LAPORAN TUGAS KECIL 2
IF2211 STRATEGI ALGORITMA

Membangun Kurva Bézier dengan Algoritma Titik Tengah berbasis Divide and Conquer

Disusun untuk memenuhi tugas mata kuliah Strategi Algoritma pada Semester 2 (dua) Tahun Akademik 2023/2024.

Disusun oleh:

Melati Anggraini	13522035
Novelya Putri Ramadhani	13522096

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

2024

DAFTAR ISI

DAFTAR ISI	1
BAB I DESKRIPSI PERSOALAN	2
BAB II TEORI SINGKAT	3
BAB III APLIKASI DIVIDE AND CONQUER, BRUTE FORCE, DAN ANALISIS	4
BAB IV SOURCE CODE	10
BAB V PENGUJIAN	13
LAMPIRAN	20
DAFTAR PUSTAKA	21

BAB I DESKRIPSI PERSOALAN

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.

Meninjau dari jumlah titik kontrolnya kurva Bézier dibedakan menjadi 3 jenis kurva diantaranya kurva Bézier linear (2 titik kontrol), kurva Bézier kubik (3 titik kontrol), kurva Bézier kuartik (4 titik kontrol), dan seterusnya. Pembentukan kurva bisa kita dapatkan dengan menggunakan perumusan untuk kurva Bézier kuadratik, kubik, dan kuartik seperti berikut :

$$kuadratik \text{ -> } R_0 = B(t) = (1-t)^2 P_0 + (1-t)tP_1 + t^2 P_2$$

$$kubik \text{ -> } S_0 = B(t) = (1-t)^3 P_0 + 3(1-t)^2 tP_1 + 3(1-t)t^2 P_2 + t^3 P_3$$

$$kuartik \text{ -> } T_0 = B(t) = (1-t)^4 P_0 + 4(1-t)^3 tP_1 + 6(1-t)^2 t^2 P_2 + 4(1-t)t^3 P_3 + t^4 P_4$$

Dengan $P_0 \ldots P_n$ merupakan titik kontrol dan $t \in [0, 1]$ sebagai rentang variasi nilai dalam pembentukan kurva.

Semakin banyak titik kontrolnya maka persamaan ini akan semakin panjang dan rumit. Oleh karena itu, penugasan kecil 2 kali ini bertujuan untuk menemukan keefisienan dalam pembuatan kurva Bézier dengan menggunakan algoritma *divide and conquer* dan algoritma *brute force* sebagai pembandingnya.

BAB II TEORI SINGKAT

Menurut Haryono [1] kurva Bezier adalah suatu kurva yang halus, kurva Bezier terdapat $n + 1$ titik untuk mengkonstruksikannya, titik tersebut yaitu titik $P_0, P_1, .., P_n$.Kurva Bezier dimulai dari titik P_0 yang merupakan titik awal dan kemudian melengkung ke arah titik P_n yang merupakan titik akhir dari kurva Bezier. Menurut Kusno [2] Persamaan kurva Bezier yang memiliki derajat n disajikan pada persamaan 1 berikut.

$$C(t) = \sum_{i=0}^{n} \mathrm{P_i}\, B_i^n(t), 0 \leq t \leq 1 \quad (1)$$

Dengan:

$$B_i^n(t) = C_i^n (1-t)^{n-i}.t^i \quad (2)$$

$$C_i^n = \frac{n!}{i!\,(n-i)!}$$

$\mathrm{P_i}$ = Koefisien geometri atau titik kontrol $C(t)$ [3]

Kurva Bezier dimulai pada titik P_0 dan berakhir di titik P_n, tetapi belum tentu melewati titik-titik kontrolnya. Untuk menggambarkan titik-titik pada kurva Bezier maka dilakukan perhitungan titik dari P_0ampai dengan P_n dengan t dari 0 sampai dengan 1.

Divide and Conquer secara umum terbagi menjadi 3 fase yakni membagi masalah kedalam sub-sub masalah yang lebih kecil, conquer yakni menyelesaikan sub-sub masalah secara rekursif, dan combine menggabungkan hasil dari penyelesaian sub-sub masalah menjadi penyelesaian yang dikehendaki. Ada 4 hal yang harus diperhatikan dalam strategi *Divide and Conquer* yaitu *branching factor* (jumlah problem minimal 2), *balance*(dapat dibagi dalam ukuran yang sama), *data dependence of divide function*(saling ketergantungan, rekursif), dan *control parallelism or sequentiality*(berurutan).

BAB III

APLIKASI DIVIDE AND CONQUER, BRUTE FORCE, DAN ANALISIS

A. Cara Menggunakan Program

Komponen yang menjadi masukan yaitu :

1. n titik kontrol, dengan $n >= 2$.
2. titik -titik kontrol sebanyak n pasang
3. Jumlah iterasi yang ingin dilakukan (m).

Pengguna memiliki 2 input masukan, yakni menggunakan input keyboard(terminal) atau memakai file.txt :

a. Jika menggunakan file.txt maka tambahkan file di folder test dengan format masukan :
   number of control_points n
   number of iterations
   algorithm to use (1 = brute force, 2 = divide and conquer)
   coordinate of n control_points
   Serta format filenya *nama_input_test.txt*
b. Jika menggunakan menggunakan keyboard, input sesuai perintah yang ditampilkan program. Khusus untuk format koordinat
   Enter the x and y coordinates of point: coordinate_1 coordinate_2

B. Analisis Kurva Bézier Menggunakan Algoritma Divide and Conquer

Ide dasar pembuatan kurva Bézier adalah dengan membagi kurva menjadi segmen-segmen yang lebih kecil dengan menggunakan titik-titik kontrol yang ada. Berikut penjelasan ide algoritma pembuatan kurva Bézier kuadratik menggunakan divide and conquer:

1. Masukkan 3 koordinat titik kontrol.
2. a. DIVIDE: Dari tiga titik kontrol awal, ambil titik tengah dari titik kontrol pertama dan kedua, dan titik tengah dari titik kontrol kedua dan ketiga sehingga didapatkan dua titik baru. Selanjutnya, titik tengah dari kedua titik baru ini diambil untuk dihubungkan dengan titik kontrol pertama dan ketiga. Hal ini dilakukan pada setiap iterasi, yaitu temukan titik tengah sebagai titik antara diantara pasangan-pasangan titik kontrol yang dimasukkan dan hubungkan titik-titik baru menjadi garis-garis baru. Kemudian ambil titik tengah dari masing-masing garis baru tersebut untuk dijadikan titik kontrol untuk iterasi selanjutnya.
   b. CONQUER: Secara rekursif ambil titik-titik tengah antara titik kontrol ataupun titik antara lalu masukkan ke dalam array of titik.

c. COMBINE: Setelah semua segmen dikonstruksi, algoritma menggabungkan semua titik yang dihasilkan dari setiap segmen untuk membentuk kurva Bézier secara keseluruhan.

d. REKURSIF:

- Algoritma menggunakan pendekatan rekursif untuk melakukan pembagian untuk setiap segmen kecil.
- Iterasi dilakukan hingga mencapai kondisi berhenti (jumlah iterasi yang ditentukan sebelumnya).

Algoritma juga dapat digeneralisasi sehingga program dapat membentuk kurva Bézier kubik, kuartik, dan selanjutnya dengan 4, 5, 6, hingga n titik kontrol. Berikut penjelasan ide algoritma pembuatan kurva Bézier menggunakan divide and conquer dengan n titik kontrol:

1. Jika jumlah titik kontrol yang dimasukkan kurang dari 3 ($n < 3$), maka jika $n = 1$ akan menampilkan pesan kesalahan karena hanya akan menghasilkan sebuah titik. (SOLVE), Jika $n = 2$ akan membentuk kurva bezier linier.
2. Jika jumlah titik kontrol lebih dari 3 ($n >= 3$):
    a. DIVIDE : Pada setiap iterasi, temukan titik tengah sebagai titik antara di antara pasangan-pasangan titik kontrol yang dimasukkan. Hubungkan titik-titik baru menjadi garis-garis baru. Kemudian ambil titik tengah dari masing-masing garis baru tersebut untuk dijadikan titik kontrol untuk iterasi selanjutnya.
    b. CONQUER : Secara rekursif ambil titik-titik tengah antara titik kontrol ataupun titik antara lalu masukkan ke dalam array of titik.
    c. COMBINE : Setelah semua segmen dikonstruksi, algoritma menggabungkan semua titik yang dihasilkan dari setiap segmen untuk membentuk kurva Bézier secara keseluruhan.
    d. REKURSIF :
        - Algoritma menggunakan pendekatan rekursif untuk melakukan pembagian untuk setiap segmen kecil.
        - Iterasi dilakukan hingga mencapai kondisi berhenti (jumlah iterasi yang ditentukan sebelumnya).

Kompleksitas Algoritma Divide and Conquer dalam Pembuatan Kurva Bézier:

1. Fungsi `midpoint(point1,point2)`: melakukan operasi sederhana penambahan dan pembagian untuk mencari titik tengah. Kompleksitas waktunya $O(1)$.

2. Fungsi `insert_midpoints(arr, midpoints)` : melakukan iterasi sepanjang array arr sebanyak $n - 1$ kali, dengan n panjang array. Kompleksitas waktu $O(n)$.
3. Fungsi `bezier(control_points, all_points, iterations)` : Iterasi sebanyak masukan m iterasi. Setiap iterasi dilakukan :
    a. Perhitungan titik-titik tengah antara pasangan titik kontrol
    b. Penyisipan titik-titik tersebut antara elemen-elemen array

Kompleksitas algoritma setiap iterasi $O(n)$, sehingga kompleksitasnya adalah $O(n * m)$ dengan kasus terburuk ketika $m >= n$,

$$Kompleksitas\ Total\ = O(1) + O(n) + O(n^2) = O(n^2)$$

Sedangkan, untuk $m < n$, $Kompleksitas\ Total\ = O(n)$.

Untuk meninjau **kompleksitas waktunya**, dapat digunakan teorema master dimana $a = 2,\ n/2\ = n/b,\ f(n) = O(1)\ \ ->\ \ T(n)\ = 2T(n/2) + O(1)$

$$= O(n \log n)$$

C. Analisis Kurva Bézier Menggunakan Algoritma Brute Force Sebagai Pembanding

Pembuatan kurva Bézier dengan metode brute force dilakukan dengan melakukan pengulangan secara rekursif, dimana memproses setiap titik kontrol secara berurutan tanpa memanfaatkan struktur atau sifat khusus dari kurva Bézier untuk mempercepat hitungan. Rumus yang digunakan adalah

$$B(t) = \sum_{k=0}^{n} C_{(n,k)} (1-t)^k t^{n-k} P_k$$

Algoritma mengaplikasikan rumus di atas pada fungsi `bezier_curve_recursive()`, program akan melakukan iterasi untuk menentukan pointnya dengan num_points yang merepresentasikan banyak titik kurva yang diambil dari kurva rumus yang dibuat. Dimana num_points itu sendiri untuk kurva Bezier kuadratik $num_points\ = 2^{iterations} + 1$ diperoleh melalui $\sum_{k=0}^{n} C_{(n,k)}$. Sedangkan untuk perumumannya atau untuk n titik kontrol sendiri diperoleh dari pengamatan dan didapat num_points :

$$num_points\ = 2^{iterasi} + 1 + 2^{iterasi-1} x(jumlah\ titik\ kontrol\ - 1)$$

Kompleksitas algoritma :

1. Setiap rekursi akan mengurangi titik kontrol menjadi sebanyak setengahnya yang dilakukan sebelum tersisa hanya 1 titik saja. Kompleksitas algoritma nya $O(log_2 n)$, dimana n adalah jumlah titik kontrol.
2. Di dalam perulangan for dalam fungsi `bezier_curve_recursive()`, perhitungan dilakukan setiap pasangan titik kontrol. Oleh karena itu, ada $n - 1$ iterasi yang dalam setiap iterasinya mengandung operasi sederhana seperti penjumlahan dan perkalian. Oleh karena itu kompleksitas waktunya $O(n)$.
3. Perulangan dalam `plot_curve_bezier()` dilakukan sebanyak $O(num_points)$

Algoritma melakukan rekursi dalam setiap iterasi untuk mengurangi jumlah titik kontrol menjadi satu, jumlah total operasi yang dilakukan dalam rekursi :

$$Kompleksitas\ Total = O(log_2 n) + O(n) + O(num_points) = O(num_points)$$

Untuk meninjau **kompleksitas waktunya**, bisa menggunakan teorema master sehingga didapatkan kompleksitas waktu $T(n) = O(num_points)$, dengan $a = num_points + 1$ merupakan ukuran submasalah yang dipanggil fungsi, $b = n/b$ merupakan ukuran setiap submasalah, yang dalam hal ini n itu sendiri karena tidak ada pembagian ukuran masalah (atau hanya menggunakan pembagian di titik sebanyak num_points). $f(n) = O(1)$ untuk setiap iterasi yang terjadi dalam loop list. Maka, didapatkan

$$T(n) = (num_point + 1).O(1) = O(num_points)$$

D. Analisis Penggunaan Algoritma Divide and Conquer dengan Algoritma Brute Force Dalam Pembuatan Kurva Bézier

No.	(jumlah titik kontrol, jumlah iterasi)	Waktu eksekusi dengan Brute Force (dalam ms)	Waktu eksekusi dengan DnC (dalam ms)
1	(3,3)	265.32	265.06
2	(3,2)	276.82	265.24
3	(3,1)	278.24	266.04
4	(3,0)	289.79	267.21

5	(3,4)	304.64	271.16
6	(3,5)	269.45	279.69
7	(3,7)	273.33	280.39
8	(3,8)	280.30	278.59
9	(3,10)	264.98	263.09
10	(3,15)	404.83	731.59

Tabel 3.1 Perbandingan waktu eksekusi kurva Bezier kuadratik

Berdasarkan tabel diatas, algoritma DnC akan memberikan waktu eksekusi yang cenderung lebih cepat daripada algoritma brute force ketika jumlah iterasi < 10. Hal ini bisa terjadi karena DnC akan membagi-bagi submasalah menjadi 2 bagian sedangkan brute force akan melakukan pemanggilan rekursif sebanyak *num_points*. Akan tetapi, untuk jumlah iterasi yang lebih besar, penggunaan brute force lebih efektif karena kenaikan waktu eksekusinya tidak eksponensial. Dalam pembentukan kurva kuadratik, $iterasi > 5$ titik kontrol sudah akan menghasilkan kurva yang cukup mulus. Oleh karena itu, dapat disimpulkan bahwa penggunaan DnC lebih sangkil dalam implementasi kurva Bezier kuadratik.

No.	(jumlah titik kontrol, jumlah iterasi)	Waktu eksekusi dengan Brute Force (dalam ms)	Waktu eksekusi dengan DnC (dalam ms)
1	(5,3)	279.37	265.03
2	(5,5)	292,97	286,38
3	(2,2)	277.72	270.61
4	(10,10)	311,49	342,61
5	(15, 17)	72327.60	3488812.61
6	(17,3)	187.44	141.86
7	(10, 7)	145.72	171.50

Tabel 3.1 Perbandingan waktu eksekusi kurva Bezier n titik kontrol

Berdasarkan tabel diatas, terlihat untuk jumlah titik kontrol yang semakin membesar dan iterasi yang semakin membesar maka pembuatan kurva bezier dengan menggunakan brute force akan lebih cepat dalam atau sesuai dengan perhitungan kompleksitas waktu diatas. Akan tetapi, untuk hasil dari kurvanya sendiri kurang sesuai (bisa dilihat di tabel 5.1), hal ini berkebalikan dengan algoritma DnC yang membutuhkan waktu lama tetapi kurva yang dihasilkan lebih sesuai. Hal ini bisa terjadi dikarenakan resolusi kurva dalam algoritma brute force ditentukan oleh `num_points` yang menghitung jumlah titik di sepanjang kurva yang dihitung. Jika `num_points` dalam brute force tidak cukup tinggi, kurva mungkin tidak tergambar sehalus yang dihasilkan oleh divide and conquer. Sedangkan, dalam metode divide and conquer, kurva disempurnakan melalui pembagian berulang-ulang. Pada setiap iterasi, titik kontrol dari kurva dibagi menjadi bagian-bagian yang lebih kecil, menciptakan satu set titik kontrol baru yang lebih mendekati jalur sebenarnya dari kurva. Metode ini menghasilkan kurva yang tidak bergantung pada jumlah nilai t yang tetap, tetapi menyempurnakan kurva melalui proses subdivisi iteratif. Oleh karena itu, dapat disimpulkan bahwa penggunaan DnC lebih sangkil dalam implementasi pembuatan kurva Bezier.

BAB IV
SOURCE CODE

4.1 Bezier Kurva dengan Algoritma Brute Force

```python
def bezier_curve_recursive(control_points, t):
    """ Recursively calculates the Bezier curve point at a specific t """
    if len(control_points) == 1:
        return control_points[0]
    new_points = []
    for i in range(len(control_points) - 1):
        new_x = control_points[i][0] * (1 - t) + control_points[i + 1][0] * t
        new_y = control_points[i][1] * (1 - t) + control_points[i + 1][1] * t
        new_points.append((new_x, new_y))
    return bezier_curve_recursive(new_points, t)
```

4.2 Bezier Kurva dengan Algoritma Divide and Conquer

```python
def bezier(control_points, all_points, iterations):
    """
    Constructs the Bezier curve points using the midpoint algorithm for the given iteration
    """
    if iterations == 0:
        return all_points
    else:
        # Calculate the midpoints
        mid1 = []
        for i in range(1, len(control_points)):
            mid1.append(midpoint(control_points[i-1], control_points[i]))
        all_points.append(mid1)
        # Calculate the midpoints of the midpoints
        mid2 = []
        for i in range(1, len(mid1)):
            mid2.append(midpoint(mid1[i-1], mid1[i]))
        # If it's already the last iteration, add the curve's points to the list and return
        if iterations == 1:
            curve = []
            curve.append(control_points[0])
            curve.extend(mid2)
            curve.append(control_points[-1])
            all_points.append(curve)
            return all_points
        # Insert both midpoints lists as new control points for the next iteration
        new_control_points = []
        new_control_points.append(control_points[0])
        new_control_points.extend(insert_midpoints(mid1, mid2))
        new_control_points.append(control_points[-1])
        # Recursively call the function for the next iteration
        return bezier(new_control_points, all_points, iterations - 1)
```

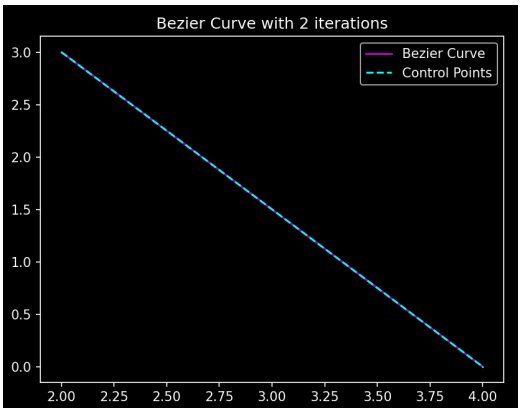
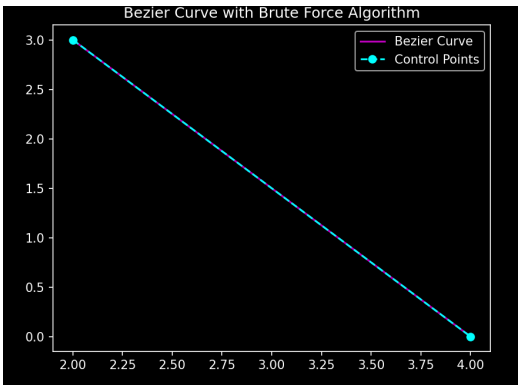
```python
def midpoint(point1, point2):
    """
    Returns the midpoint between two points
    """
    return (point1[0] + point2[0]) / 2, (point1[1] + point2[1]) / 2

def insert_midpoints(arr, midpoints):
    """
    Inserts midpoints into in between the elements of the array
    """
    for i in range(len(arr) - 1, 0, -1):
        arr.insert(i, midpoints[i - 1])
    return arr
```

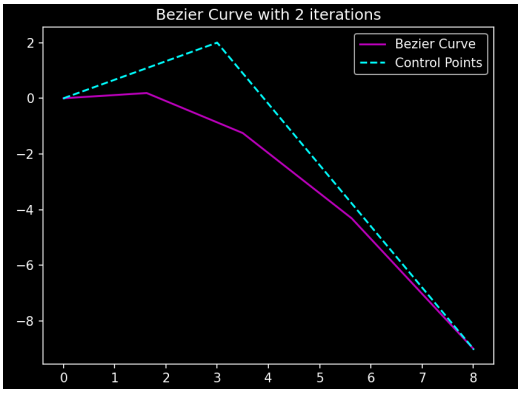
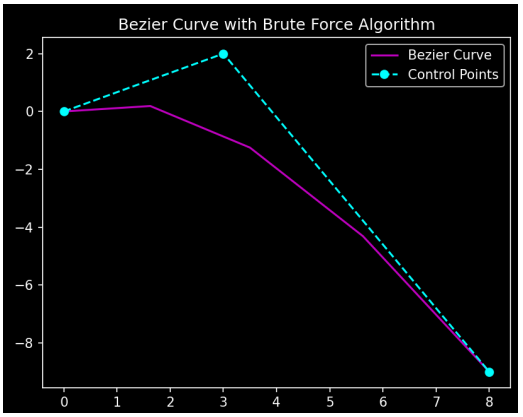
4.3 main.py

```python
def main():
    display_welcome_message()
    # Select the input method
    print("Select the input method:")
    print("1. Manual input")
    print("2. Read from a file")
    input_method = int(input("Enter the number of the input method: "))
    if input_method == 1:
        # Take input from the user for the control points
        control_points = []
        n = int(input("\nEnter the number of control points: "))
        if n < 2:
            print("At least 2 control points are required to form a Bezier curve.")
            return
        for i in range(n):
            x, y = map(float, input(f"Enter the x and y coordinates of point {i + 1}: ").split())
            control_points.append((x, y))
        # Take input from the user for the number of iterations
        iterations = int(input("Enter the number of iterations: "))
        if iterations < 0:
            print("At least 0 iteration is required.")
            return
        # Input selection for the algorithm to use
        print("\nSelect the algorithm to use:")
        print("1. Brute Force")
        print("2. Divide and Conquer")
        algorithm = int(input("Enter the number of the algorithm: "))
    elif input_method == 2:
        file_path = input("\nEnter the file path: ")
        file_path = "../test/" + file_path
        n, iterations, algorithm, control_points = read_input_from_txt(file_path)
    else:
        print("Invalid input. Please enter 1 or 2.")
        return

    if algorithm == 1:
        brute.plot_bezier_brute_force(control_points, iterations)
    elif algorithm == 2:
        print("\nDo you want to animate the curve? (y/n)")
        animate = input().lower()
        if animate == 'y':
            dac.plot_bezier_divide_and_conquer_animate(control_points, iterations)
        else:
            dac.plot_bezier_divide_and_conquer(control_points, iterations)
    else:
        print("Invalid input. Please enter 1 or 2.")
        return

    print("\nGoodbye! Hope you enjoyed the bezier curve 💕")
```

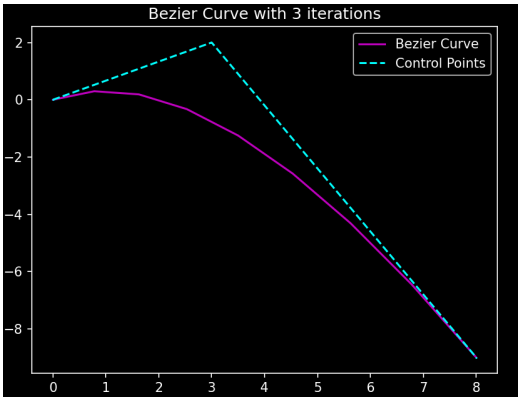
BAB V

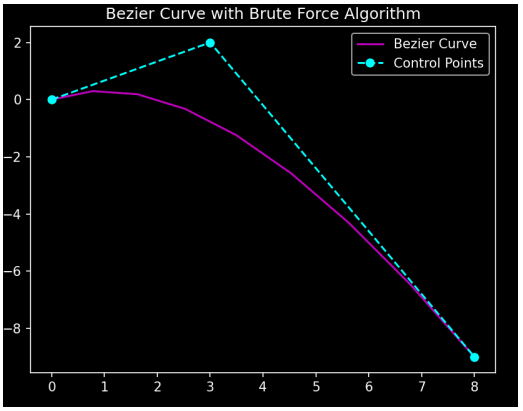
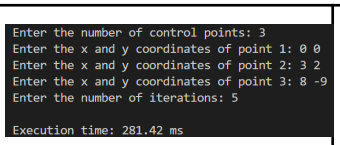
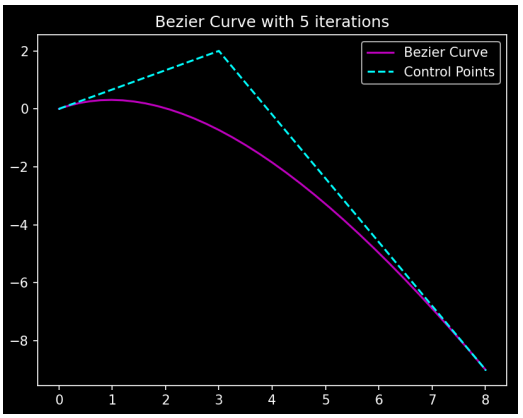
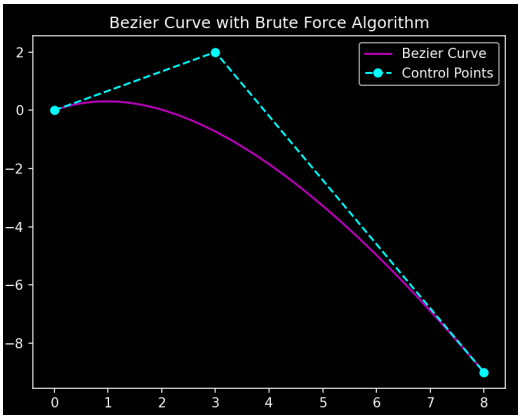
PENGUJIAN

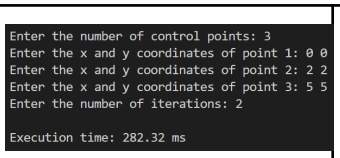
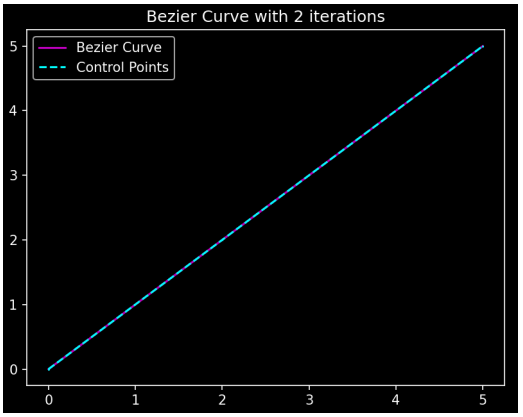
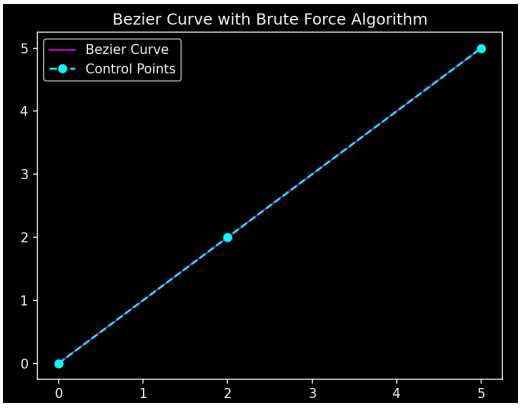
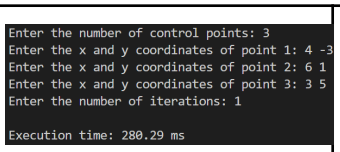
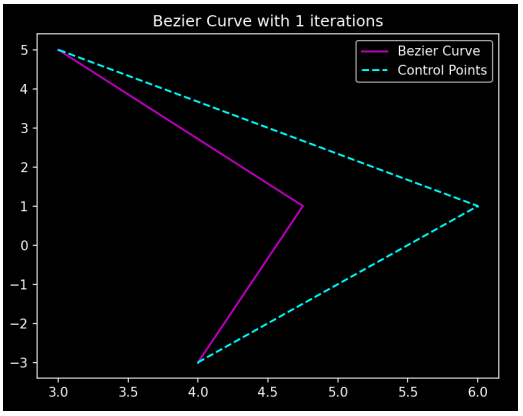
No	(jumlah titik kontrol(n), jumlah iterasi(m))	Titik kontrol	Output
1.	Jika titik kontrol $n < 3$		
	(2,2)	Enter the number of control points: 2 Enter the x and y coordinates of point 1: 2 3 Enter the x and y coordinates of point 2: 4 0 Enter the number of iterations: 2 Execution time: 273.31 ms	 
	(1,2)	Enter the number of control points: 1 At least 2 control points are required to form a Bezier curve.	-
2.	Jika titik kontrol $n >= 3$ dan jumlah iterasi(m) < jumlah titik kontrol(n)		

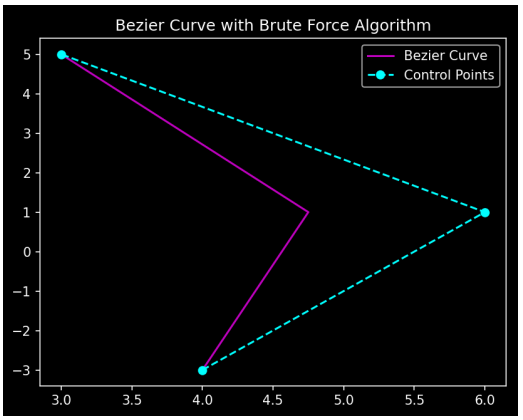
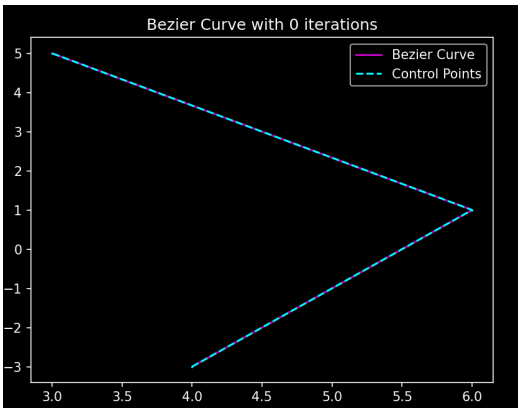
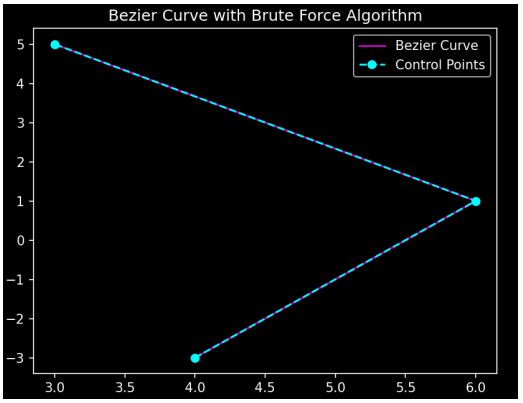
	(3,2)	`Enter the number of control points: 3`
		`Enter the x and y coordinates of point 1: 0 0`
		`Enter the x and y coordinates of point 2: 3 2`
		`Enter the x and y coordinates of point 3: 8 -9`
		`Enter the number of iterations: 2`
		
		`Execution time: 278.38 ms`
			
			 

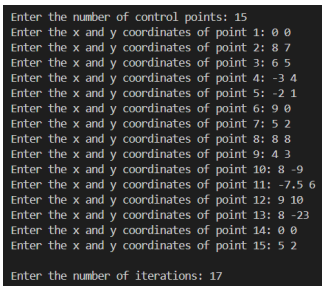
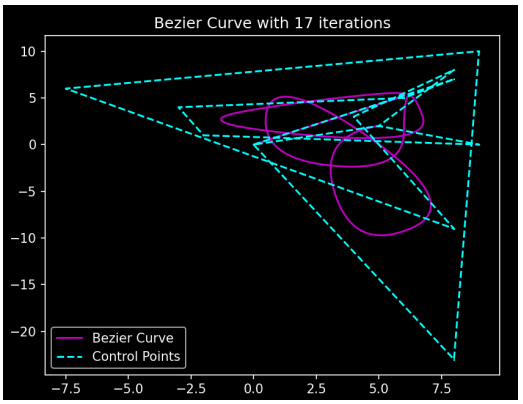
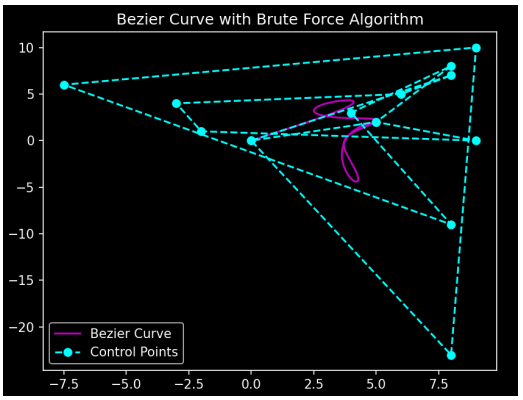
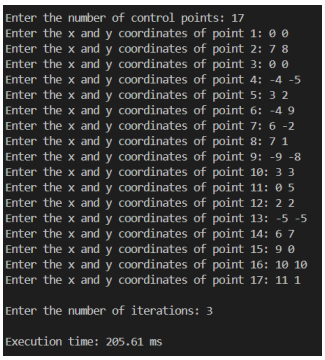
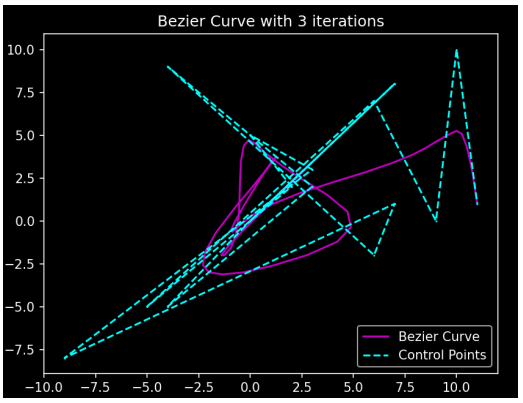
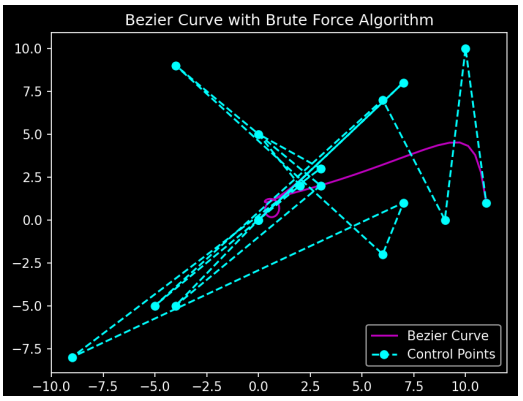
3.	Jika jumlah titik kontrol sama dengan jumlah iterasi $n = m$		

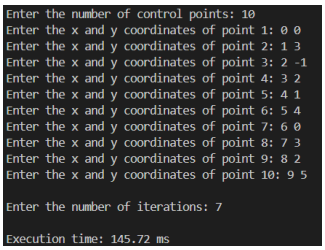
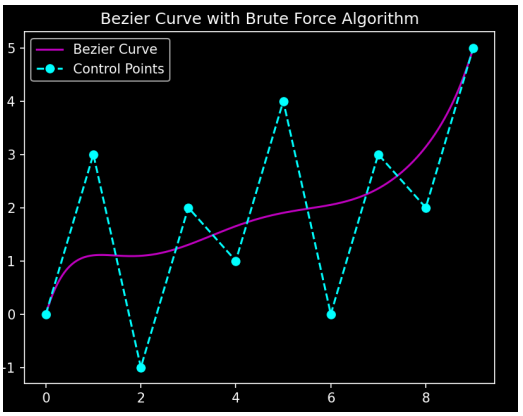
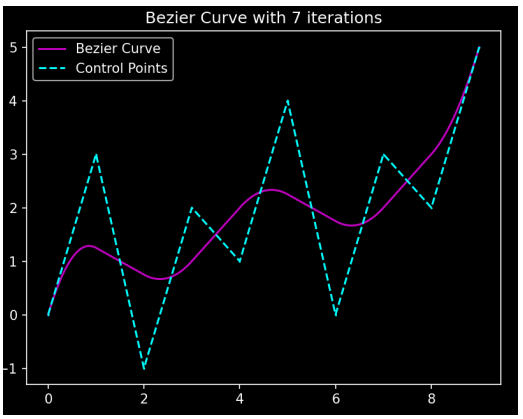
	(3,3)	`Enter the number of control points: 3`
		`Enter the x and y coordinates of point 1: 0 0`
		`Enter the x and y coordinates of point 2: 3 2`
		`Enter the x and y coordinates of point 3: 8 -9`
		`Enter the number of iterations: 3`
		
		`Execution time: 290.88 ms`
			

			
4.	Jika jumlah iterasi lebih dari jumlah titik kontrol $m > n$		
	(3,5)		 
5.	Jika titik-titik kontrol membentuk garis lurus		

	(3,2)		 
6.	Jika iterasinya 1		
	(3,1)		

			
7.	Jika iterasinya 0		
	(3,0)	Enter the number of control points: 3 Enter the x and y coordinates of point 1: 4 -3 Enter the x and y coordinates of point 2: 6 1 Enter the x and y coordinates of point 3: 3 5 Enter the number of iterations: 0 Execution time: 286.43 ms	 
8.	Untuk n kontrol dan jumlah iterasi yang nilainya lebih banyak		

9	(15,17)	 Enter the number of control points: 15 Enter the x and y coordinates of point 1: 0 0 Enter the x and y coordinates of point 2: 8 7 Enter the x and y coordinates of point 3: 6 5 Enter the x and y coordinates of point 4: -3 4 Enter the x and y coordinates of point 5: -2 1 Enter the x and y coordinates of point 6: 9 0 Enter the x and y coordinates of point 7: 5 2 Enter the x and y coordinates of point 8: 8 8 Enter the x and y coordinates of point 9: 4 3 Enter the x and y coordinates of point 10: 8 -9 Enter the x and y coordinates of point 11: -7.5 6 Enter the x and y coordinates of point 12: 9 10 Enter the x and y coordinates of point 13: 8 -23 Enter the x and y coordinates of point 14: 0 0 Enter the x and y coordinates of point 15: 5 2 Enter the number of iterations: 17	 Bezier Curve with 17 iterations Bezier Curve Control Points  Bezier Curve with Brute Force Algorithm Bezier Curve Control Points
10	(17,3)	 Enter the number of control points: 17 Enter the x and y coordinates of point 1: 0 0 Enter the x and y coordinates of point 2: 7 8 Enter the x and y coordinates of point 3: 0 0 Enter the x and y coordinates of point 4: -4 -5 Enter the x and y coordinates of point 5: 3 2 Enter the x and y coordinates of point 6: -4 9 Enter the x and y coordinates of point 7: 6 -2 Enter the x and y coordinates of point 8: 7 1 Enter the x and y coordinates of point 9: -9 -8 Enter the x and y coordinates of point 10: 3 3 Enter the x and y coordinates of point 11: 0 5 Enter the x and y coordinates of point 12: 2 2 Enter the x and y coordinates of point 13: -5 -5 Enter the x and y coordinates of point 14: 6 7 Enter the x and y coordinates of point 15: 9 0 Enter the x and y coordinates of point 16: 10 10 Enter the x and y coordinates of point 17: 11 1 Enter the number of iterations: 3 Execution time: 205.61 ms	 Bezier Curve with 3 iterations Bezier Curve Control Points  Bezier Curve with Brute Force Algorithm Bezier Curve Control Points

11	(10,7)		 

Tabel 5.1 pengujian program

LAMPIRAN

Repository: https://github.com/novelxv/Tucil2_13522035_13522096

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

Komarullah, Hafif. (2023). MODELISASI LAMPION MELALUI TEKNIK PENGGABUNGAN BENTUK-BENTUK DASAR GEOMETRI. Diakses pada 19 Maret 2024 dari http://prosiding.himatikauny.org/index.php/prosidinglsm/article/view/311

Chicago.(2024). Modul_5_Output_Primitif_5_Kurva_Bezier-libre.pdf. Diakses dari pdf :https://www.academia.edu/

Irfan, Muhammad. (2016). Implementasi Algoritma Divide And Conquer Pada Aplikasi Belajar Ilmu Tajwid Diakses pada 19 Maret 2024 dari https://join.if.uinsgd.ac.id/

Munir, R. (2024). Strategi Algoritma #1 Algoritma Divide and Conquer (Bagian 1). Diakses dari Homepage Rinaldi Munir: https://informatika.stei.itb.ac.id/~rinaldi

Munir, R. (2024). Strategi Algoritma #1 Algoritma Divide and Conquer (Bagian 2). Diakses dari Homepage Rinaldi Munir: https://informatika.stei.itb.ac.id/~rinaldi

Munir, R. (2024). Strategi Algoritma #1 Algoritma Divide and Conquer (Bagian 3). Diakses dari Homepage Rinaldi Munir: https://informatika.stei.itb.ac.id/~rinaldi

Munir, R. (2024). Strategi Algoritma #1 Algoritma Divide and Conquer (Bagian 4). Diakses dari Homepage Rinaldi Munir: https://informatika.stei.itb.ac.id/~rinaldi