

LAPORAN TUGAS KECIL
IF2211 Strategi Algoritma
Membangun Kurva Bézier dengan Algoritma Titik
Tengah berbasis Divide and Conquer



Disusun oleh:

Angelica Kierra Ninta Gurning	(13522048)
Julian Chandra Sutadi	(13522080)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023

DAFTAR ISI

DAFTAR ISI.....	1
BAB I.....	2
 Deskripsi Persoalan.....	2
1.1. Penjelasan Bézier Curve.....	2
1.2. Penjelasan Algoritma Divide and Conquer.....	2
BAB II.....	3
 Deskripsi Algoritma.....	3
2.1. Analisis dan Implementasi Algoritma Brute Force.....	3
2.2. Analisis dan Implementasi Algoritma Divide and Conquer.....	3
BAB III.....	5
 Hasil Implementasi.....	5
3.1. Tangkapan Layar Algoritma Brute Force.....	5
3.2. Tangkapan Layar Algoritma Divide and Conquer.....	8
BAB IV.....	14
 Analisis Perbandingan Algoritma.....	14
4.1. Pembangkitan Kurva Bézier Kuadratik dengan Bruteforce.....	14
4.2. Pembangkitan Kurva Bézier Kuadratik dengan Divide and Conquer..	14
4.3. Perbandingan Algoritma Brute Force dan Divide and Conquer.....	14
BAB V.....	16
 Implementasi Bonus.....	16
5.1. Penjelasan Generalisasi Algoritma.....	16
5.2. Penjelasan Visualisasi Pembentukan Kurva Bézier.....	17
BAB VII.....	28
 Lampiran.....	28

BAB I

Deskripsi Persoalan

1.1. Penjelasan Bézier Curve

Kurva Bézier adalah kurva halus yang didefinisikan oleh beberapa titik kontrol. Suatu kurva bázier disebut memiliki orde n jika didefinisikan oleh sebanyak $n + 1$ titik. Kurva ini memiliki banyak aplikasi, misalnya pada *pen tools*, pembuatan animasi yang halus dan realistik, pembuatan font yang unik, dsb.

Jika dimiliki himpunan titik-titik kontrol sebanyak $n + 1$, yaitu P_0, P_1, \dots, P_n , maka P_1 sampai dengan $P(n-1)$ dapat disebut sebagai titik kontrol antara, sedangkan titik P_0 dan P_n menjadi titik-titik ujung kurva. Titik-titik pada himpunan titik kontrol ini akan menjadi masukkan bagi suatu fungsi parametris yang menggambarkan kurva. Secara umum, kurva berorde n dapat dibentuk dengan fungsi berikut

$$B(t) = \sum_{i=0}^n \binom{n}{i} (1 - t)^{n-i} t^i, \quad t \in [0, 1]$$

1.2. Penjelasan Algoritma Divide and Conquer

Algoritma Divide Conquer merupakan suatu algoritma yang memecah-mecah masalah besar menjadi masalah yang lebih kecil dan lebih mudah untuk diselesaikan. Terdapat 3 langkah utama dalam Divide and Conquer, yaitu Divide, Conquer, dan Combine.

1. Divide

Merupakan langkah untuk membagi persoalan menjadi beberapa sub persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil, idealnya setiap upa-persoalan berukuran hampir sama. Pembagian ini bisa dilakukan secara berulang (rekursif) hingga sub-masalah yang didapat cukup sederhana untuk diselesaikan secara langsung.

2. Conquer

Menyelesaikan sub-sub masalah yang sudah dibagi. Jika sub masalah masih berukuran besar, akan diterapkan algoritma Divide and Conquer secara rekursif.

3. Combine

Semua hasil dari sub masalah akan digabungkan untuk mendapat solusi akhir dari masalah utama.

BAB II

Deskripsi Algoritma

2.1. Analisis dan Implementasi Algoritma Brute Force

2.1.1 Penjelasan Strategi

Algoritma dengan pendekatan *brute force* digunakan sebagai pembanding terhadap algoritma *divide and conquer*. Oleh karena itu, untuk jumlah iterasi i , akan dicari sebanyak $2^i - 1$ titik pada kurva. Nilai dari tiap kurva akan didapatkan dengan menjadikan tiga titik kontrol dan satu buah parameter

$$t = \frac{i}{\text{banyak titik dicari} + 1}, i = 1, 2, \dots, \text{banyak titik dicari}$$

sebagai input.

2.1.2 Source Code Program



```
1  """-----FUNGSI BRUTE FORCE-----"""
2
3  def bezier_brute_force(initial_points, num_of_iterations):
4      num_created_points = 2 ** num_of_iterations - 1
5      length = num_created_points + 2
6      final_points = [(0, 0)] * length
7      final_points[0] = initial_points[0]
8      for i in range(num_created_points):
9          t = (i + 1) / (num_created_points + 1)
10         x = ((1 - t) ** 2) * initial_points[0][0] + ((1-t) * t) * initial_points[1][0] + (t ** 2) * initial_points[2][0]
11         y = ((1 - t) ** 2) * initial_points[0][1] + ((1-t) * t) * initial_points[1][1] + (t ** 2) * initial_points[2][1]
12         final_points[i+1] = (x,y)
13     final_points[length-1] = initial_points[2]
14
15  return final_points
```

2.2. Analisis dan Implementasi Algoritma Divide and Conquer

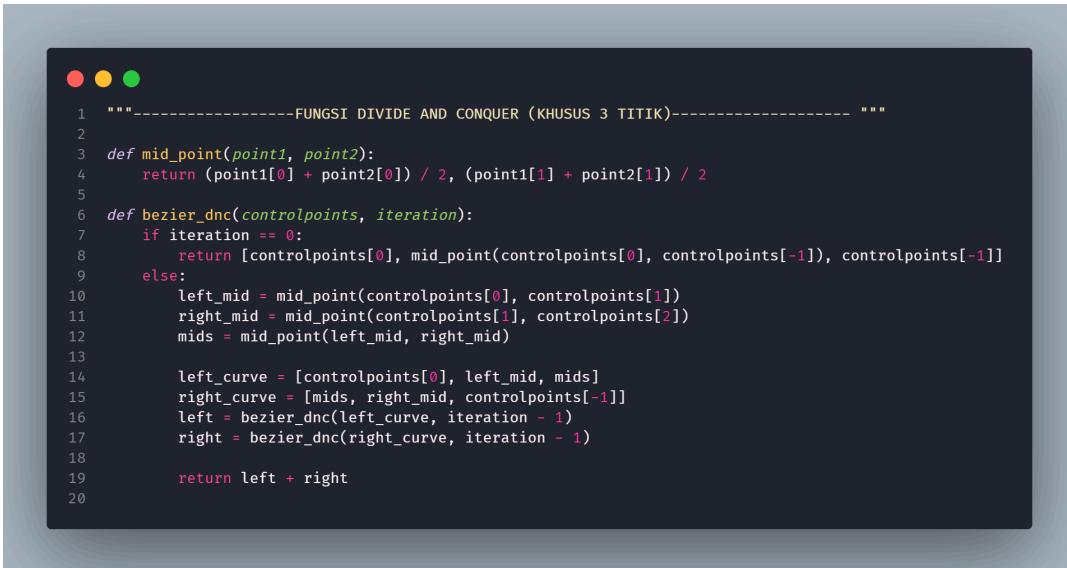
2.2.1 Penjelasan Strategi

Untuk menyelesaikan masalah pembentukan Bézier Curve dengan 3 titik kontrol, akan diterapkan strategi Divide and Conquer menggunakan *Subdivision/Mid-Point Algorithm*, dengan langkah-langkah sebagai berikut:

1. Fungsi akan memanggil strategi divide and conquer secara rekursif. Fungsi akan menerima 2 masukkan, yakni larik titik-titik kontrol (P_0, P_1, P_2) dan juga jumlah iterasi (i). Basis dari fungsi tersebut merupakan jika iterasi mencapai 0, maka fungsi akan mengembalikan titik-titik kurva bázier.
2. Pertama akan **dicari dua titik tengah** yaitu, titik tengah antara P_0, P_1 (Q_0) dan titik tengah antara P_1, P_2 (Q_1).

3. Setelah mendapat kan **Q0** dan **Q1** akan dicari titik tengah dari kedua titik tersebut, yaitu R0.
4. Setelah mendapat R0 titik kontrol yang baru terdiri dari P0,Q0,R0,Q1,P2. Titik kontrol ini akan **dipecah menjadi bagian kanan dan kiri**. Bagian kiri merupakan sebuah titik kontrol baru yaitu P0,Q0,R0. Bagian kanan merupakan titik kontrol R0,Q1,P2.
5. Setelah dibagi menjadi dua bagian, fungsi Divide and Conquer akan **dipanggil secara rekursif**
6. Setelah mendapatkan titik titik dari tiap bagian, fungsi akan **menggabungkan hasil-hasil titik** tersebut dan menghasilkan solusi
7. Solusi akhir merupakan titik-titik untuk membentuk kurva bézier yang nantikan akan di *plot* untuk visualisasi

2.2.2 Source Code Program



```

1  """-----FUNGSI DIVIDE AND CONQUER (KHUSUS 3 TITIK)----- """
2
3 def mid_point(point1, point2):
4     return (point1[0] + point2[0]) / 2, (point1[1] + point2[1]) / 2
5
6 def bezier_dnc(controlpoints, iteration):
7     if iteration == 0:
8         return [controlpoints[0], mid_point(controlpoints[0], controlpoints[-1]), controlpoints[-1]]
9     else:
10        left_mid = mid_point(controlpoints[0], controlpoints[1])
11        right_mid = mid_point(controlpoints[1], controlpoints[2])
12        mids = mid_point(left_mid, right_mid)
13
14        left_curve = [controlpoints[0], left_mid, mids]
15        right_curve = [mids, right_mid, controlpoints[-1]]
16        left = bezier_dnc(left_curve, iteration - 1)
17        right = bezier_dnc(right_curve, iteration - 1)
18
19    return left + right
20

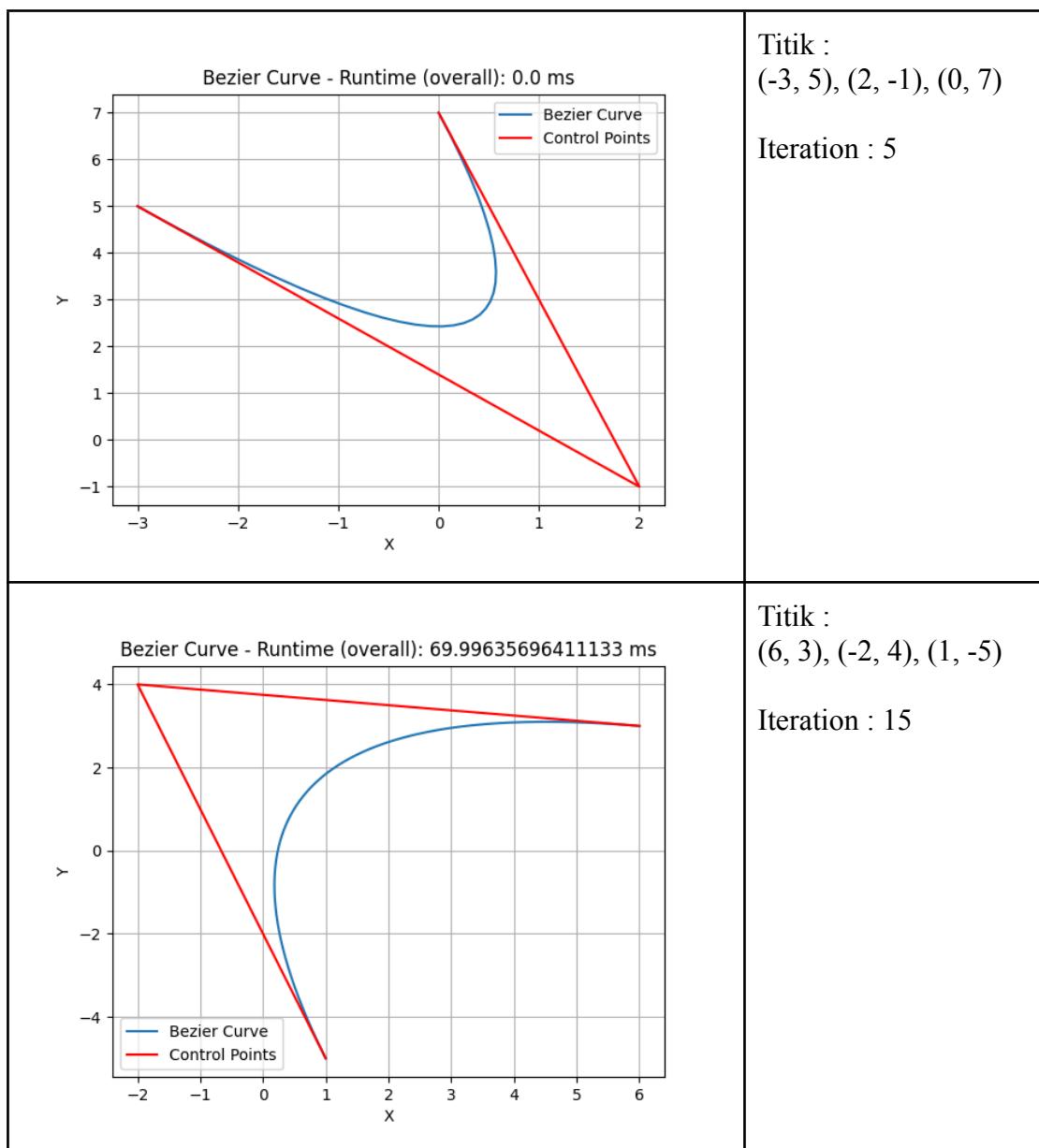
```

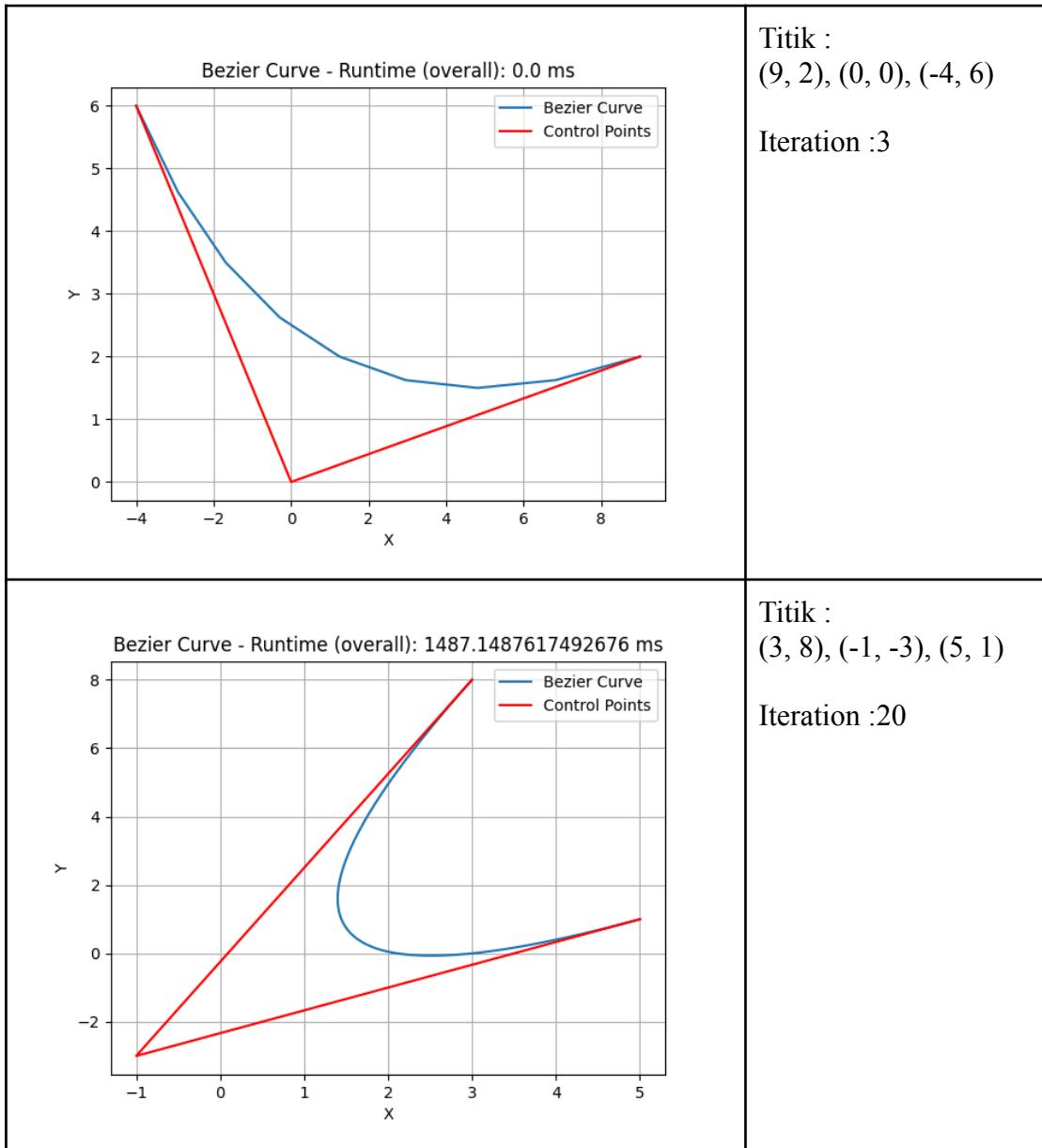
BAB III

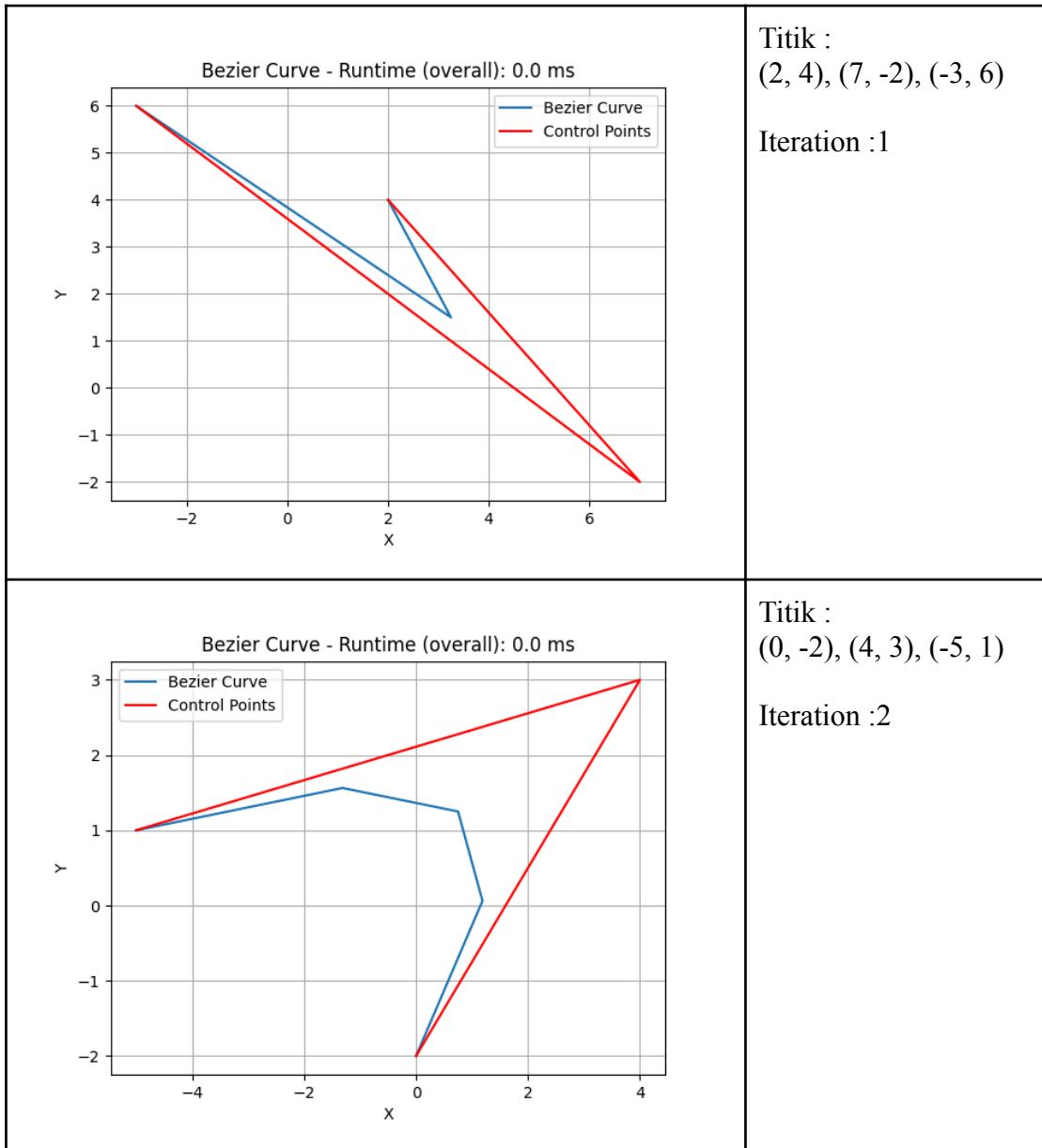
Hasil Implementasi

3.1. Tangkapan Layar Algoritma Brute Force

Tabel 1. Hasil implementasi menggunakan algoritma

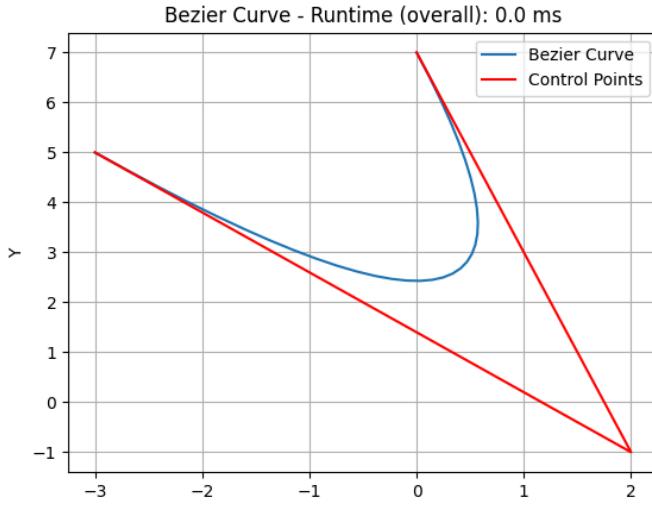
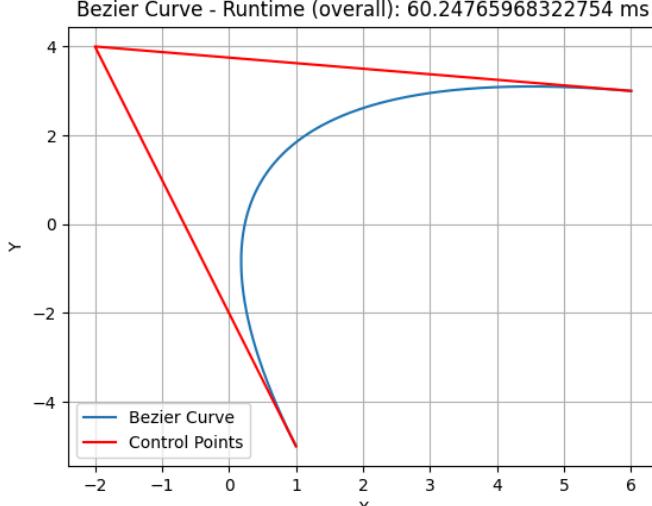


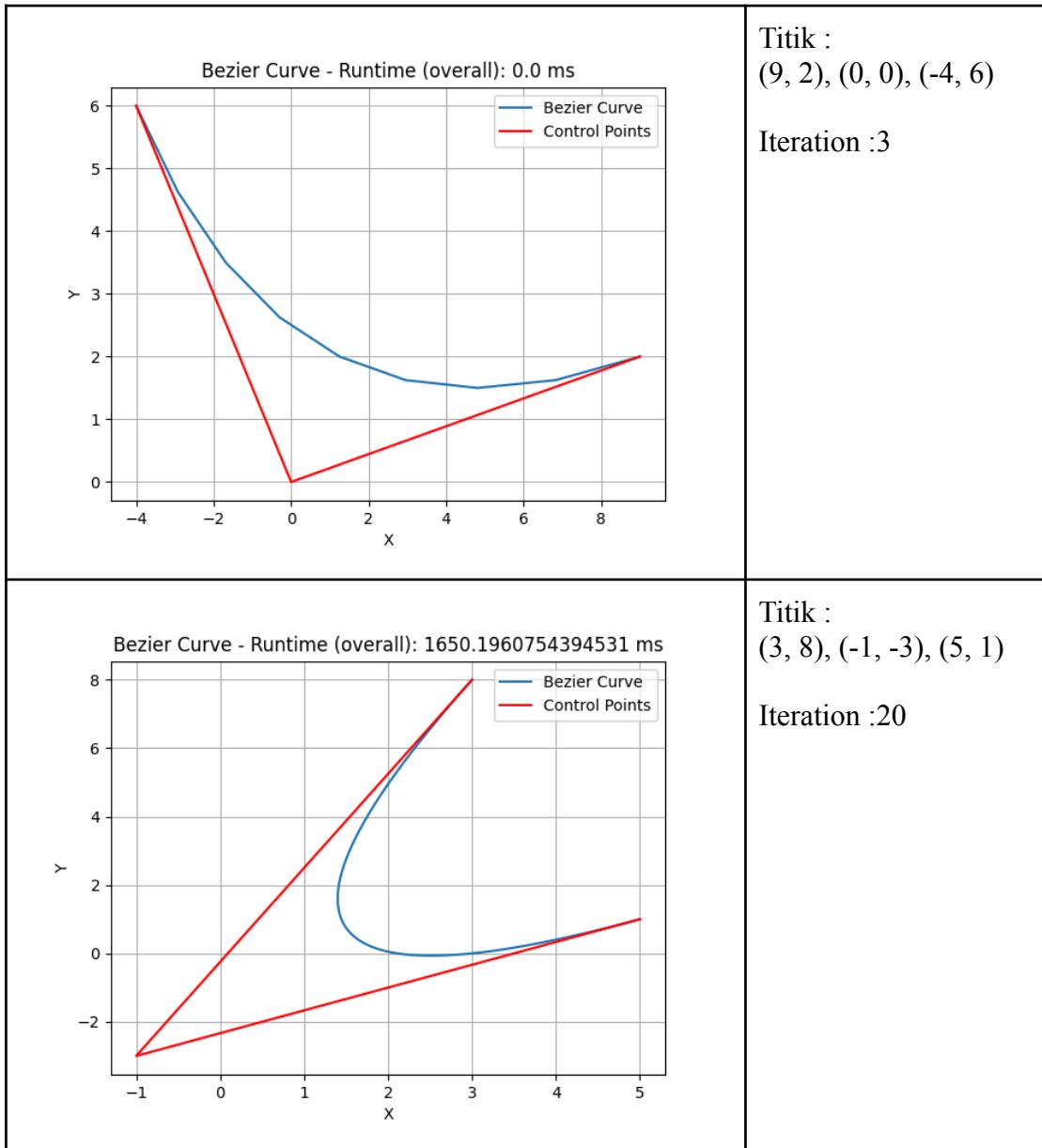


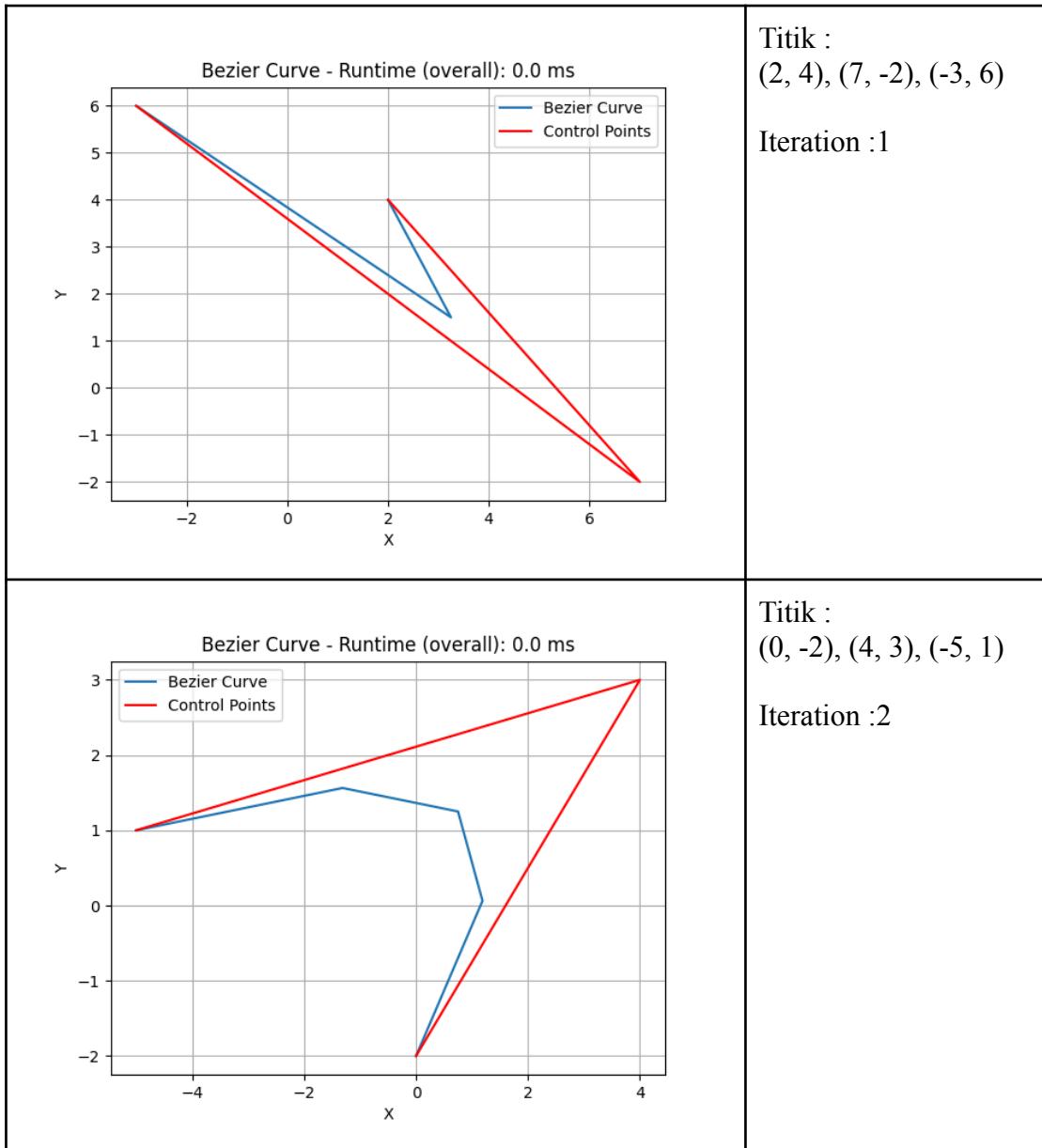


3.2. Tangkapan Layar Algoritma Divide and Conquer

Tabel 2. Hasil implementasi algoritma divide and conquer untuk 3 titik

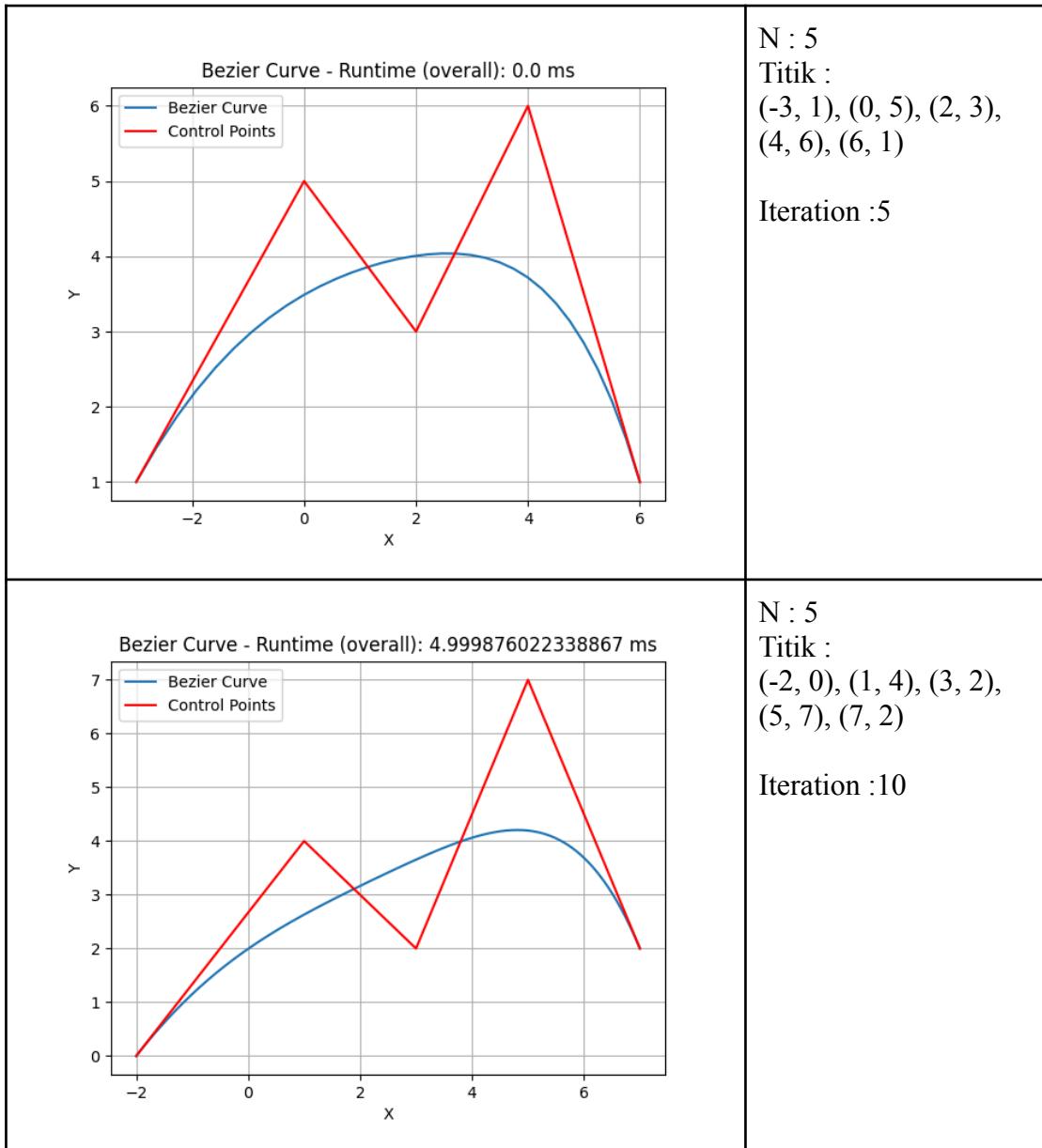
 <p>Bezier Curve - Runtime (overall): 0.0 ms</p> <p>Y-axis: -1 to 7 X-axis: -3 to 2</p> <p>Legend: Bezier Curve (blue line), Control Points (red line)</p> <p>The graph shows a blue Bezier curve and three red control points. The curve starts at (-3, 5), dips to a minimum near (0.5, 2.5), and ends at (2, -1). The red line connects the points (-3, 5), (2, -1), and (0, 7).</p>	<p>Titik : (-3, 5), (2, -1), (0, 7)</p> <p>Iteration : 5</p>
 <p>Bezier Curve - Runtime (overall): 60.24765968322754 ms</p> <p>Y-axis: -4 to 4 X-axis: -2 to 6</p> <p>Legend: Bezier Curve (blue line), Control Points (red line)</p> <p>The graph shows a blue Bezier curve and three red control points. The curve starts at (6, 3), dips to a minimum near (1, -5), and ends at (-2, 4). The red line connects the points (6, 3), (-2, 4), and (1, -5).</p>	<p>Titik : (6, 3), (-2, 4), (1, -5)</p> <p>Iteration : 15</p>

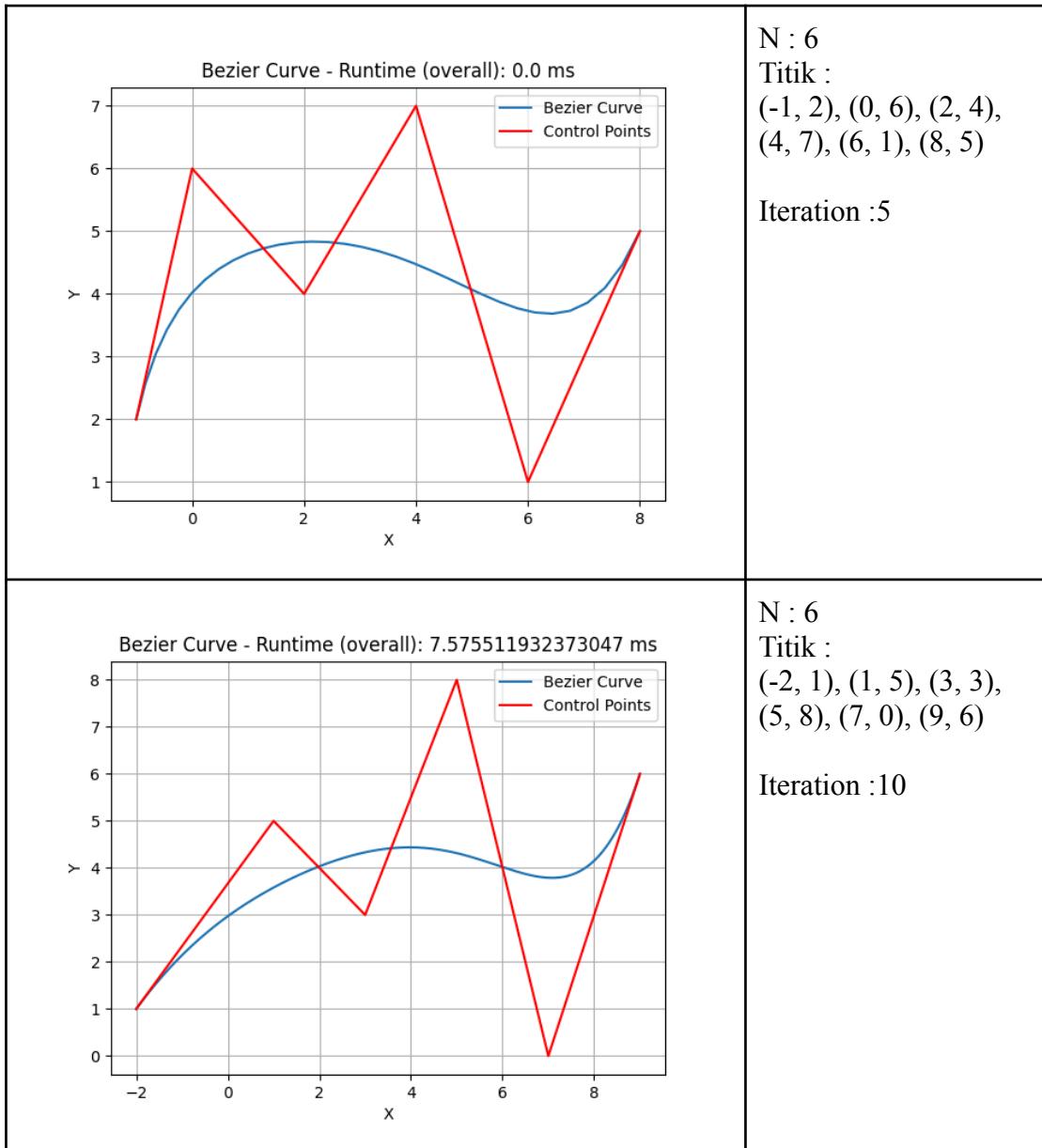




Tabel 3. Hasil implementasi algoritma divide and conquer untuk N titik

<p>Bezier Curve - Runtime (overall): 0.0 ms</p> <p>Bezier Curve Control Points</p>	N : 4 Titik : (3, 7), (-2, 4), (0, -1), (5, 2) Iteration : 5
<p>Bezier Curve - Runtime (overall): 3.0031204223632812 ms</p> <p>Bezier Curve Control Points</p>	N : 4 Titik : (1, -3), (6, 2), (4, 5), (-2, 1) Iteration : 10





BAB IV

Analisis Perbandingan Algoritma

4.1. Pembangkitan Kurva Bézier Kuadratik dengan *Bruteforce*

Untuk membangkitkan suatu kurva Bézier kuadratik dengan *bruteforce* sebanyak n iterasi (jika dibangkitkan dengan *Subdivision Algorithm*), akan dicari sebanyak $2^n - 1$ titik di luar dua titik kontrol P_0 dan P_2 yang menjadi ujung-ujung kurva. Untuk mencari setiap titik, akan dilakukan traversal larik berukuran $2^n - 1$, kemudian dilakukan operasi berdasarkan rumus yang sudah ditentukan. Operasi fungsi yang menerima input parameter t serta larik titik dapat dianggap konstan (karena sudah pasti hanya ada tiga titik input) untuk semua titik (misalnya c). Maka, kompleksitas algoritma ini menjadi $T(n) = 2c(2^n - 1)$, sehingga kompleksitas waktu asimptotiknya adalah $O(2^n)$.

4.2. Pembangkitan Kurva Bézier Kuadratik dengan *Divide and Conquer*

Algoritma *divide and conquer* yang digunakan pada pembangkitan kurva Bézier adalah *Mid-Point Algorithm*. Secara rekurens, kompleksitas waktu algoritma dapat dinyatakan sebagai berikut

$$T(n) = \begin{cases} 2T(n-1) + k & n > 1 \\ l & n = 1 \end{cases}$$

Dari persamaan tersebut diturunkan bahwa untuk iterasi n sebanyak m kali,

$$T(n) = 2T(n-1) + k = 4T(n-2) + 3k = \dots = 2^m T(1) + (2^m - 1).$$

Persamaan tersebut dapat disederhanakan menjadi

$$T(n) = 2^m(T(1) + 1) - 1 = T(n) = 2^m(l + 1) - 1.$$

Sehingga, kompleksitas waktunya adalah $T(n) = 2^n(l + 1) - 1$ dan dalam notasi asimptotik $O(2^n)$.

4.3. Perbandingan Algoritma *Brute Force* dan *Divide and Conquer*

Analisis kompleksitas yang telah dilakukan bahwa penggunaan algoritma *divide and conquer* tidak membuat algoritma menjadi lebih efisien. Pada uji coba yang kami lakukan, kami menemukan bahwa seringkali algoritma *brute force*

berjalan lebih cepat dari algoritma *divide and conquer*. Walau terdapat kesamaan kompleksitas asimptotik antara kedua algoritma, algoritma *divide and conquer* diimplementasikan secara rekursif. Akibatnya, terdapat *overhead* pada sistem yang lebih besar untuk memanggil fungsi rekursif bila dibandingkan penggunaan kalang pada algoritma *brute force*.

BAB V

Implementasi Bonus

5.1. Penjelasan Generalisasi Algoritma

Algoritma pembangkitan titik-titik kurva Bézier *Subdivision/Mid-Point Algorithm* dapat digeneralisasi untuk menerima masukkan sebanyak n buah titik. Algoritma yang kami implementasikan memodifikasi (teks yang ditebalkan) algoritma pembangkitan kurva berderajat dua sebagai berikut:

1. Fungsi akan memanggil strategi divide and conquer secara rekursif. Fungsi akan menerima 3 masukkan, **yakni jumlah titik kontrol (n), larik titik-titik kontrol (P_0, P_1, \dots, P_n)** dan juga jumlah iterasi (i). Basis dari fungsi tersebut merupakan jika iterasi mencapai 0, maka fungsi akan mengembalikan titik-titik kurva bérzier.
2. **Pencarian titik tengah dilakukan sampai dihasilkan satu titik tengah hasil. Artinya, dicari titik tengah dari P_0, P_1, \dots, P_n , yaitu Q_0, Q_1, \dots, Q_{n-1} . Kemudian dicari titik tengah di antara dua titik yang bersebelahan pada larik Q , yaitu R_0, R_1, \dots, R_{n-2} . Proses dilakukan sampai hanya tersisa satu titik.**
3. Setelah mendapat sebanyak $2n + 1$ titik kontrol yang baru (atau pada *source code* dinamakan *subcontrol points*), titik kontrol ini akan dipecah menjadi bagian kanan dan kiri. Bagian kiri dan kanan merupakan himpunan titik kontrol baru yaitu n elemen pertama dari larik *sub control points* dan n elemen terakhir dari larik *subcontrol points*.
4. Setelah dibagi menjadi dua bagian, fungsi Divide and Conquer akan dipanggil secara rekursif
5. Setelah mendapatkan titik titik dari tiap bagian, fungsi akan menggabungkan hasil-hasil titik tersebut dan menghasilkan solusi
6. Solusi akhir merupakan titik-titik untuk membentuk kurva bérzier yang nantikan akan di *plot* untuk visualisasi

Source Code Program



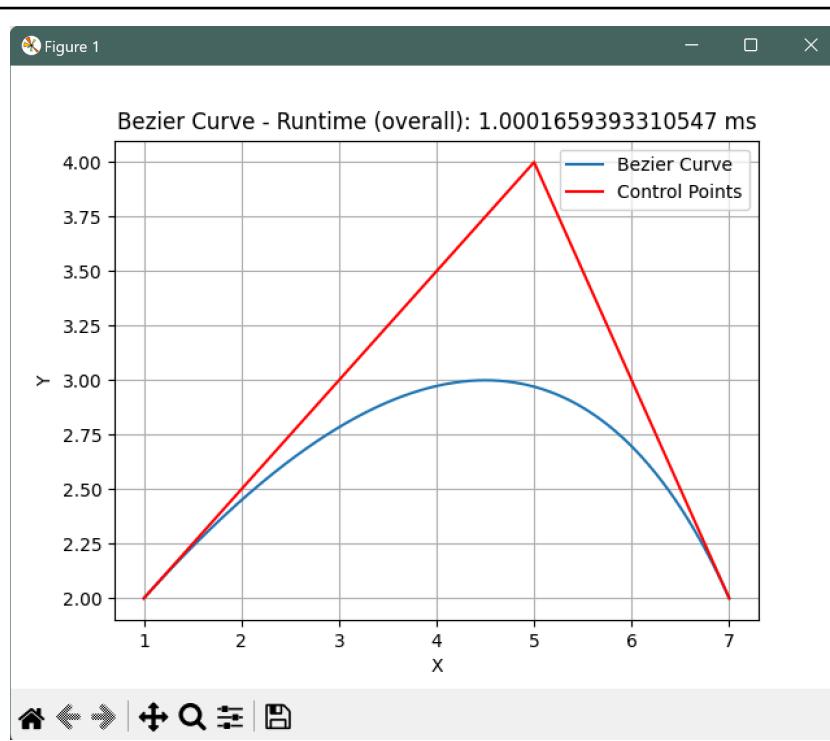
```
 1 def mid_point(point1, point2):
 2     return (point1[0] + point2[0]) / 2, (point1[1] + point2[1]) / 2
 3
 4 def n_bezier_dnc(control_points, num_of_iterations):
 5     if num_of_iterations == 1:
 6         subcontrol_points = [(0, 0)] * (len(control_points) * 2 - 1)
 7         subcontrol_points[0] = control_points[0]
 8         subcontrol_points[-1] = control_points[-1]
 9         new_subcontrol_points = get_subcontrol_points(control_points, subcontrol_points, 0)
10     return [control_points[0]] + [new_subcontrol_points[(len(subcontrol_points)-1)//2]] + [control_points[-1]]
11
12     else:
13         subcontrol_points = [(0, 0)] * (len(control_points) * 2 - 1)
14         subcontrol_points[0] = control_points[0]
15         subcontrol_points[-1] = control_points[-1]
16         subcontrol_points = get_subcontrol_points(control_points, subcontrol_points, 1)
17
18         left = n_bezier_dnc(subcontrol_points[:len(control_points)], num_of_iterations - 1)
19         right = n_bezier_dnc(subcontrol_points[-len(control_points):], num_of_iterations - 1)
20
21     return left + right
22
23 def get_subcontrol_points(control_points, subcontrol_points, counter):
24     if len(control_points) == 2:
25         new_subcontrol_points = subcontrol_points
26         new_subcontrol_points[(len(subcontrol_points)-1)//2] = mid_point(control_points[0], control_points[1])
27         return subcontrol_points
28
29     else:
30         length = len(control_points) - 1
31         mid_points = []
32         for i in range(length):
33             mid_points.append(mid_point(control_points[i], control_points[i+1]))
34         new_subcontrol_points = subcontrol_points
35         new_subcontrol_points[counter] = mid_points[0]
36         new_subcontrol_points[-(1 + counter)] = mid_points[-1]
37
38     return get_subcontrol_points(mid_points, new_subcontrol_points, counter + 1)
```

5.2. Penjelasan Visualisasi Pembentukan Kurva Bézier

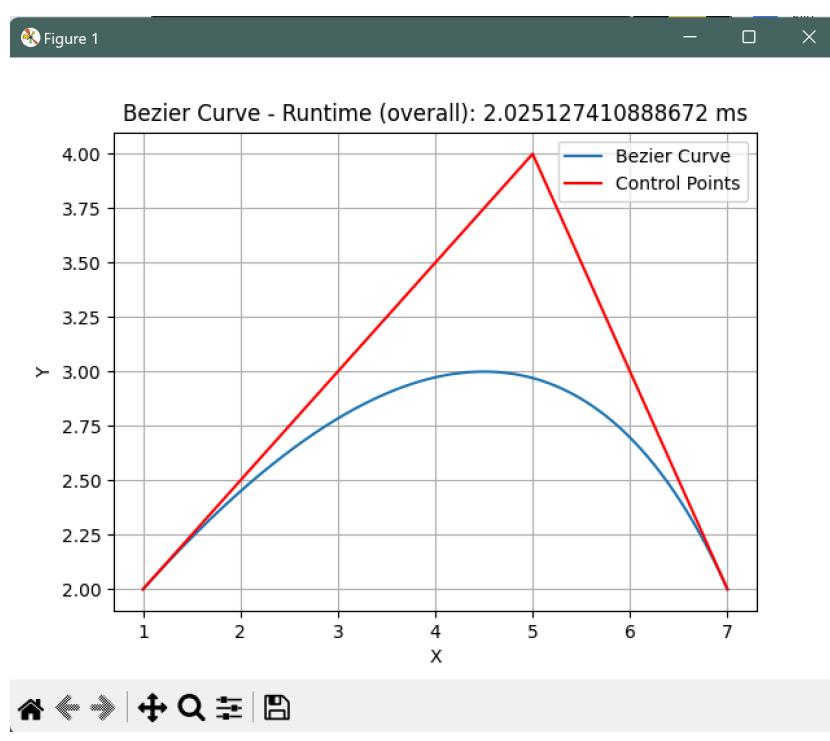
Visualisasi pembentukan titik kurva bezier dilakukan dengan menggunakan tampilan GUI (Graphical User Interface). Program akan memberikan opsi untuk tipe algoritma pembentukan kurva bezier. Selanjutnya pengguna akan memasukkan input. Hasil dari kurva bezier akan divisualisasikan dengan tiga cara, yaitu:

1. Visualisasi Statis

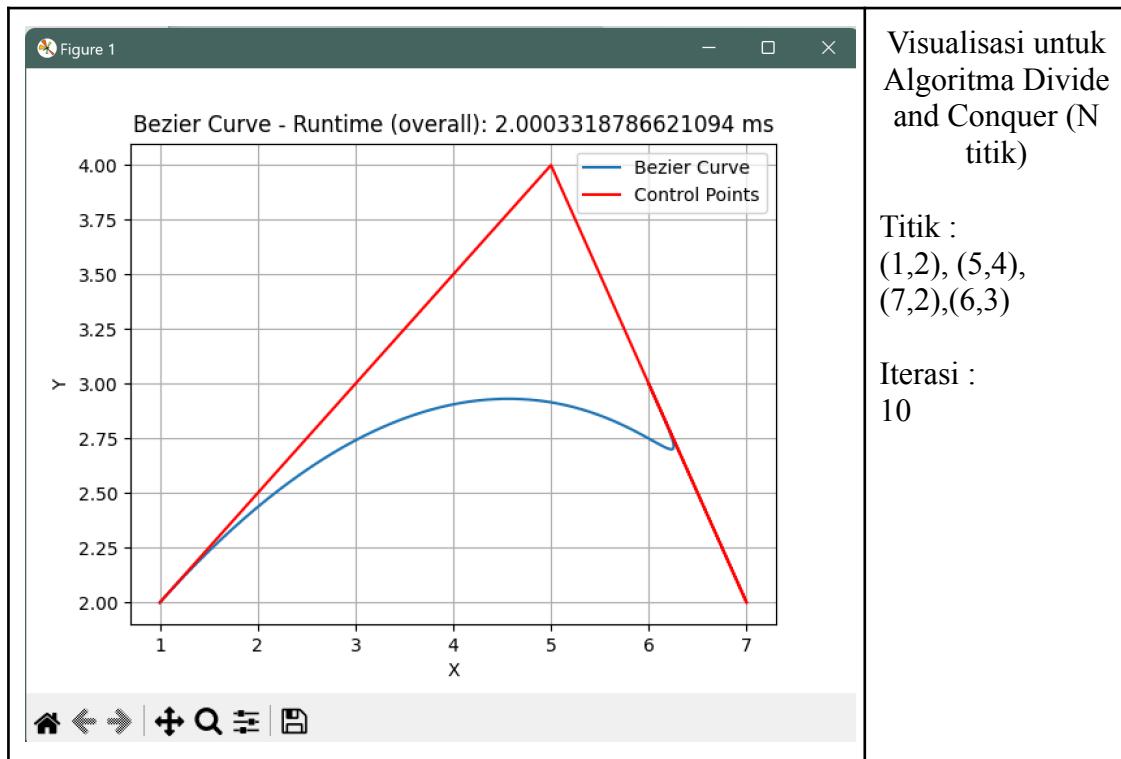
Visualisasi statis hanya akan menampilkan hasil akhir dari hasil perhitungan titik-titik kurva bezier



Visualisasi untuk
Algoritma Brute
Force

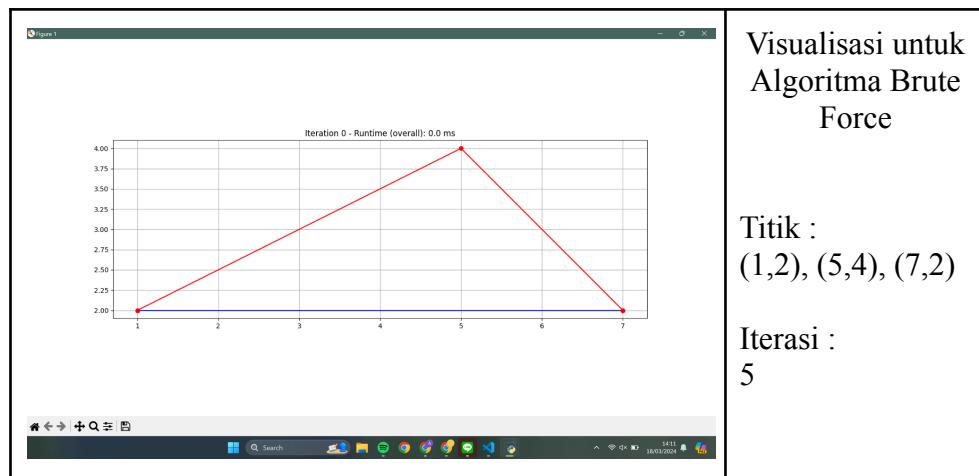


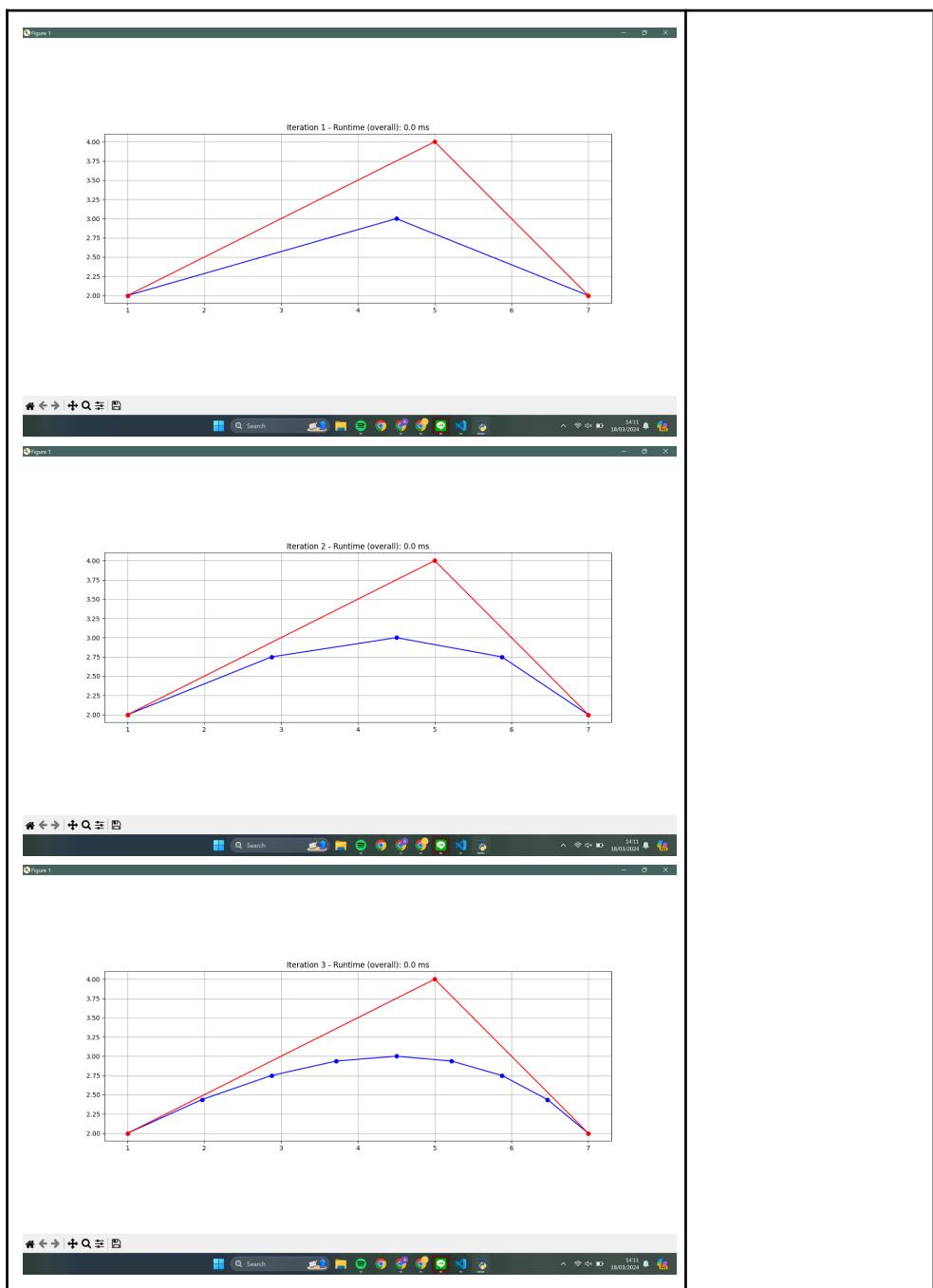
Visualisasi untuk
Algoritma Divide
and Conquer (3
titik)

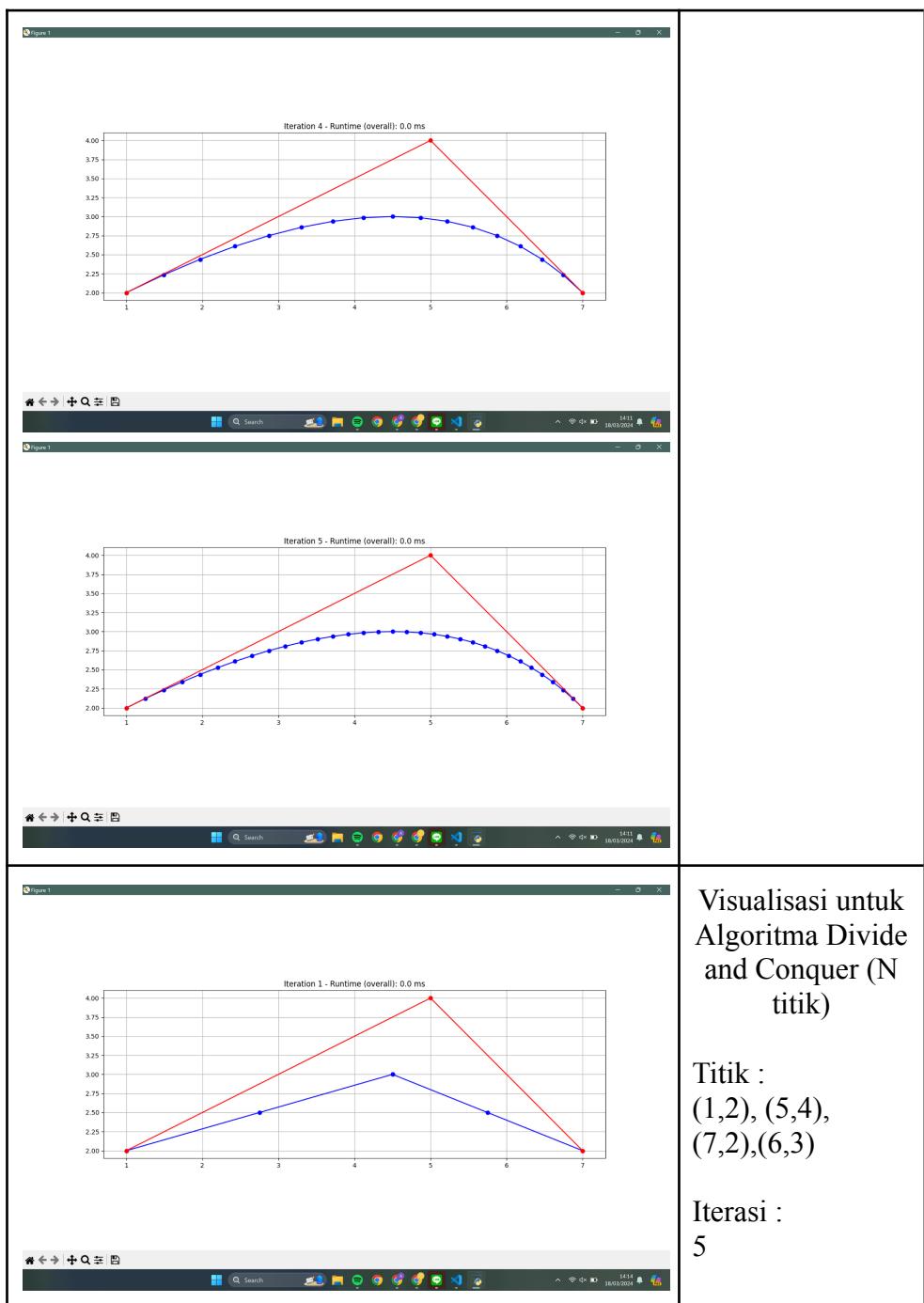


2. Visualisasi Animasi Per Iterasi

Visualisasi ini akan menampilkan kurva yang dibentuk pada tiap iterasi dan akan menampilkan kurva *final*.



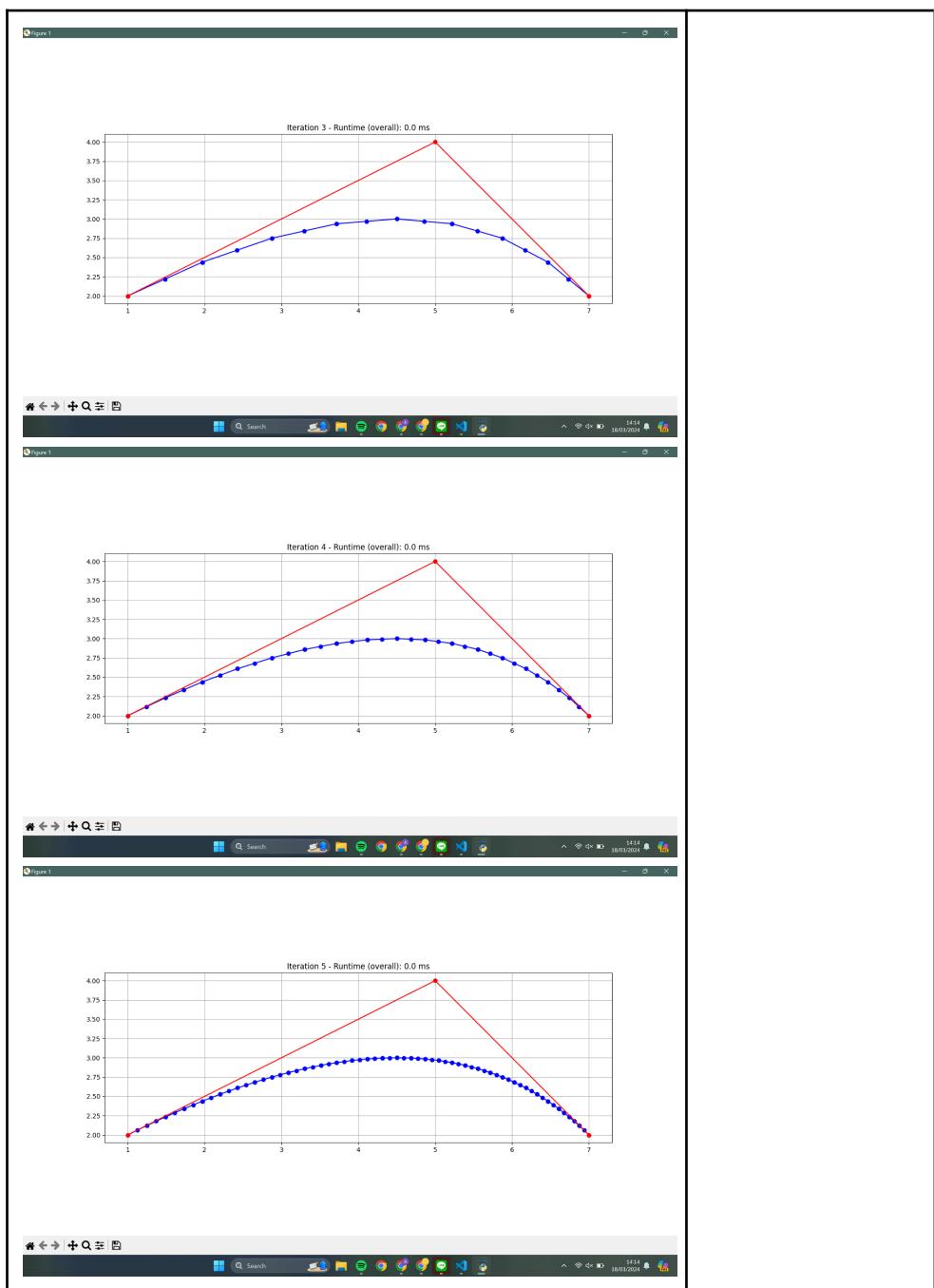


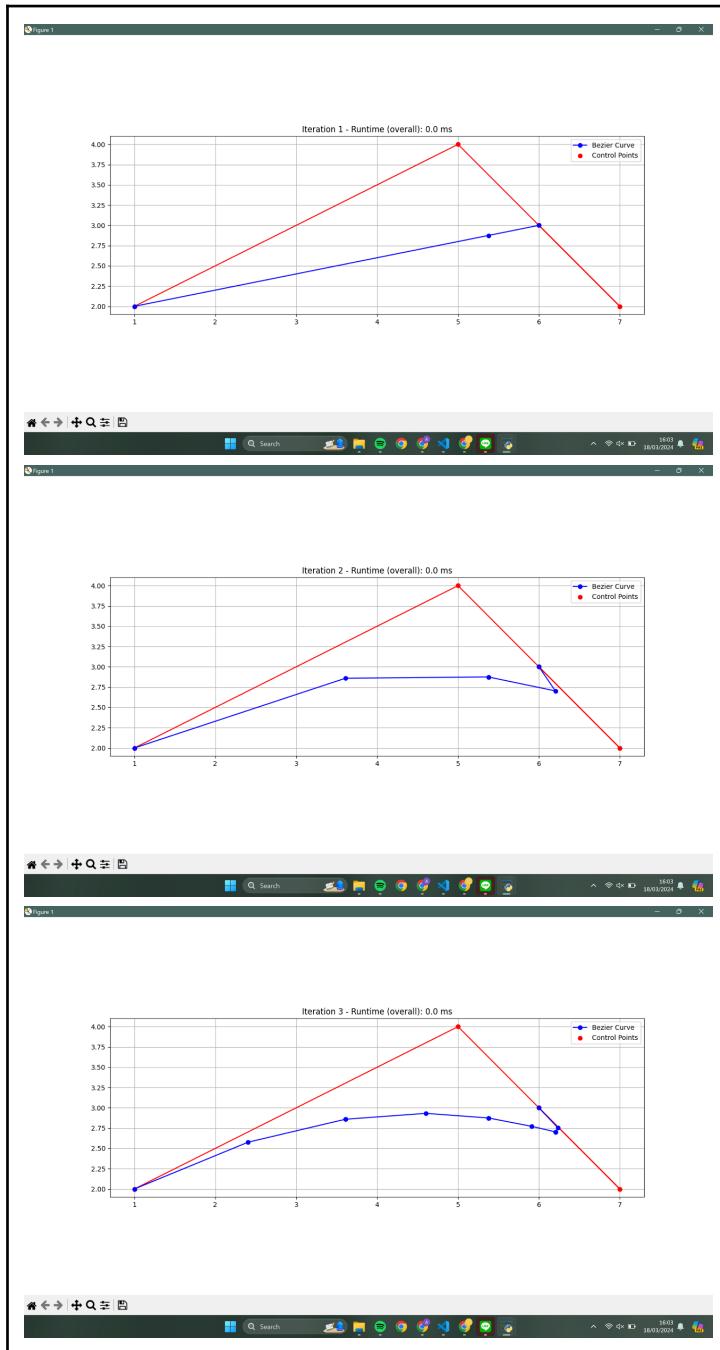


Visualisasi untuk
Algoritma Divide
and Conquer (N
titik)

Titik :
 $(1,2), (5,4),$
 $(7,2),(6,3)$

Iterasi :
 5

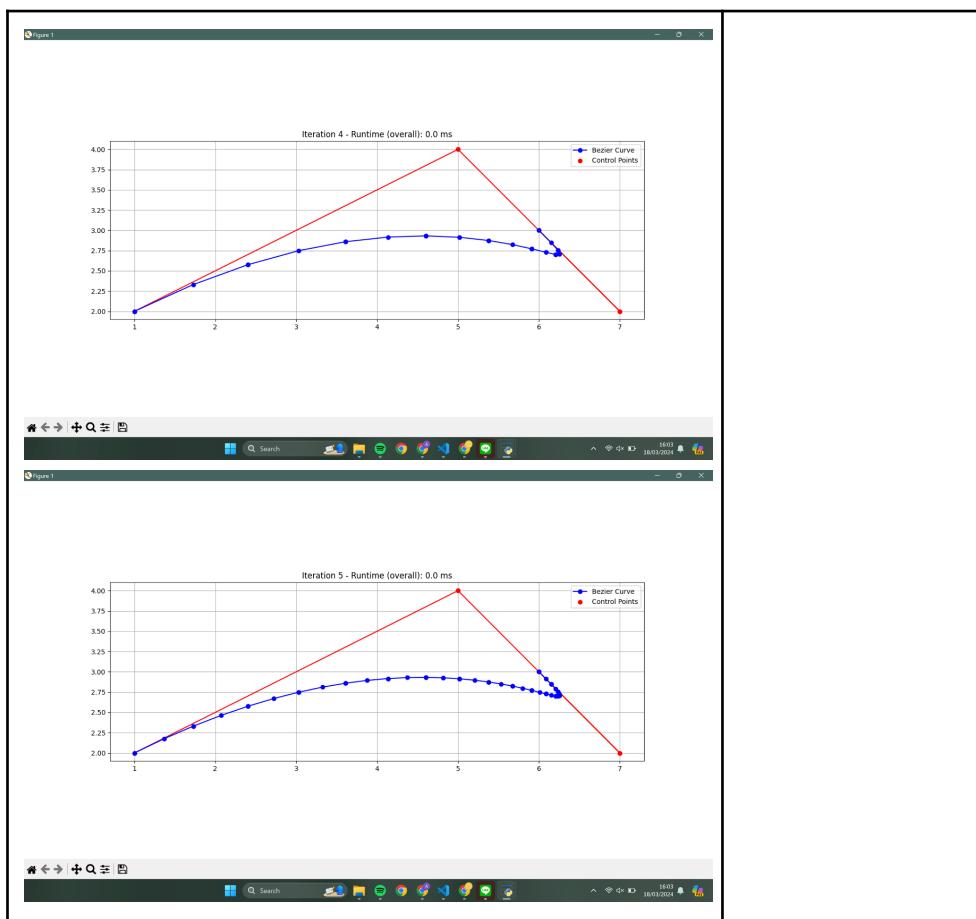




Visualisasi untuk
Algoritma Divide
and Conquer (N
titik)

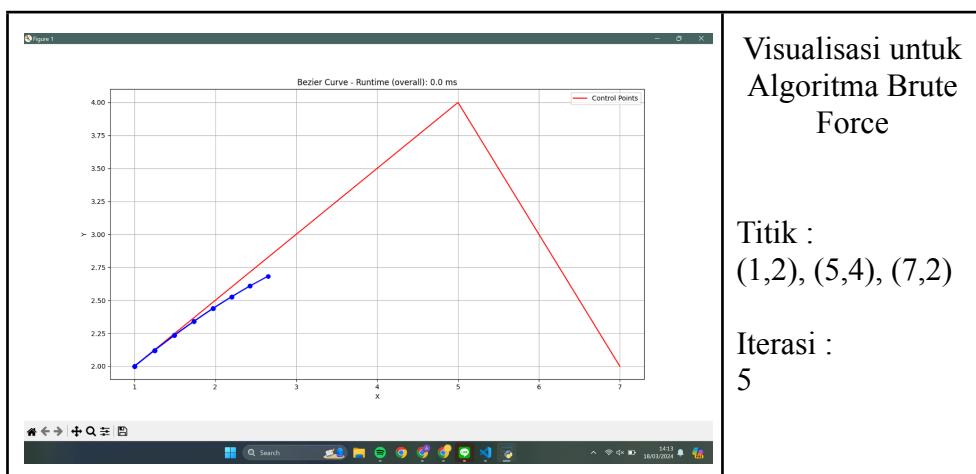
Titik :
(1,2), (5,4),
(7,2),(6,3)

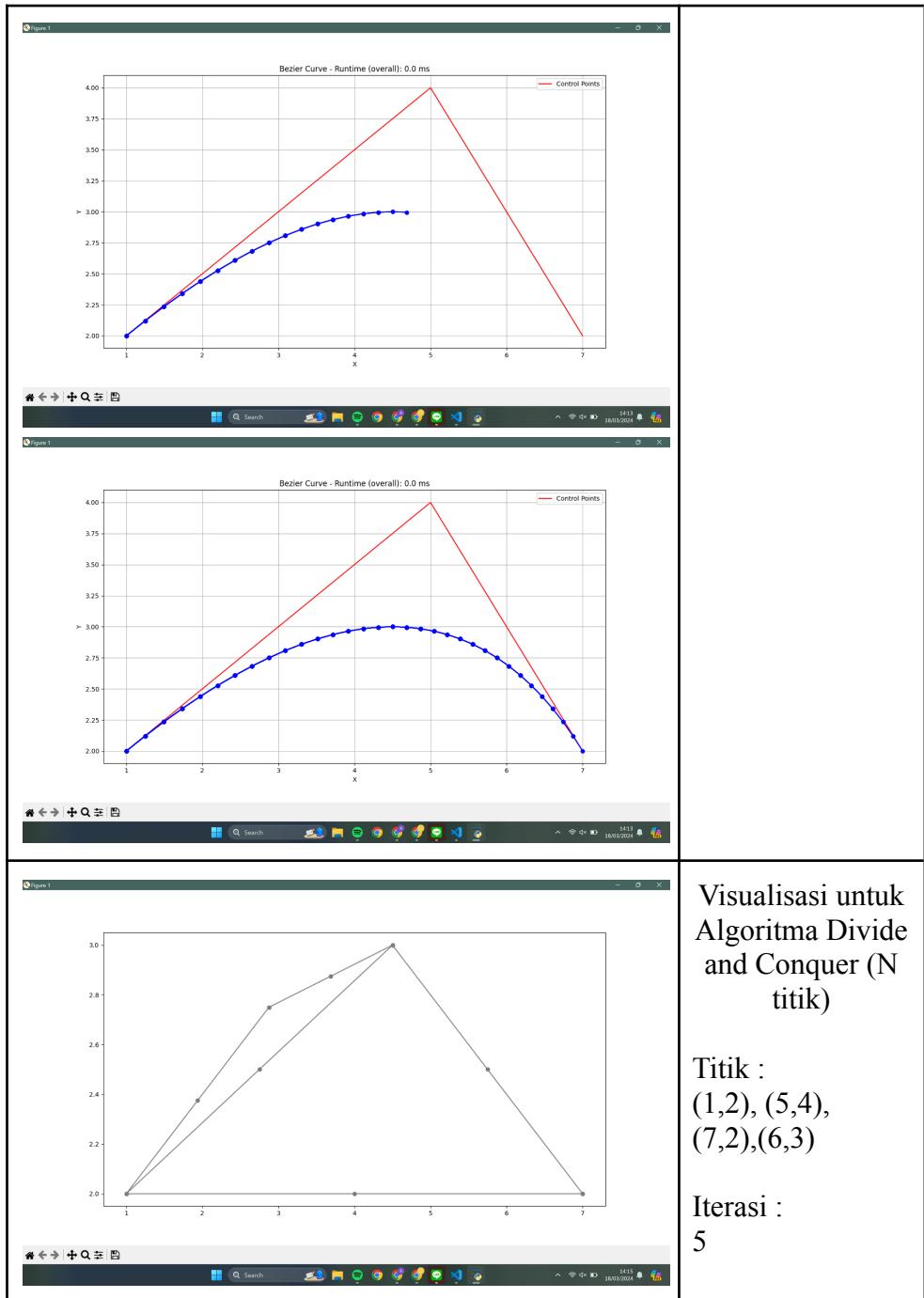
Iterasi :
5



3. Visualisasi Animasi Per Titik

Visualisasi ini akan menampilkan kurva titik bezier yang dibuat tiap iterasi, namun tidak digambarkan langsung bentuk kurva, melainkan titik-titik yang dihasilkan. Pada akhirnya kurva bezier *final* akan ditampilkan.

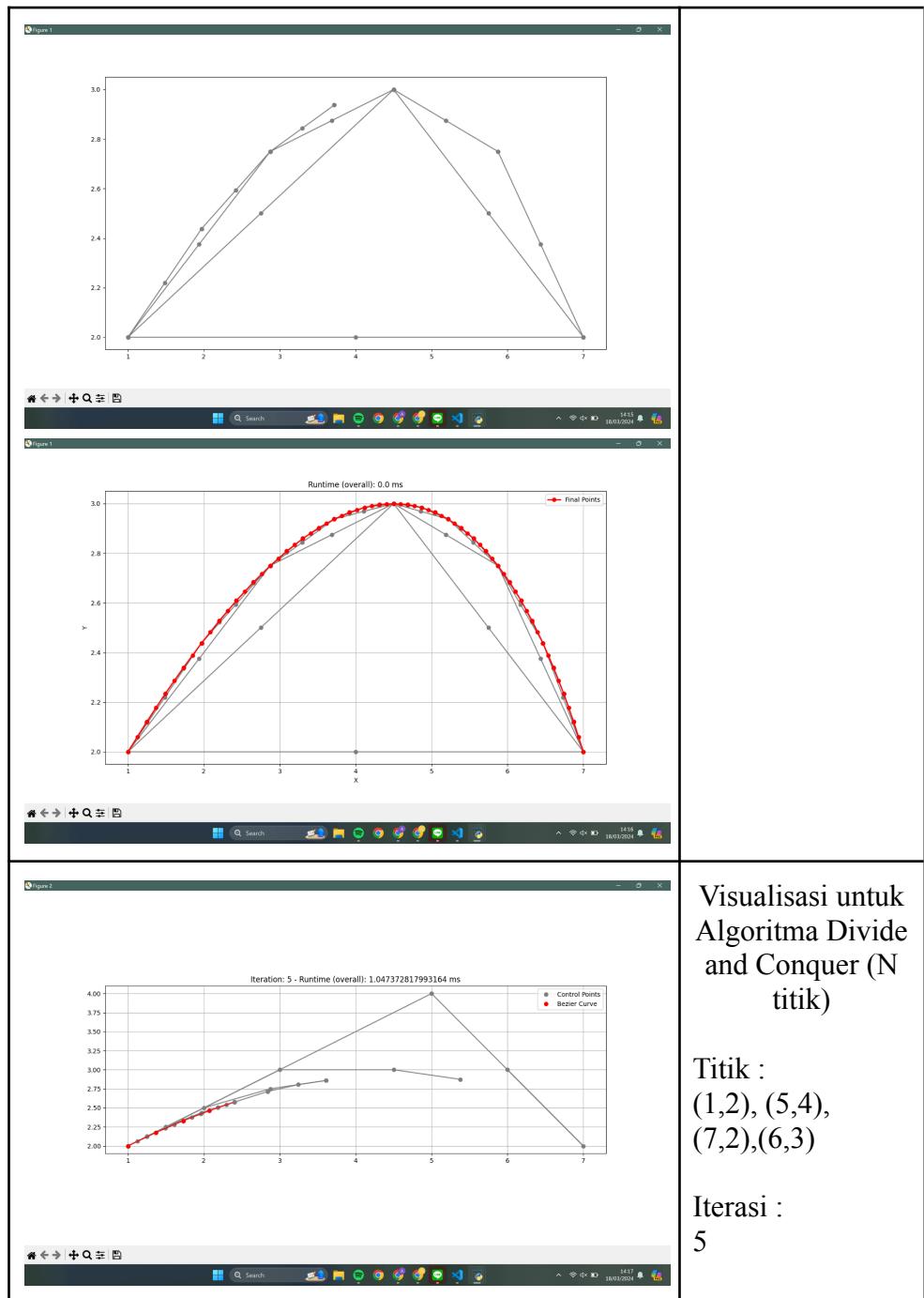


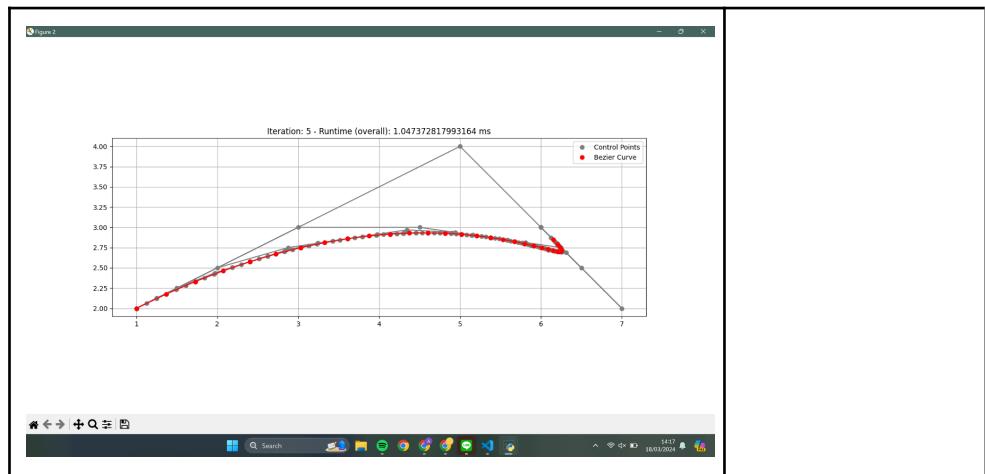


Visualisasi untuk
Algoritma Divide
and Conquer (N
titik)

Titik :
 $(1,2), (5,4),$
 $(7,2),(6,3)$

Iterasi :
 5





BAB VII

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.	✓	

Link Repository GitHub :

https://github.com/angiekriera/Tucil2_13522048_13522080