan demikian, algoritma ini menghasilkan kurva Bezier dari sekumpulan n titik kontrol yang diberikan dengan menggunakan metode divide and conquer.

BAB 4

Source Code Program

4.1 Screenshot Algoritma Brute Force

brute_force.py

```python
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from models import Point

class BruteForceBezier:
    def __init__(self, points:list[Point], iterations:int) -> None:
        self.curve_points =[]
        self.points = points
        self.iterations = iterations

    def calculate_pascal_triangle_row(self,row_number: int) -> list[int]:
        """Calculate a row in Pascal's Triangle based on the given row number."""
        if row_number == 0:
            return [1]

        row = [1]
        for i in range(1, row_number + 1):
            # Initialize a new row with 1s and update the middle elements
            new_row = [1] * (i + 1)
            for j in range(1, i):
                new_row[j] = row[j - 1] + row[j]
            row = new_row

        return row

    def create_bezier_curve(self) -> None:
        """Approximate the Bezier curve by brute-force iteration."""
        curve = []
        n = len(self.points) - 1
        coefficients = self.calculate_pascal_triangle_row(n)
        for i in range(self.iterations + 1):
            t = i / self.iterations
            x = y = 0
            # Build the curve point by iterating through the control points
            for j in range(n + 1):
                coefficient = coefficients[j] * (1 - t) ** (n - j) * t ** j
                x += coefficient * self.points[j].x
                y += coefficient * self.points[j].y
            curve.append(Point(x, y))
        self.curve_points = curve
```

Snipped

4.2 Screenshot Algoritma Divide and Conquer

divide_and_conquer.py

```python
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from models import Point
import timeit

class DNCBezier:
    def __init__(self, points:list[Point], iterations:int):
        self.curve_points = []
        self.points = points
        self.iterations = iterations

    def midpoint(self, point1: Point, point2: Point) -> Point:
        """Calculate the midpoint between two points."""
        return Point((point1.x + point2.x) / 2, (point1.y + point2.y) / 2)

    def populate_bezier_points(self, cp: list[Point],iterations ,current_iteration: int, bezier_points: list[Point]) -> None:
        """Populate Bezier points based on control points and current iteration."""
        if current_iteration < iterations:
            left_cp = []
            right_cp = []
            mids = cp.copy()

            # Calculate midpoints until there is only one midpoint left
            while len(mids) > 1:
                # Append new control point on left and right bezier latest points
                left_cp.append(mids[0])
                right_cp.insert(0, mids[-1])

                # Calculate midpoints for each pair of control points
                mids = [self.midpoint(mids[i], mids[i + 1]) for i in range(len(mids) - 1)]

            left_cp.append(mids[0])
            right_cp.insert(0, mids[0])

            # Recursively populate bezier points for left and right control points
            self.populate_bezier_points(left_cp,iterations, current_iteration + 1, bezier_points)
            bezier_points.extend(mids)
            self.populate_bezier_points(right_cp, iterations,current_iteration + 1, bezier_points)

    # Calculate Bezier points based on control points and iterations
    def calculate_dnc_bezier_points(self, iterations) -> list[Point]:
        # If there are less than 3 control points, return the control points
        if len(self.points) < 3:
            return self.points

        # Initialize the list of bezier points with the first control point
        bezier_points = [self.points[0]]
        self.populate_bezier_points(self.points,iterations, 0, bezier_points)
        bezier_points.append(self.points[-1])
        self.curve_points = bezier_points

        return bezier_points

    def generate_bezier_dnc(self, control_points: list[Point], iterations: int) -> list[list[Point]]:
        """Generate Bezier curve points based on control points and iterations."""
        result = []

        for i in range(iterations):
            result.append(self.calculate_dnc_bezier_points(i+1))

        return result
```

Snipped

BAB 5

Output Program

5.1 Input dan Output untuk Algoritma Brute Force

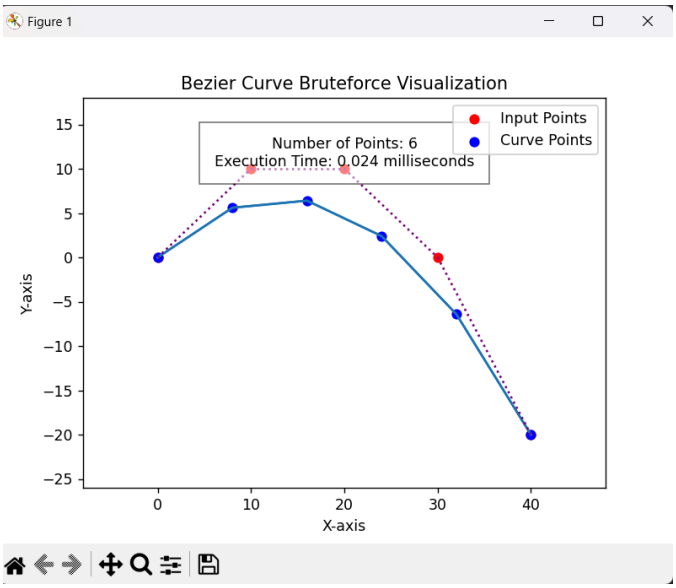
1. Test case 1

```
Method available:
1. Brute Force
2. Divide and Conquer
3. Exit

Choose method: 1
Number of points: 5
Input your points in the format 'x,y'. Example: 1,2

P1: 0,0
P2: 10,10
P3: 20,10
P4: 30,0
P5: 40,-20
Iterations: 5
Number of Points: 6
Execution Time: 0.024 milliseconds
Curve point 1: 0.0, 0.0
Curve point 2: 8.0, 5.6
Curve point 3: 16.0, 6.4
Curve point 4: 24.0, 2.4
Curve point 5: 32.0, -6.4
Curve point 6: 40.0, -20.0
```

Gambar 5.1.1 Input test case 1 brute force



Gambar 5.1.2 Output test case 1 brute force

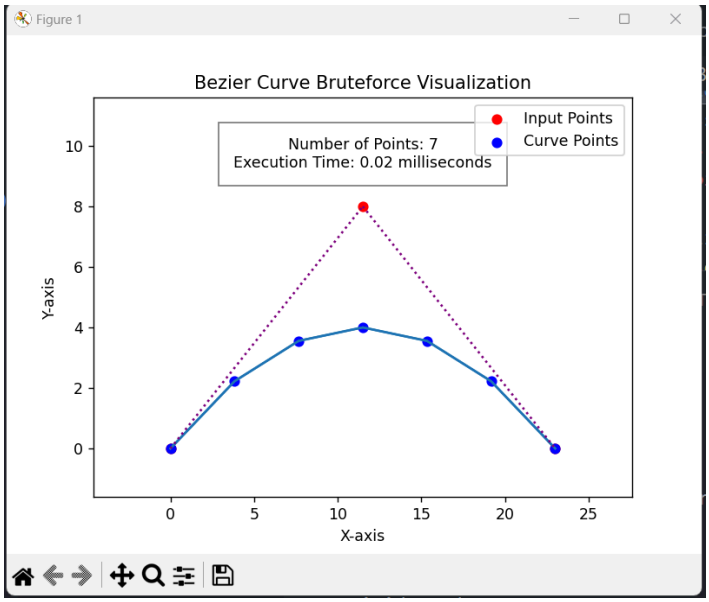
2. Test case 2

```
Method available:
1. Brute Force
2. Divide and Conquer
3. Exit

Choose method: 1
Number of points: 3
Input your points in the format 'x,y'. Example: 1,2

P1: 0,0
P2: 11.5,8
P3: 23,0
Iterations: 6
Number of Points: 7
Execution Time: 0.02 milliseconds
Curve point 1: 0.0, 0.0
Curve point 2: 3.8333, 2.2222
Curve point 3: 7.6667, 3.5556
Curve point 4: 11.5, 4.0
Curve point 5: 15.3333, 3.5556
Curve point 6: 19.1667, 2.2222
Curve point 7: 23.0, 0.0
```

Gambar 5.1.3 Input test case 2 brute force



Gambar 5.1.4 Output test case 2 brute force

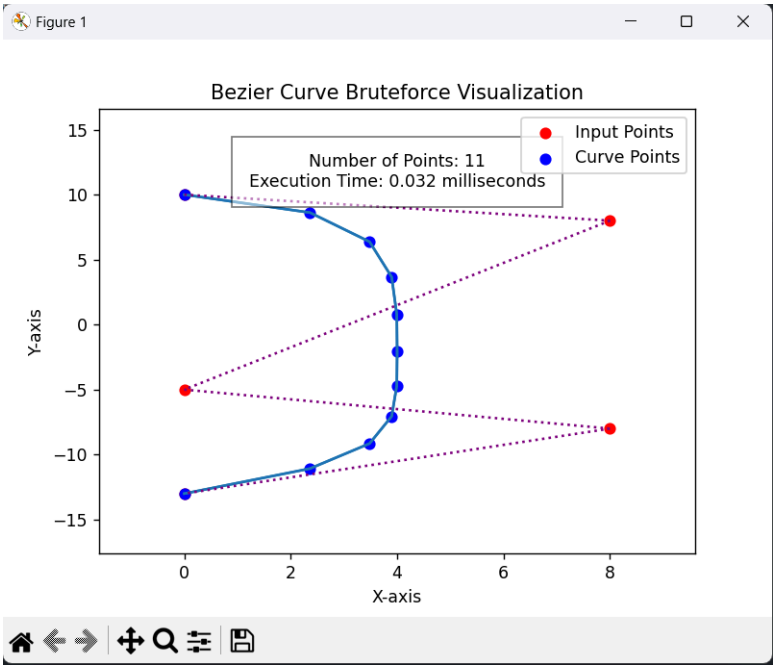
3. Test case 3

```
Method available:
1. Brute Force
2. Divide and Conquer
3. Exit

Choose method: 1
Number of points: 5
Input your points in the format 'x,y'. Example: 1,2

P1: 0,10
P2: 8,8
P3: 0,-5
P4: 8,-8
P5: 0,-13
Iterations: 10
Number of Points: 11
Execution Time: 0.032 milliseconds
Curve point 1: 0.0, 10.0
Curve point 2: 2.3616, 8.6207
Curve point 3: 3.4816, 6.3792
Curve point 4: 3.8976, 3.6607
Curve point 5: 3.9936, 0.7712
Curve point 6: 4.0, -2.0625
Curve point 7: 3.9936, -4.6928
Curve point 8: 3.8976, -7.0513
Curve point 9: 3.4816, -9.1488
Curve point 10: 2.3616, -11.0753
Curve point 11: 0.0, -13.0
```

Gambar 5.1.5 Input test case 3 brute force



Gambar 5.1.6 Output test case 3 brute force

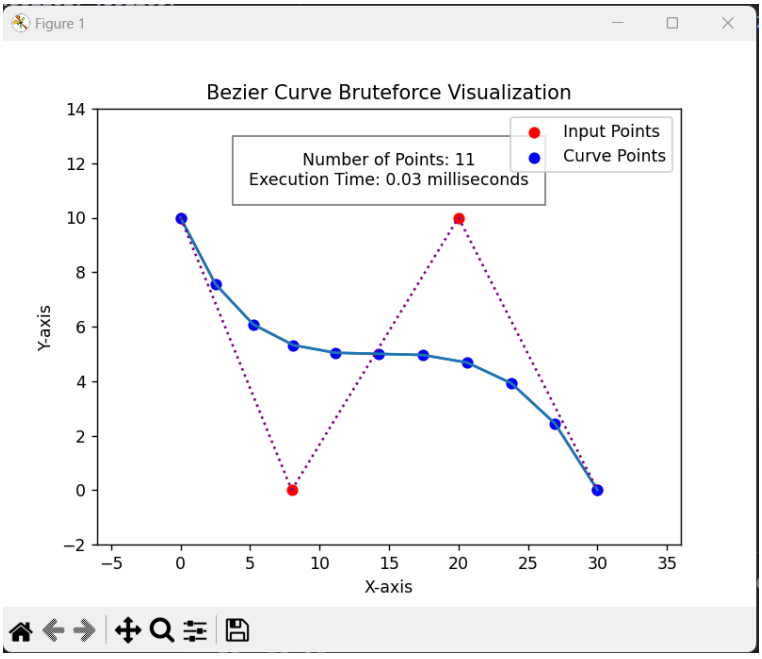
4. Test case 4

```
Method available:
1. Brute Force
2. Divide and Conquer
3. Exit

Choose method: 1
Number of points: 4
Input your points in the format 'x,y'. Example: 1,2

P1: 0,10
P2: 8.0
Invalid input format. Please enter two numbers separated by a comma.
P2: 8,0
P3: 20,10
P4: 30,0
Iterations: 10
Number of Points: 11
Execution Time: 0.03 milliseconds
Curve point 1: 0.0, 10.0
Curve point 2: 2.514, 7.56
Curve point 3: 5.232, 6.08
Curve point 4: 8.118, 5.32
Curve point 5: 11.136, 5.04
Curve point 6: 14.25, 5.0
Curve point 7: 17.424, 4.96
Curve point 8: 20.622, 4.68
Curve point 9: 23.808, 3.92
Curve point 10: 26.946, 2.44
Curve point 11: 30.0, 0.0
```

Gambar 5.1.7 Input test case 4 brute force



Gambar 5.1.8 Output test case 4 brute force

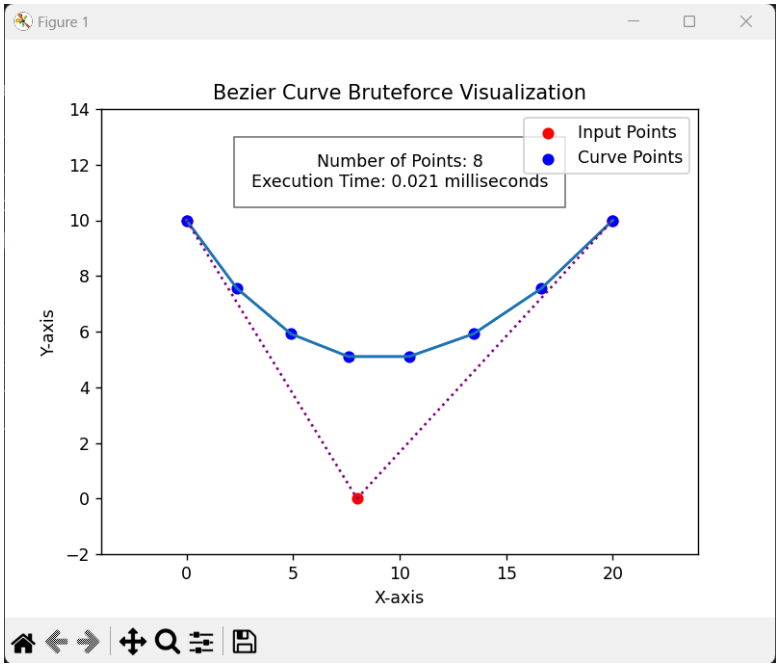
5. Test case 5

```
Method available:
1. Brute Force
2. Divide and Conquer
3. Exit

Choose method: 1
Number of points: 3
Input your points in the format 'x,y'. Example: 1,2

P1: 0,10
P2: 8,0
P3: 20,10
Iterations: 7
Number of Points: 8
Execution Time: 0.021 milliseconds
Curve point 1: 0.0, 10.0
Curve point 2: 2.3673, 7.551
Curve point 3: 4.898, 5.9184
Curve point 4: 7.5918, 5.102
Curve point 5: 10.449, 5.102
Curve point 6: 13.4694, 5.9184
Curve point 7: 16.6531, 7.551
Curve point 8: 20.0, 10.0
```

Gambar 5.1.9 Input test case 5 brute force



Gambar 5.1.10 Output test case 5 brute force

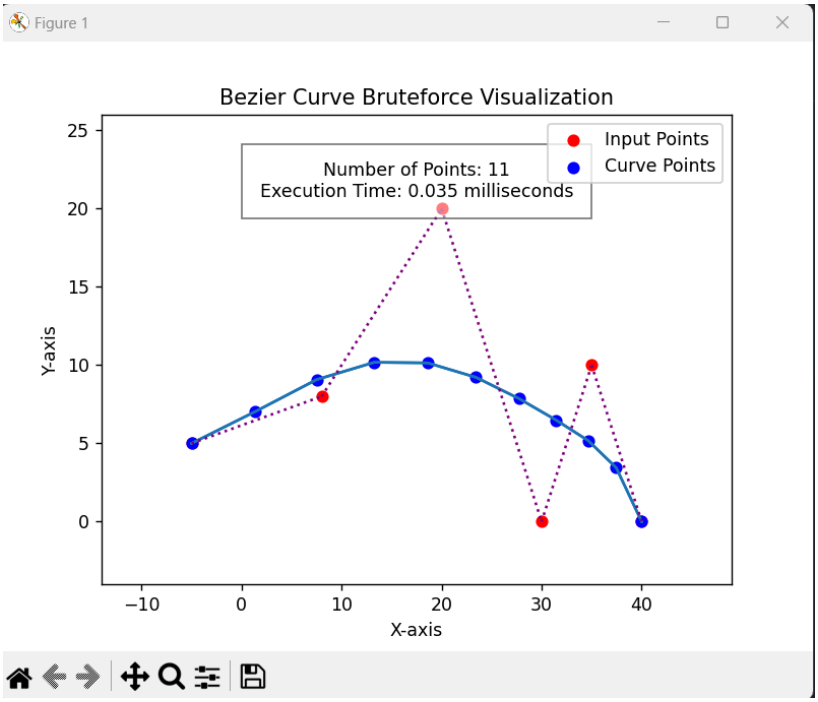
6. Test case 6

```
1. Brute Force
2. Divide and Conquer
3. Exit

Choose method: 1
Number of points: 6
Input your points in the format 'x,y'. Example: 1,2

P1: -5,5
P2: 8,8
P3: 20,20
P4: 30,0
P5: 35,10
P6: 40,0
Iterations: 10
Number of Points: 11
Execution Time: 0.035 milliseconds
Curve point 1: -5.0, 5.0
Curve point 2: 1.3891, 7.0394
Curve point 3: 7.5072, 9.0752
Curve point 4: 13.2733, 10.179
Curve point 5: 18.6064, 10.1424
Curve point 6: 23.4375, 9.2188
Curve point 7: 27.7216, 7.8656
Curve point 8: 31.4497, 6.4865
Curve point 9: 34.6608, 5.1728
Curve point 10: 37.4539, 3.4461
Curve point 11: 40.0, 0.0
```

Gambar 5.1.11 Input test case 6 brute force



Gambar 5.1.12 Output test case 6 brute force

5.2 Input dan Output untuk Algoritma Divide and Conquer

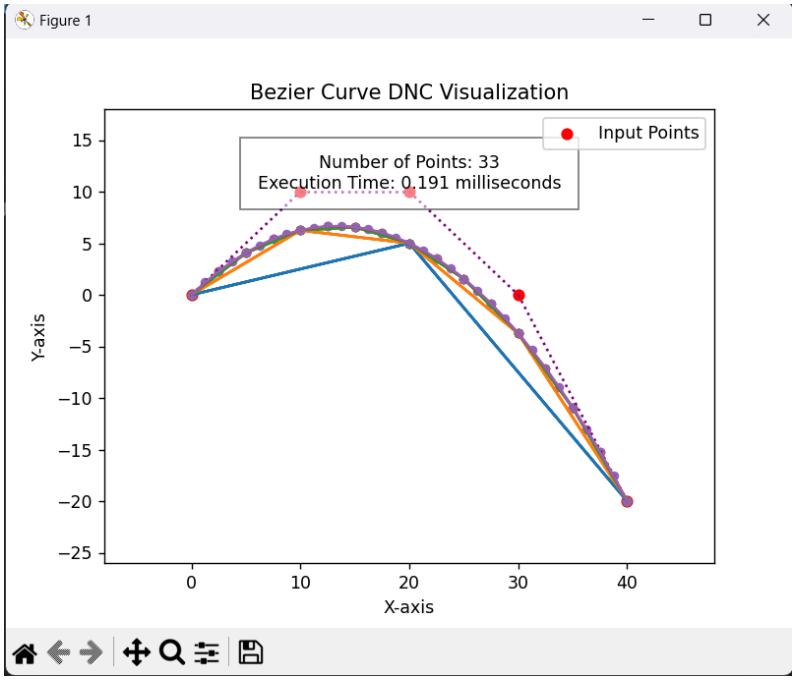
1. Test case 1

```
Method available:
1. Brute Force
2. Divide and Conquer
3. Exit

Choose method: 2
Number of points: 5
Input your points in the format 'x,y'. Example: 1,2

P1: 0,0
P2: 10,10
P3: 20,10
P4: 30,0
P5: 40,-20
Iterations: 5
Number of Points: 33
Execution Time: 0.191 milliseconds
```

Gambar 5.2.1 Input test case 1 divide and conquer



Gambar 5.2.2 Output test case 1 divide and conquer

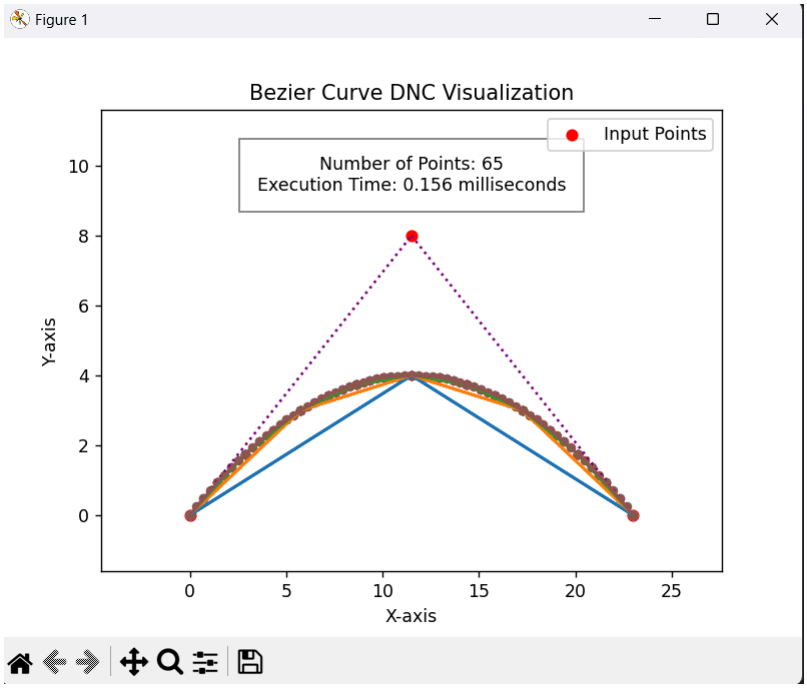
2. Test case 2

```
Method available:
1. Brute Force
2. Divide and Conquer
3. Exit

Choose method: 2
Number of points: 3
Input your points in the format 'x,y'. Example: 1,2

P1: 0,0
P2: 11.5,8
P3: 23,0
Iterations: 6
Number of Points: 65
Execution Time: 0.156 milliseconds
```

Gambar 5.2.3 Input test case 2 divide and conquer



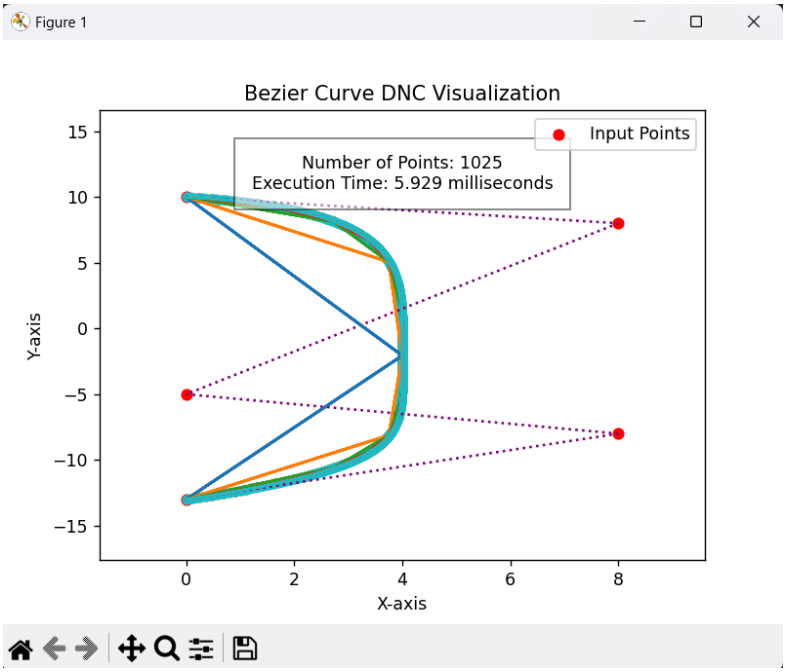
Gambar 5.2.4 Output test case 2 divide and conquer

3. Test case 3

```
Choose method: 2
Number of points: 5
Input your points in the format 'x,y'. Example: 1,2

P1: 0.10\
Invalid input format. Please enter two numbers separated by a comma.
P1: 0,10
P2: 8,8
P3: 0,-5
P4: 8,-8
P5: 0,-13
Iterations: 10
Number of Points: 1025
Execution Time: 5.929 milliseconds
```

Gambar 5.2.5 Input test case 3 divide and conquer



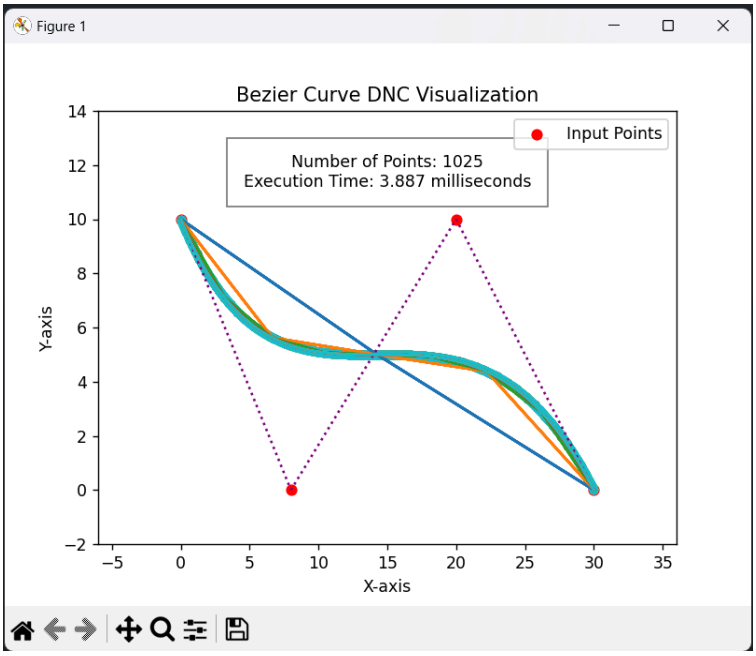
Gambar 5.2.6 Output test case 3 divide and conquer

4. Test case 4

```
Choose method: 2
Number of points: 4
Input your points in the format 'x,y'. Example: 1,2

P1: 0,10
P2: 8,0
P3: 20,10
P4: 30,0
Iterations: 10
Number of Points: 1025
Execution Time: 3.887 milliseconds
```

Gambar 5.2.7 Input test case 4 divide and conquer



Gambar 5.2.8 Output test case 4 divide and conquer

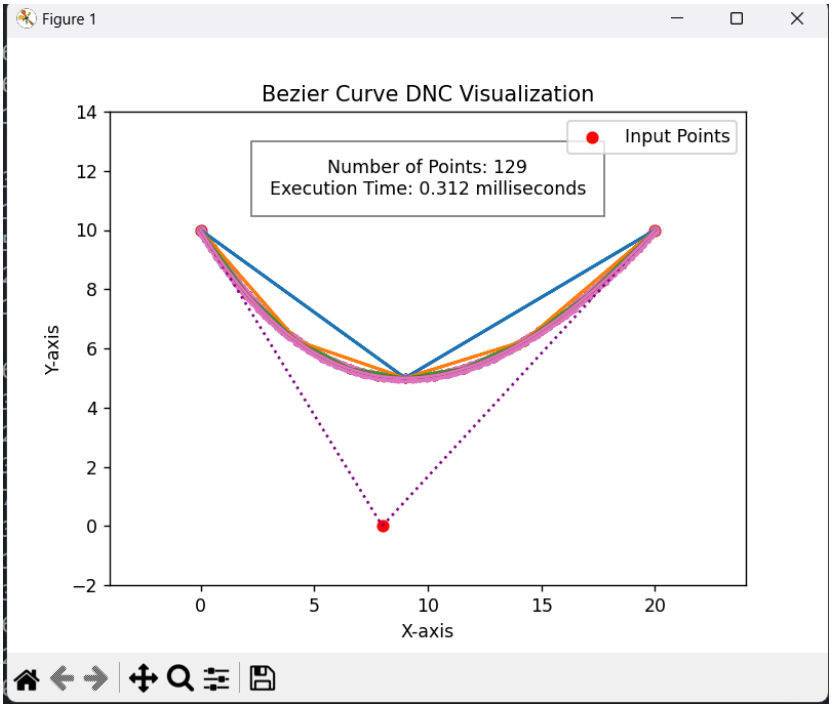
5. Test case 5

```
Method available:
1. Brute Force
2. Divide and Conquer
3. Exit

Choose method: 2
Number of points: 3
Input your points in the format 'x,y'. Example: 1,2

P1: 0,10
P2: 8,0
P3: 20,10
Iterations: 7
Number of Points: 129
Execution Time: 0.312 milliseconds
```

Gambar 5.2.9 Input test case 5 divide and conquer



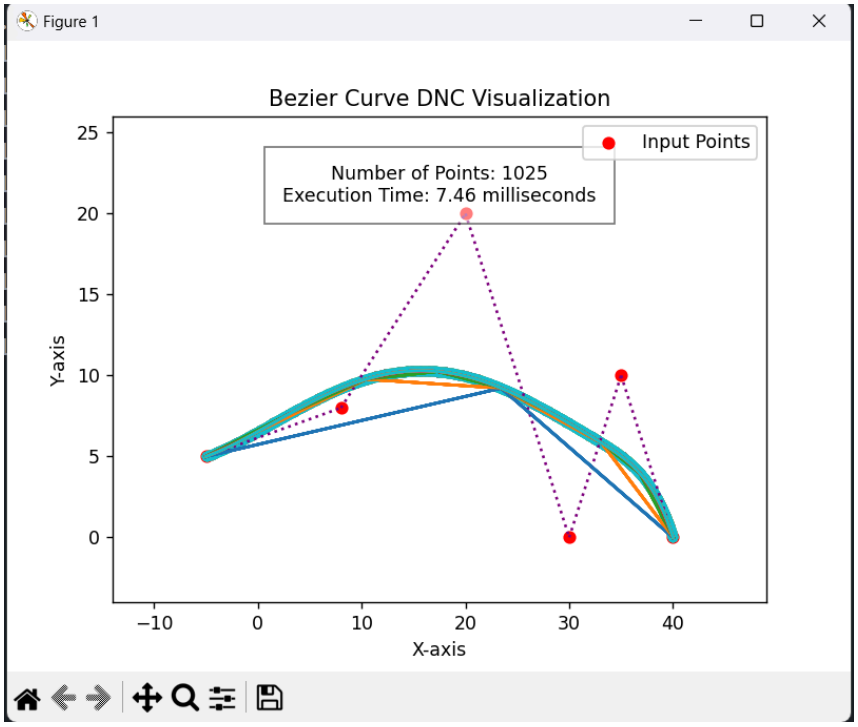
Gambar 5.2.10 Output test case 5 divide and conquer

6. Test case 6

```
Choose method: 2
Number of points: 6
Input your points in the format 'x,y'. Example: 1,2

P1: -5,5
P2: 8,8
P3: 20,20
P4: 30,0
P5: 35,10
P6: 40,0
Iterations: 10
Number of Points: 1025
Execution Time: 7.46 milliseconds
```

Gambar 5.2.11 Input test case 6 divide and conquer

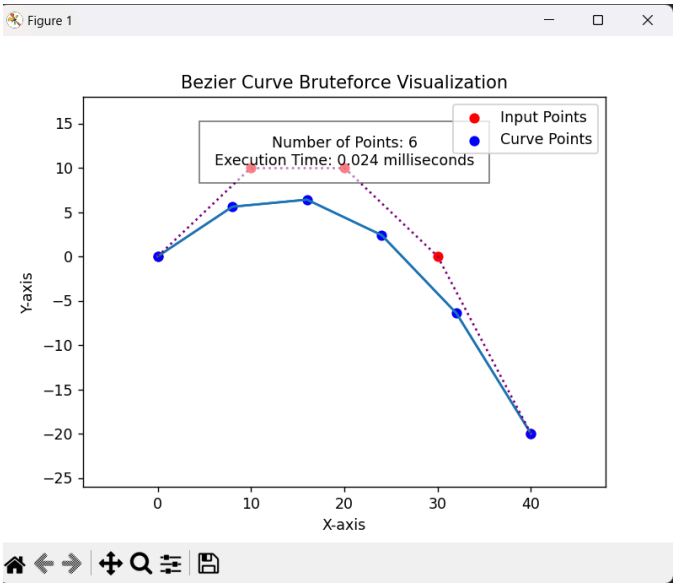


Gambar 5.2.12 Output test case 6 divide and conquer

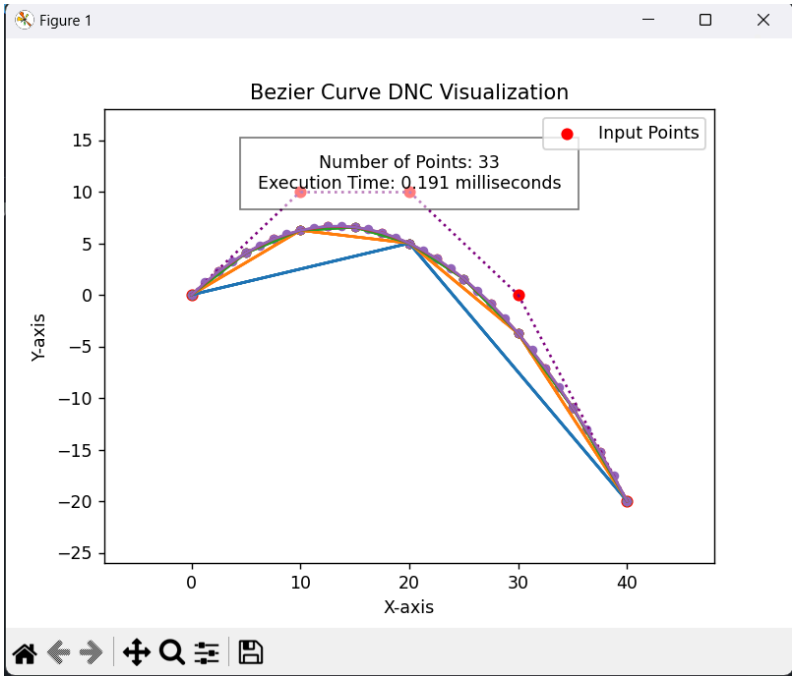
BAB 6

Analisis

Berdasarkan hasil uji test case antara algoritma brute force dengan divide and conquer, pada jumlah titik dan iterasi yang sama, diperoleh hasil yang akurat pada algoritma divide and conquer. Namun, algoritma brute force memberikan hasil yang lebih cepat, tetapi output yang dihasilkan tidak semulus pada algoritma divide and conquer. Sebagai perbandingan, akan digunakan test case satu pada masing-masing algoritma:



Gambar 6.1 Hasil dari Brute Force



Gambar 6.2 Hasil dari Divide and Conquer

Dengan jumlah titik kontrol dan iterasi yang sama, algoritma divide and conquer menghasilkan titik bezier jauh lebih banyak daripada brute force. Jika kita melakukan analisis kompleksitas, akan diperoleh kesimpulan:

1. **Metode Brute Force**

   Algoritma ini menghitung setiap titik pada kurva dengan mencoba semua kombinasi nilai parameter. Kompleksitas waktu untuk metode ini adalah:

   $$O(n.\ k)$$

   Dengan n adalah jumlah titik kontrol dan k adalah jumlah iterasi, algoritma ini nantinya akan menghasilkan sebanyak k + 1 titik bezier.

2. **Metode Divide and Conquer**

   Algoritma ini memecah daftar titik kontrol menjadi dua kali dari semula (menggunakan titik tengah) dan melakukan ini secara rekursif sampai mencapai jumlah iterasi yang ditentukan. Kompleksitas waktu untuk metode ini adalah

$$O(n \cdot 2^k)$$

Dengan n adalah jumlah titik kontrol dan k adalah jumlah iterasi. 2^n ini disebabkan karena tiap iterasi menghasilkan dua sub-masalah baru yang berukuran setengah dari ukuran asli dan n. 2^k berasal dari jumlah iterasi yang dilakukan. Algoritma ini akan menghasilkan 2^k + 1 titik.

Berdasarkan hasil analisis diatas, dapat disimpulkan bahwa algoritma brute force jauh lebih sangkil dibandingkan algoritma divide and conquer jika ditinjau dari kompleksitas waktu. Hal ini bisa saja terjadi karena algoritma divide and conquer tidak menjamin selalu menghasilkan solusi yang efisien. Selain itu, algoritma divide and conquer juga menghasilkan titik bezier yang jauh lebih banyak dibandingkan algoritma brute force sehingga jika diberikan titik kontrol dan jumlah iterasi yang sama, algoritma divide and conquer menghasilkan titik yang lebih banyak sehingga kurva yang dihasilkan jauh lebih mulus dibandingkan algoritma brute force.

BAB 7

Penjelasan Bonus

7.1 Algoritma Divide and Conquer dengan n titik kontrol

Algoritma Divide and Conquer yang digunakan salah satu metode untuk menghasilkan kurva Bezier dari titik kontrol. Algoritma ini bekerja dengan membagi masalah menjadi submasalah yang lebih kecil, menyelesaikan submasalah tersebut, dan kemudian menggabungkan solusi untuk mendapatkan solusi akhir dengan menghasilkan titik-titik kurva Bezier dari titik kontrol yang diberikan.

Langkah-langkahnya dapat dijelaskan sebagai berikut:

1. Inisialisasi
   Algoritma dimulai dengan menerima titik-titik kontrol sebagai input. Titik-titik kontrol ini adalah titik-titik yang akan membentuk kurva Bezier.
2. Pembagian (Divide)
   Langkah selanjutnya adalah membagi sekumpulan titik kontrol menjadi dua subset, yaitu titik-titik kiri dan titik-titik kanan. Pembagian ini dilakukan dengan mengambil titik kontrol tengah sebagai pemisah. Cara kerjanya adalah dengan menentukan titik tengah dari setiap titik tengah dari titik kontrol hingga menyisakan titik tengah tunggal sebagai pemisah, dalam prosesnya sekaligus menginisialisasi titik kontrol yang baru untuk disiapkan dalam iterasi berikutnya.
   Setelah pembagian, algoritma secara rekursif memanggil dirinya sendiri untuk menghitung titik-titik kurva Bezier dari subset kiri dan kanan yang dihasilkan dari pembagian sebelumnya.
   Proses rekursi ini terus berlanjut hingga mencapai basis dari rekursif yaitu ketika iterasi sekarang sudah mencapai iterasi yang ditentukan oleh user.
3. Penggabungan (Combine)
   Tahap terakhir adalah menggabungkan hasil dari subset kiri dan kanan yang sudah dihitung titik-titik kurva Beziernya. Penggabungan ini dilakukan dengan mempertimbangkan urutan dan proporsi dari masing-masing titik kontrol pada subset kiri dan kanan.

Dengan menggabungkan kedua subset, kita mendapatkan kurva Bezier secara keseluruhan yang dihasilkan dari sekumpulan n titik kontrol yang diberikan.

7.2 Proses visualisasi pembuatan kurva tiap iterasi

Proses visualisasi dilakukan dengan menggunakan matplotlib untuk menggambar kurva Bezier dan titik-titik kontrolnya. Proses ini dilakukan dalam beberapa iterasi, di mana setiap iterasi menambahkan lebih banyak titik ke kurva Bezier, sehingga kurva semakin halus.

Langkah-langkah visualisasi dapat dijelaskan sebagai berikut:

1. Inisialisasi Plot
   Awalnya, plot kosong dibuat dengan margin yang sesuai dengan titik-titik kontrol yang diberikan agar grafik bisa terlihat.
2. Iterasi
   Algoritma memperbarui plot untuk setiap iterasi. Setiap iterasi menambahkan lebih banyak titik ke kurva Bezier, yang secara bertahap menggambarkan pembentukan kurva secara keseluruhan.
3. Animasi
   Proses visualisasi diatur sebagai animasi setiap titik di mana plot diperbarui untuk setiap iterasi.
4. Informasi Tambahan
   Selain kurva Bezier dan titik kontrolnya, informasi tambahan seperti jumlah titik pada kurva dan waktu eksekusi juga ditampilkan pada plot.

Dengan visualisasi ini, user dapat melihat secara langsung bagaimana algoritma bekerja untuk membuat kurva Bezier dari titik kontrol di setiap iterasi, serta mengamati perubahan kurva setiap kali ada penambahan titik baru. Hal ini memudahkan untuk memahami dan menganalisis kinerja algoritma secara visual.

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.	✓	

Daftar Pustaka

Munir, Rinaldi. "Algoritma Brute Force - Bagian 1." Institut Teknologi Bandung, 2022

Algoritma-Brute-Force-(2022)-Bag1.pdf (itb.ac.id)

Munir, Rinaldi. "Algoritma Divide and Conquer- Bagian 1." Institut Teknologi Bandung, 2024

Algoritma-Divide-and-Conquer-(2024)-Bagian1.pdf (itb.ac.id)

Link Repository GitHub

AlthariqFairuz/Tucil2_13522027_13522057 (github.com)