

LAPORAN TUGAS KECIL 2
IF2211 STRATEGI ALGORITMA



Disusun oleh:

Ahmad Mudabbir Arif (13522072)

Muhammad Naufal Aulia (13522074)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2024

DAFTAR ISI

DAFTAR ISI.....	1
BAB I.....	2
DESKRIPSI MASALAH.....	2
1.1 Kurva Bézier.....	2
1.2 Algoritma Divide and Conquer.....	2
1.3 Algoritma Divide and Conquer dalam Membentuk Kurva Bézier.....	2
1.4 Algoritma Brute Force dalam Membentuk Kurva Bézier.....	3
BAB II.....	5
2.1 Source Code: Algoritma Divide and Conquer.....	5
2.2 Source Code: Algoritma Brute Force.....	7
BAB III.....	9
3.1 Tampilan Program (GUI).....	9
3.2 Hasil Uji Coba DnC vs Brute Force.....	10
3.2.1 Hasil Uji Coba Tiga Titik Kontrol.....	10
3.2.1 Hasil Uji Coba N Titik Kontrol.....	14
3.3 Perbandingan Kompleksitas Algoritma Brute Force dengan Divide and Conquer.....	18
3.4 Implementasi Bonus.....	19
BAB IV.....	22
4.1 Link Repository.....	22
4.2 Checklist.....	22
REFERENSI.....	23

BAB I

DESKRIPSI MASALAH

1.1 Kurva Bézier

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 P_0 sampai P_n , 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 P_0 , sedangkan titik kontrol terakhir adalah P_3 . Titik kontrol P_1 dan P_2 disebut sebagai titik kontrol antara yang tidak terletak dalam kurva yang terbentuk.

1.2 Algoritma Divide and Conquer

Divide artinya membagi, membagi membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil. *Conquer* dalam hal ini berarti menyelesaikan (*solve*) masing-masing upa-persoalan (secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar).

Dengan demikian, algoritma *divide and conquer* merupakan algoritma rekursif yang memecah masalah menjadi submasalah yang lebih kecil, menyelesaikan submasalah tersebut, dan menggabungkan solusi dari submasalah tersebut untuk memecahkan masalah asal. Algoritma ini sering digunakan dalam berbagai konteks, seperti algoritma pengurutan (misalnya merge sort dan quick sort), algoritma pencarian (misalnya binary search), dan algoritma geometri (misalnya algoritma untuk menghitung jarak terdekat antara titik-titik dalam ruang dua dimensi).

1.3 Algoritma Divide and Conquer dalam Membentuk Kurva Bézier

Dalam membentuk kurva Bézier, dapat diterapkan pendekatan algoritma divide and conquer. Dengan pendekatan ini, kurva Bézier dapat dibentuk dengan cara membagi persoalan titik kontrol menjadi dua bagian, menyelesaikan masing-masing bagian secara rekursif, dan menggabungkan

hasilnya. Untuk kurva Bézier kuadratik, algoritma ini dapat diimplementasikan dengan cara sebagai berikut:

1. Kumpulkan titik awal (P_0), titik kontrol (P_1), dan titik akhir (P_2) dari masukan.
2. Cari titik tengah dari titik P_0 dan P_1 serta titik P_1 dan P_2 (antara dua titik pertama dan antara dua titik terakhir). Sebutlah titik tersebut sebagai Q_0 dan Q_1 .
3. Cari titik tengah dari kedua titik tengah sebelumnya, yaitu titik R
4. Titik tengah (R) yang dibentuk inilah yang akan menjadi titik hasil pada kurva Bézier untuk iterasi pertama.
5. Lakukan pembagian masalah menjadi dua submasalah, yaitu kumpulan titik kiri dan kumpulan titik kanan. Kumpulkan titik awal P_0 , titik tengah Q_0 , dan titik tengah R sebagai titik akhir dari bagian kiri. Untuk bagian kanan, kumpulkan titik tengah R sebagai titik awal, titik tengah Q_1 , dan titik akhir P_2 .
6. Lakukan rekursi fungsi yang sama untuk bagian kiri dahulu lalu dilanjut dengan rekursi untuk bagian kanan. Rekursi ini akan terus membagi titik ke titik menjadi bagian yang lebih kecil sesuai dengan iterasi masukan hingga iterasi mencapai 0 (selesai).
7. Gabungkan hasil dari kedua submasalah dengan cara mengkonkatenasi titik-titik hasil rekursi bagian kiri dengan bagian kanan.
8. Hasil penggabungan ini adalah titik-titik pada kurva Bézier yang dibentuk sesuai jumlah iterasinya.

1.4 Algoritma Brute Force dalam Membentuk Kurva Bézier

Brute force adalah sebuah pendekatan yang lempang (straightforward) dalam memecahkan suatu masalah, didasarkan pada pernyataan masalah (problem statement) dan melibatkan definisi konsep. Algoritma ini memecahkan masalah secara langsung dengan cara yang jelas.

Dalam membentuk kurva bézier menggunakan pendekatan algoritma brute force, kita dapat menggunakan interpolasi linear antara semua pasangan titik berurutan (kurva Bézier linier) dan melakukan rekursi dengan titik-titik baru. Langkah-langkah dalam membuat kurva bézier menggunakan pendekatan algoritma brute force adalah sebagai berikut:

1. Kumpulkan semua titik kontrol ke dalam sebuah array

2. Untuk setiap iterasi, hitung parameter 't' sesuai jumlah titik dalam rentang 0 sampai 1. Parameter ini digunakan untuk mendapatkan titik yang akan diinterpolasi menggunakan rumus kurva Bézier linear.
3. Untuk setiap nilai 't', hitung titik pada kurva menggunakan rumus kurva Bézier linear. Hal ini dilakukan dengan mencari titik-titik antara pasangan titik kontrol berurutan.
4. Tambahkan titik hasil interpolasi ke dalam array.
5. Setelah iterasi selesai, array yang berisi titik-titik hasil interpolasi akan menjadi kurva Bézier yang diinginkan.

BAB II

IMPLEMENTASI ALGORITMA

Implementasi pembentukan kurva Bezier menggunakan algoritma Divide and Conquer dan Brute Force sebagai pembanding. Implementasi kedua algoritma penulis lakukan menggunakan bahasa pemrograman Python yang terdiri dari dua file yaitu `divideandconquer.py` yang berisi algoritma Divide and Conquer dan file `bruteforce.py` yang berisi algoritma Brute Force. Untuk algoritma Divide and Conquer sudah dilengkapi dengan GUI tkinter sebagai antarmuka pengguna dalam menjalankan program.

Program ini menggunakan library-library python berikut:

1. numpy: digunakan untuk operasi matematika
2. matplotlib: digunakan untuk membuat plot kurva Bezier
3. tkinter: digunakan untuk membuat GUI
4. time: digunakan untuk mengukur waktu eksekusi
5. os: digunakan untuk mengakses path file
6. sys: digunakan untuk mengakses path file

2.1 Source Code: Algoritma Divide and Conquer

```
##### ALGORITMA DIVIDE AND CONQUER #####
# Pembuat titik kontrol tengah
def generate_control_points(points):
    control_points = []
    for i in range(len(points) - 1):
        control_points.append((points[i] + points[i + 1]) / 2)
    if len(control_points) > 1:
        recursive_control_points = generate_control_points(control_points)
        control_points.extend(recursive_control_points)
    return control_points

# Divide and Conquer kurva Bezier
def divide_and_conquer(points, iterations):
    global all_points

    # Basis case
```

```

if iterations == 0:
    return [points[0], points[-1]]

# Membuat titik kontrol tengah
control_points = generate_control_points(points)
mid_point = control_points[-1]

all_points.append(control_points)

# Kumpulan titik bagian kiri
array_left_points = []
array_left_points.append(points[0])
array_left_points.append(control_points[0])

len_points = len(points)
i = len_points - 1
while i < len(control_points):
    array_left_points.append(control_points[i])
    len_points -= 1
    i += len_points - 1

# Kumpulan titik bagian kanan
array_right_points = []
control_points.reverse()
array_right_points.append(control_points[0])
array_right_points.append(control_points[1])

start_index = 3
i = start_index
while i < len(control_points):
    array_right_points.append(control_points[i])
    i += start_index
    start_index += 1

array_right_points.append(points[-1])

# Rekursi pada bagian kiri dan kanan
left_points = divide_and_conquer(array_left_points, iterations - 1)

```

```
right_points = divide_and_conquer(array_right_points, iterations - 1)
```

```
# Menggabungkan hasil dari kedua submasalah
```

```
return left_points[:-1] + [mid_point] + right_points
```

2.2 Source Code: Algoritma Brute Force

```
##### ALGORITMA BRUTE FORCE #####  
def bezier_linear(points, t):  
    # Basis case: jika hanya satu titik tersisa, kembalikan titik tersebut  
    if len(points) == 1:  
        return points[0]  
  
    # Lakukan interpolasi linear antara semua pasangan titik berurutan (kurva Bézier linier)  
    new_points = []  
    for i in range(len(points) - 1):  
        new_points.append((1 - t) * points[i] + t * points[i + 1])  
  
    # Rekursi dengan titik-titik baru  
    return bezier_linear(new_points, t)  
  
def bezier_curve_brute_force(points, iterations):  
    # Kalkulasi titik dari iterasi yang diberikan  
    # Dari ekuivalensi parameter iteration pada Divide and Conquer  
    n_point = (2 ** iterations) + 1  
  
    print("ALGORITMA BRUTE FORCE")  
    print(f'Jumlah iterasi : {iterations} (ekivalensi iterasi DnC)')  
    print(f'Jumlah titik : {n_point}')  
  
    # Array untuk menyimpan titik-titik pada kurva Bezier  
    curve_points = []  
  
    # Iterasi parameter t sesuai jumlah titik  
    for t in np.linspace(0, 1, n_point):  
        curve_point = bezier_linear(points, t)
```



```
curve_points.append(curve_point)

return curve_points

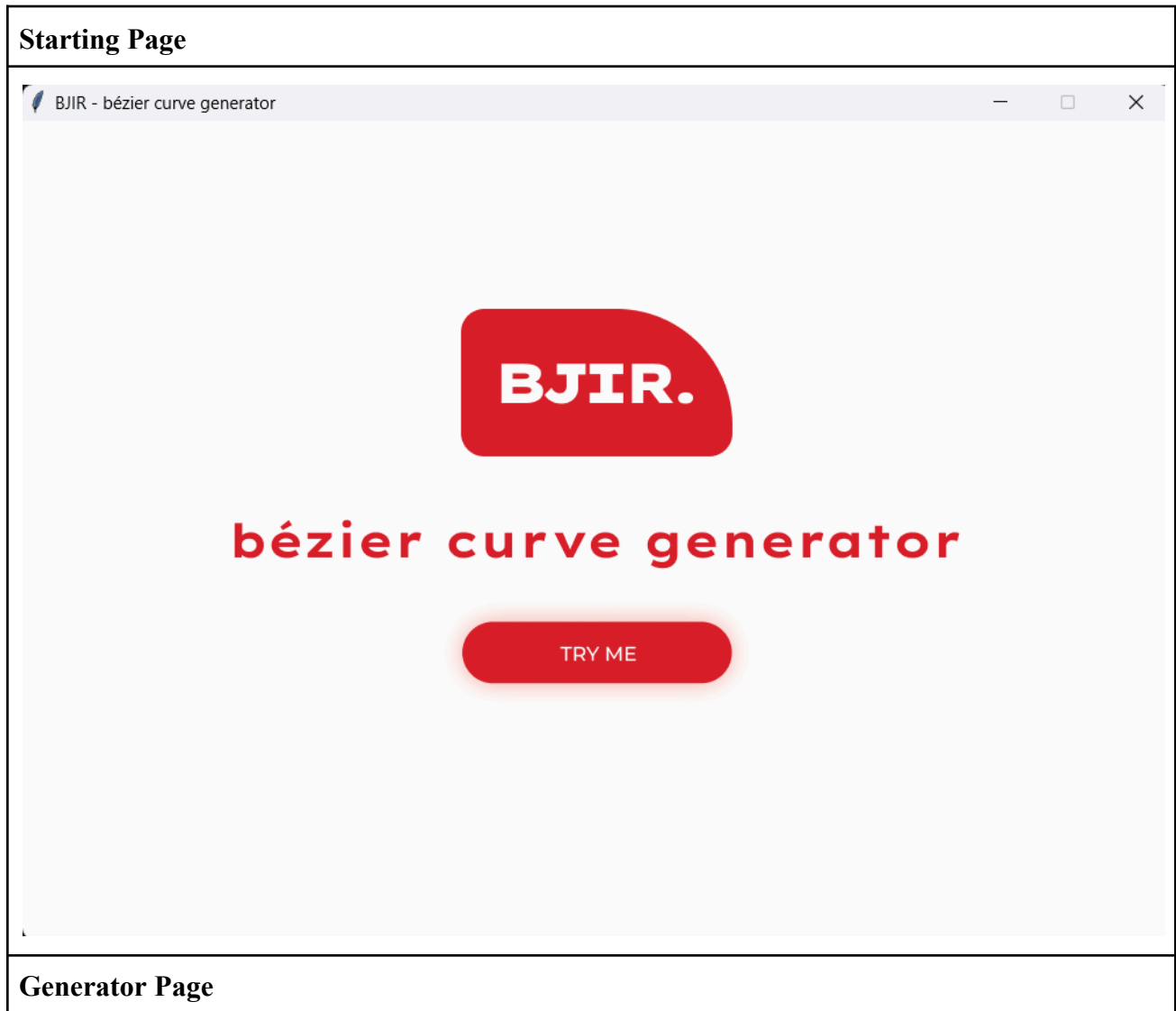
# input titik dan iterasi
points_input = input("Masukkan titik-titik kontrol (pisahkan dengan spasi, misal: 0,0 1.5,4 4,0 5,-3): ")
points = np.array([[float(coord) for coord in point.split(',') for point in points_input.split()])
iterations = int(input("Masukkan jumlah iterasi: "))
print()
```

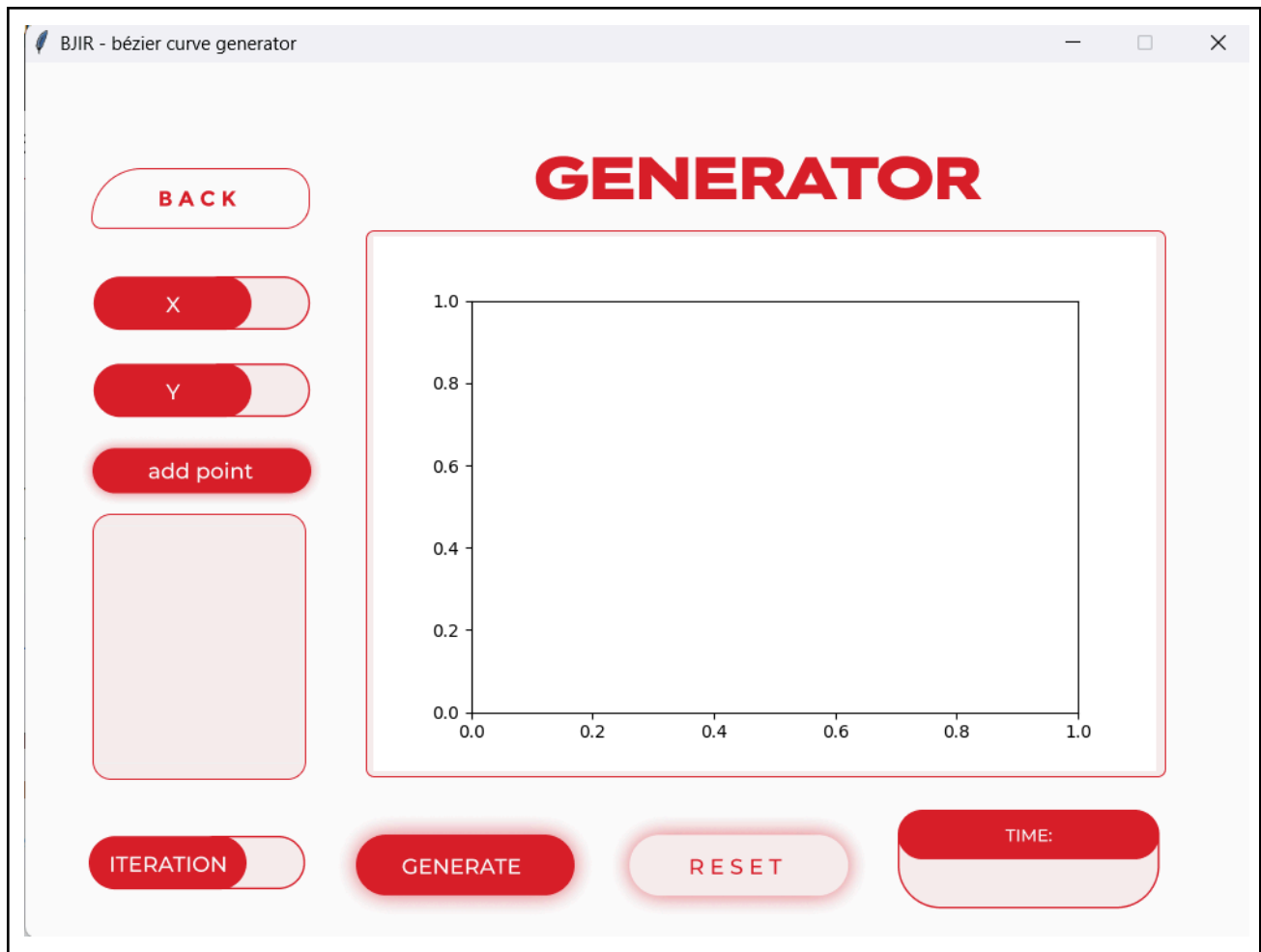
Untuk *source code* GUI karena terlalu panjang untuk ditampilkan, maka dapat langsung dilihat melalui pranala github repository pada [bab iv](#) di bagian bawah dokumen ini.

BAB III

HASIL PENGUJIAN DAN ANALISIS

3.1 Tampilan Program (GUI)





3.2 Hasil Uji Coba DnC vs Brute Force

3.2.1 Hasil Uji Coba Tiga Titik Kontrol

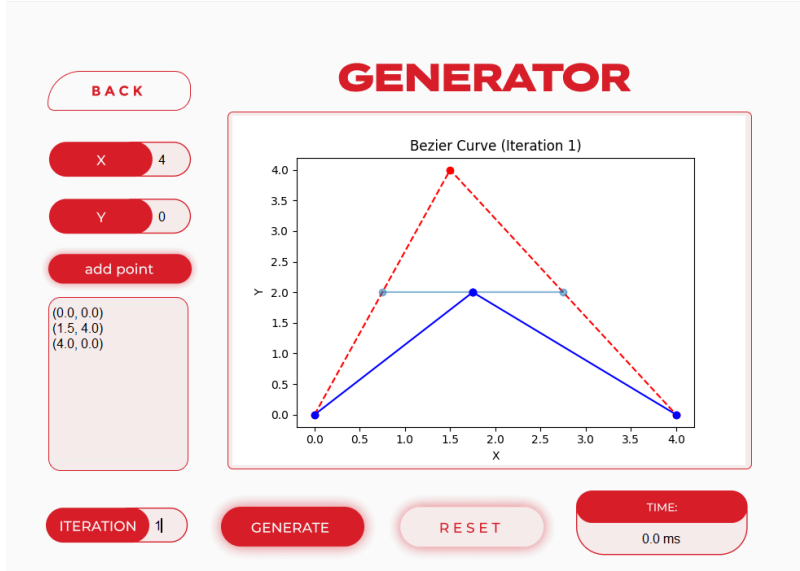
Soal 1 :

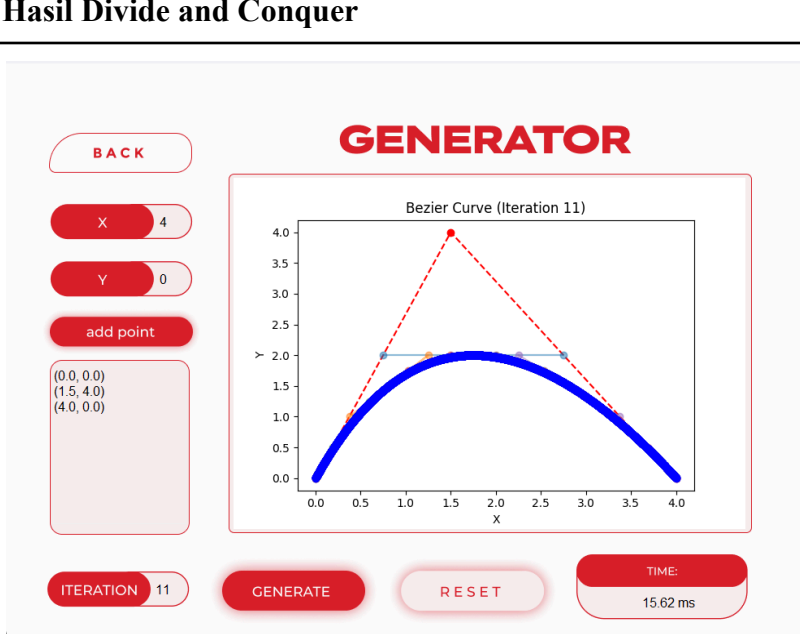
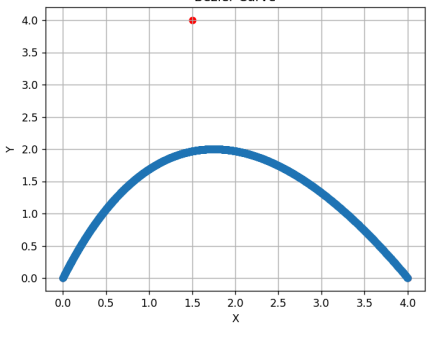
Titik: (0,0), (1.5,4), (4,0)

Iterasi: 1

Hasil Divide and Conquer

Referensi Brute Force

	<p>Masukkan titik-titik kontrol (pisahkan dengan spasi, misal: 0,0 1.5,4 4,0 5,-3): 0,0 1.5,4 4,0</p> <p>Masukkan jumlah iterasi: 1</p> <p>ALGORITMA BRUTE FORCE</p> <p>Jumlah iterasi : 1 (ekivalensi iterasi DnC)</p> <p>Jumlah titik : 3</p> <p>Waktu eksekusi : 0.0 ms</p>
<p>Keterangan : Solusi yang dihasilkan kedua algoritma adalah sama. Dalam jumlah titik dan iterasi ini, belum dapat dibandingkan waktu eksekusi kedua algoritma.</p>	

<p>Soal 2 : Titik: (0,0), (1.5,4), (4,0) Iterasi: 11</p>	
<p>Hasil Divide and Conquer</p> 	<p>Referensi Brute Force</p> <p>Masukkan titik-titik kontrol (pisahkan dengan spasi, misal: 0,0 1.5,4 4,0 5,-3): 0,0 1.5,4 4,0</p> <p>Masukkan jumlah iterasi: 11</p> <p>ALGORITMA BRUTE FORCE</p> <p>Jumlah iterasi : 11 (ekivalensi iterasi DnC)</p> <p>Jumlah titik : 2049</p> <p>Waktu eksekusi : 20.0 ms</p> 
<p>Keterangan : Solusi yang dihasilkan kedua algoritma adalah sama. Dalam jumlah titik dan iterasi ini, sudah dapat dibandingkan waktu eksekusi kedua algoritma. Dapat terlihat bahwa waktu</p>	

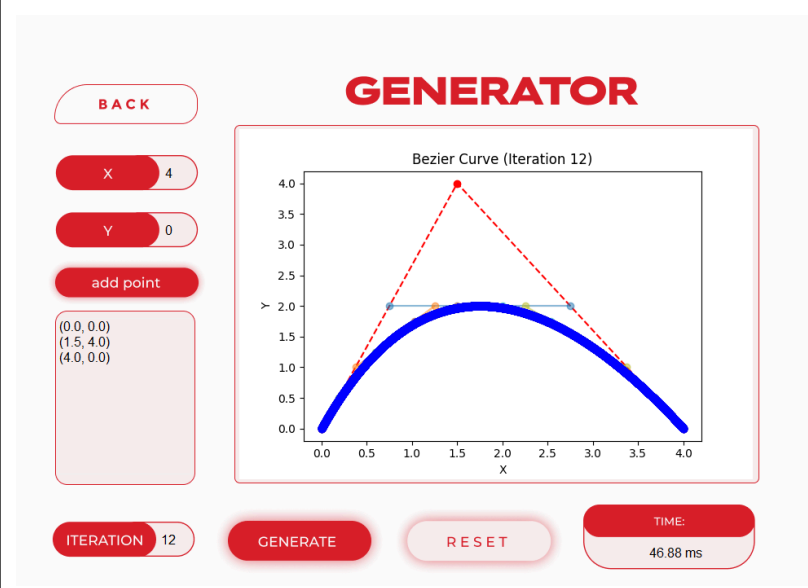
eksekusi algoritma divide and conquer sedikit lebih cepat dibanding brute force dengan selisih 4,38 ms.

Soal 3 :

Titik: (0,0), (1.5,4), (4,0)

Iterasi: 12

Hasil Divide and Conquer



Referensi Brute Force

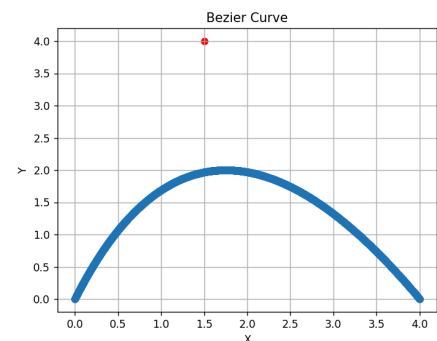
Masukkan titik-titik kontrol (pisahkan dengan spasi, misal: 0,0 1.5,4 4,0 5,-3): 0,0 1.5,4 4,0
Masukkan jumlah iterasi: 12

ALGORITMA BRUTE FORCE

Jumlah iterasi : 12 (ekivalensi iterasi DnC)

Jumlah titik : 4097

Waktu eksekusi : 60.0 ms



Keterangan : Solusi yang dihasilkan kedua algoritma adalah sama. Dengan iterasi berjumlah 1 lebih banyak dibandingkan sebelumnya, sudah dapat dibandingkan waktu eksekusi kedua algoritma. Dapat terlihat bahwa waktu eksekusi algoritma divide and conquer sedikit lebih cepat dibanding brute force dengan selisih 13,12 ms.

Soal 4 :

Titik: (0,0), (4,4), (1,-10)

Iterasi: 12

Hasil Divide and Conquer

Referensi Brute Force

BACK

X1

Y-10

add point

(0,0,0,0)
(4,0,4,0)
(1,0,-10,0)

ITERATION12

GENERATE

RESET

TIME:
46.88 ms

GENERATOR

Masukkan titik-titik kontrol (pisahkan dengan spasi, misal: 0,0 1.5,4 4,0 5,-3): 0,0 4,4 1,-10

Masukkan jumlah iterasi: 12

ALGORITMA BRUTE FORCE

Jumlah iterasi : 12 (ekivalensi iterasi DnC)

Jumlah titik : 4097

Waktu eksekusi : 50.0 ms

Keterangan : Solusi yang dihasilkan kedua algoritma adalah sama. Dengan jumlah iterasi yang sama dengan sebelumnya namun titik berbeda, dapat terlihat bahwa waktu eksekusi algoritma divide and conquer sedikit lebih cepat dibanding brute force dengan selisih 3,12 ms.

Soal 5 :

Titik: (0,0), (4,4), (1,-10)

Iterasi: 13

BACK

X1

Y-10

add point

(0,0,0,0)
(4,0,4,0)
(1,0,-10,0)

ITERATION13

GENERATE

RESET

TIME:
78.12 ms

GENERATOR

Masukkan titik-titik kontrol (pisahkan dengan spasi, misal: 0,0 1.5,4 4,0 5,-3): 0,0 4,4 1,-10

Masukkan jumlah iterasi: 13

ALGORITMA BRUTE FORCE

Jumlah iterasi : 13 (ekivalensi iterasi DnC)

Jumlah titik : 8193

Waktu eksekusi : 80.0 ms

13

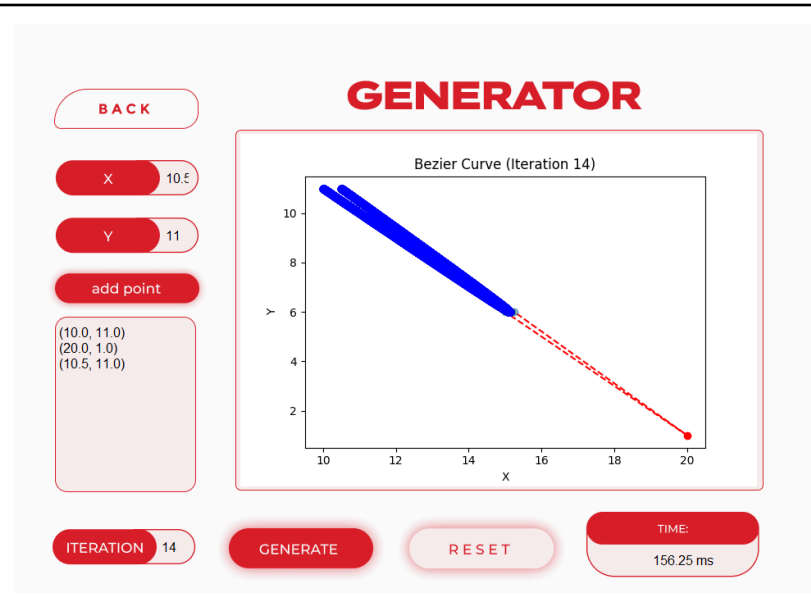
Keterangan : Solusi yang dihasilkan kedua algoritma adalah sama. Dengan iterasi berjumlah 1 lebih banyak dibandingkan sebelumnya, dapat terlihat bahwa waktu eksekusi algoritma divide and conquer sedikit lebih cepat dibanding brute force dengan selisih 1,88 ms.

Soal 6 :

Titik: (10,11), (20,1), (10.5,11)

Iterasi: 14

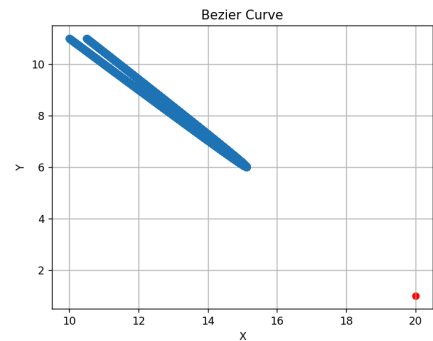
Hasil Divide and Conquer



Referensi Brute Force

Masukkan titik-titik kontrol (pisahkan dengan spasi, misal: 0,0 1.5,4 4,0 5,-3): 10,11 20,1 10.5,11
Masukkan jumlah iterasi: 14

ALGORITMA BRUTE FORCE
Jumlah iterasi : 14 (ekivalensi iterasi DnC)
Jumlah titik : 16385
Waktu eksekusi : 170.0 ms



Keterangan : Solusi yang dihasilkan kedua algoritma adalah sama. Dengan titik yang berbeda dan lebih banyak 1 iterasi dibanding sebelumnya, dapat terlihat bahwa waktu eksekusi algoritma divide and conquer sedikit lebih cepat dibanding brute force dengan selisih 13,75 ms.

3.2.1 Hasil Uji Coba N Titik Kontrol

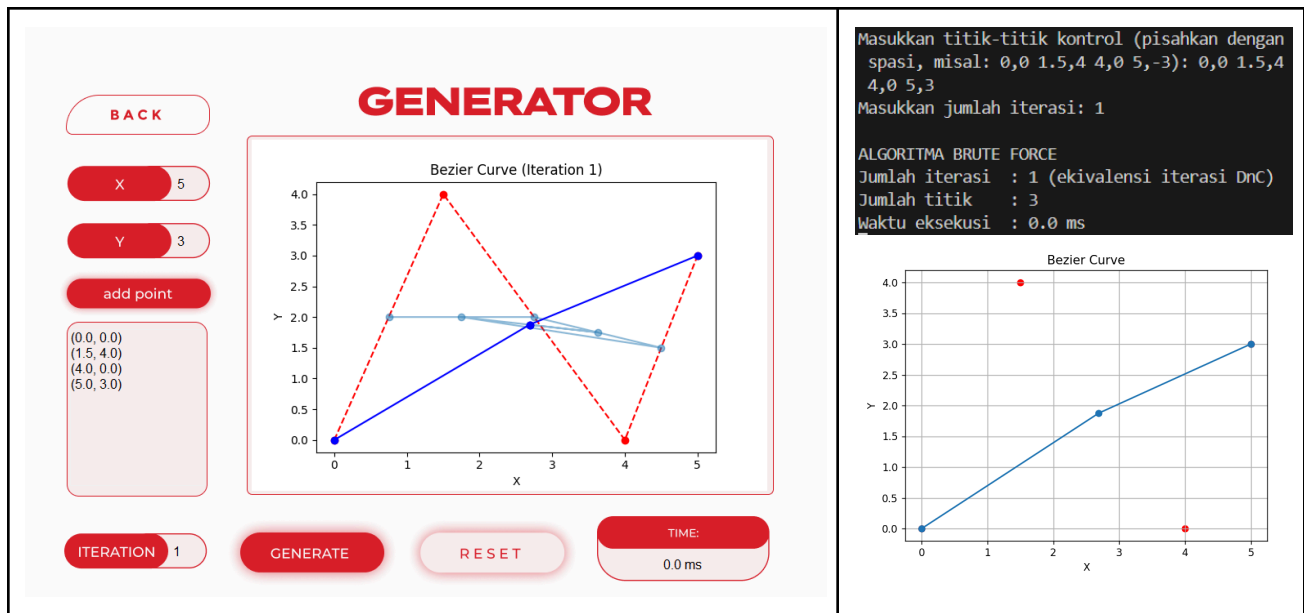
Soal 7 :

Titik: (0,0), (1.5,4), (4,0) (5,3) [4 titik]

Iterasi: 13

Hasil Divide and Conquer

Referensi Brute Force



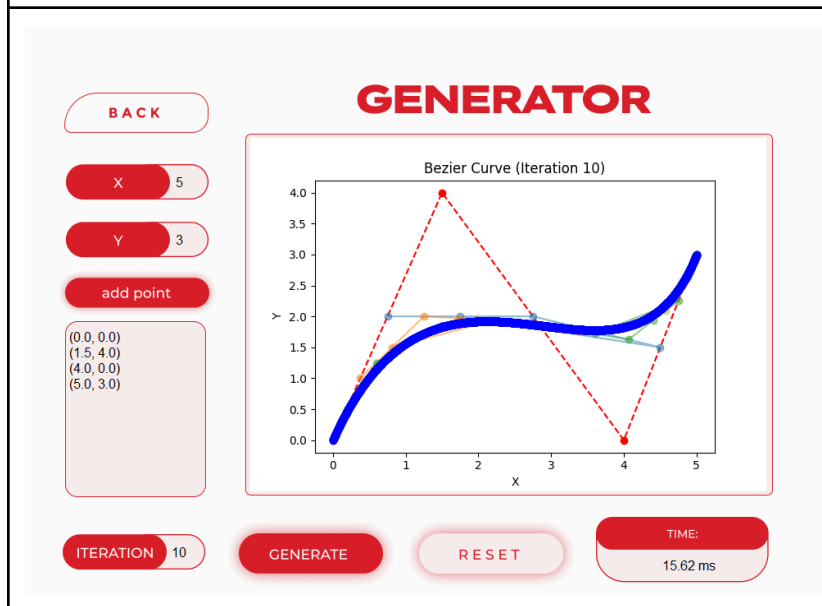
Keterangan : Solusi yang dihasilkan kedua algoritma adalah sama. Namun, dalam jumlah titik dan iterasi ini, belum dapat dibandingkan waktu eksekusi kedua algoritma.

Soal 8 :

Titik: (0,0), (1.5,4), (4,0) (5,3) [4 titik]

Iterasi: 10

Hasil Divide and Conquer



Referensi Brute Force

Masukkan titik-titik kontrol (pisahkan dengan spasi, misal: 0,0 1.5,4 4,0 5,-3): 0,0 1.5,4 4,0 5,3

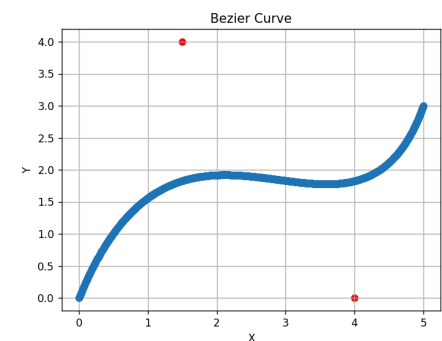
Masukkan jumlah iterasi: 10

ALGORITMA BRUTE FORCE

Jumlah iterasi : 10 (ekivalensi iterasi DnC)

Jumlah titik : 1025

Waktu eksekusi : 20.0 ms



Keterangan : Solusi yang dihasilkan kedua algoritma adalah sama. Dalam jumlah iterasi ini,

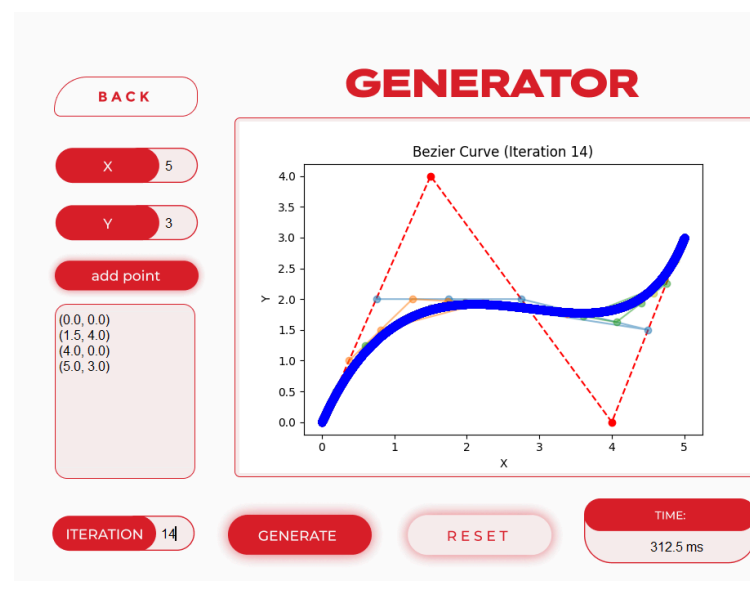
dapat terlihat bahwa waktu eksekusi algoritma divide and conquer tetap sedikit lebih cepat dibanding brute force dengan selisih 4,38 ms, sama seperti pada 3 titik kontrol.

Soal 9 :

Titik: (0,0), (1.5,4), (4,0) (5,3) [4 titik]

Iterasi: 14

Hasil Divide and Conquer

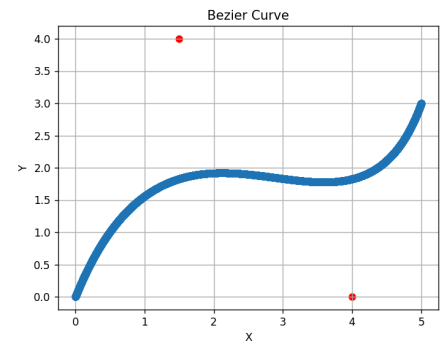


Referensi Brute Force

Masukkan titik-titik kontrol (pisahkan dengan spasi, misal: 0,0 1.5,4 4,0 5,-3): 0,0 1.5,4 4,0 5,3
Masukkan jumlah iterasi: 14

ALGORITMA BRUTE FORCE

Jumlah iterasi : 14 (ekivalensi iterasi DnC)
Jumlah titik : 16385
Waktu eksekusi : 330.0 ms



Keterangan : Solusi yang dihasilkan kedua algoritma adalah sama. Dengan jumlah iterasi yang sama seperti soal 6 namun jumlah titik bertambah 1 lebih banyak, dapat terlihat bahwa waktu eksekusi algoritma divide and conquer menjadi bertambah. Jika dibandingkan dengan brute force, sedikit lebih cepat dengan selisih 17,5 ms.

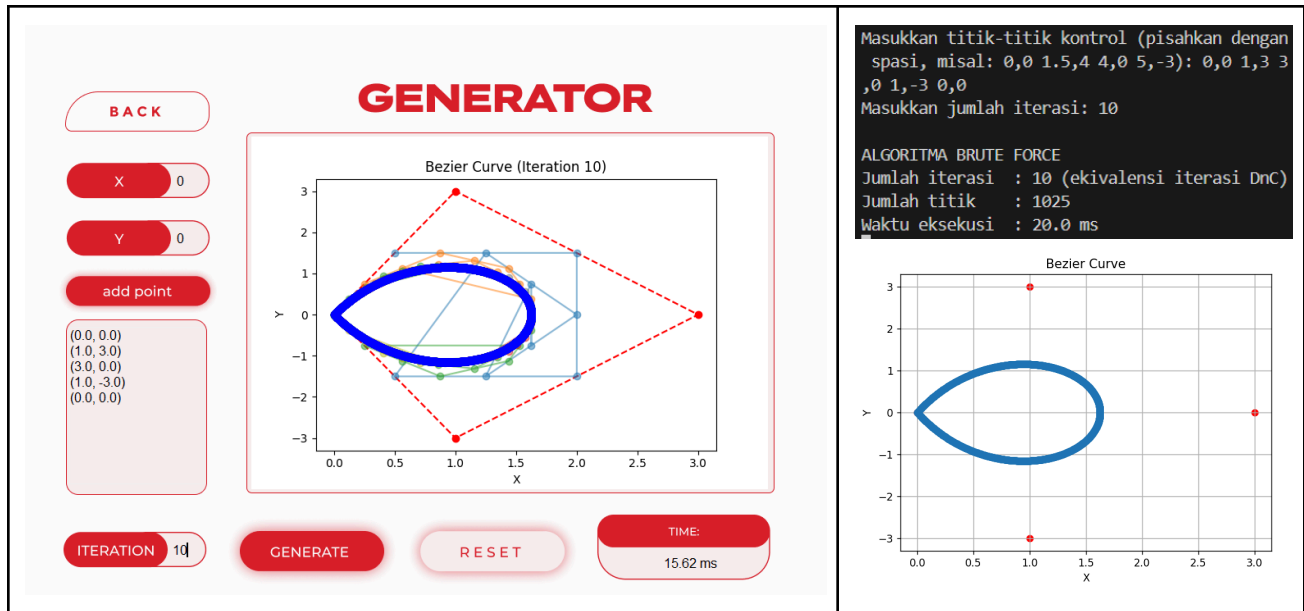
Soal 10 :

Titik: (0,0), (1,3), (3,0) (1,-3), (0,0) [5 titik]

Iterasi: 10

Hasil Divide and Conquer

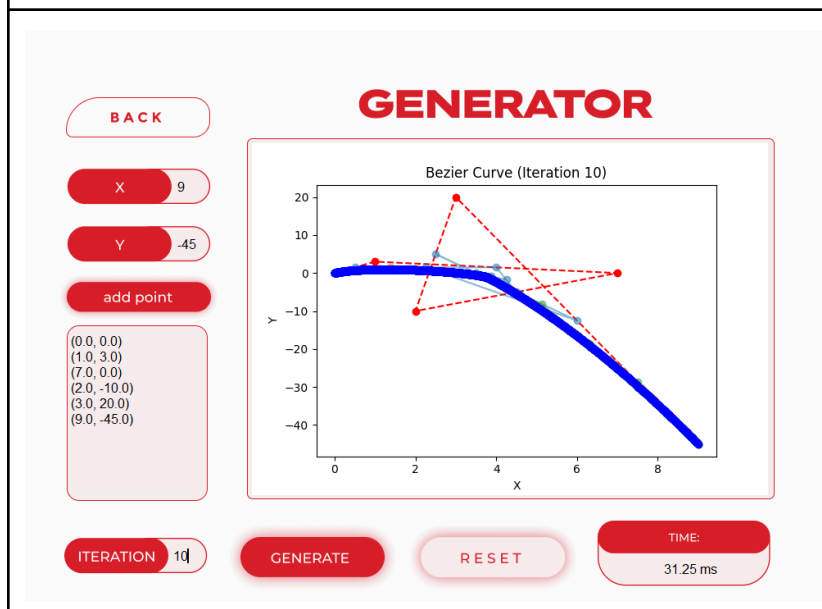
Referensi Brute Force



Keterangan : Solusi yang dihasilkan kedua algoritma adalah sama. Dengan jumlah iterasi 10 namun jumlah titik bertambah 1 lebih banyak, dapat terlihat bahwa waktu eksekusi algoritma divide and conquer tetap sama seperti semua titik beriterasi 10 lainnya. Jika dibandingkan dengan brute force, sedikit lebih cepat dengan selisih 4,38 ms (sama seperti lainnya).

Soal 11 :
Titik: (0,0), (1,3), (7,0) (2,-10), (3,20), (9,-45) [6 titik]
Iterasi: 10

Hasil Divide and Conquer



Referensi Brute Force

Masukkan titik-titik kontrol (pisahkan dengan spasi, misal: 0,0 1.5,4 4,0 5,-3): 0,0 1,3 7,0 2,-10 3,20 9,-45

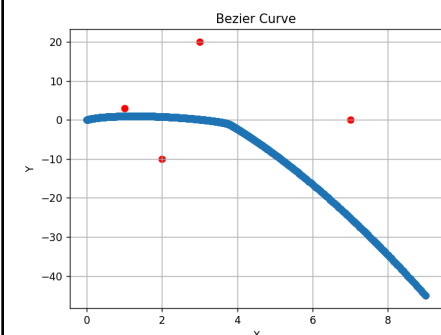
Masukkan jumlah iterasi: 10

ALGORITMA BRUTE FORCE

Jumlah iterasi : 10 (ekivalensi iterasi DnC)

Jumlah titik : 1025

Waktu eksekusi : 50.0 ms



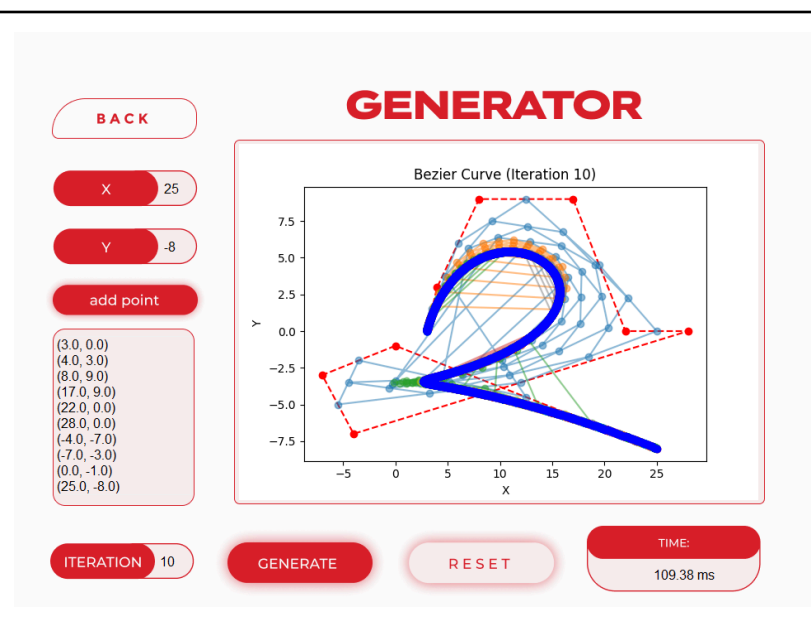
Keterangan : Solusi yang dihasilkan kedua algoritma adalah sama. Dengan jumlah iterasi 10 namun jumlah titik bertambah 1 lebih banyak, dapat terlihat bahwa waktu eksekusi algoritma divide and conquer menjadi bertambah. Jika dibandingkan dengan brute force, sedikit lebih cepat dengan selisih 18,75 ms.

Soal 12 :

Titik: (3,0), (4,3), (8,9), (17,9), (22,0), (28,0), (-4,-7), (-7,-3), (0,-1), (25,-8) [10 titik]

Iterasi: 10

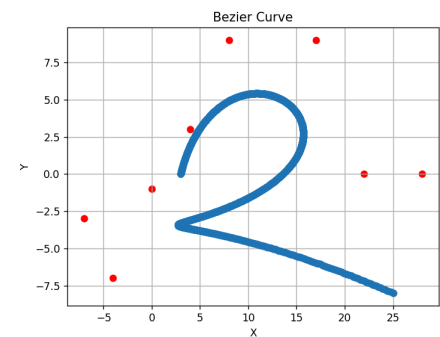
Hasil Divide and Conquer



Referensi Brute Force

Masukkan titik-titik kontrol (pisahkan dengan spasi, misal: 0,0 1.5,4 4,0 5,-3): 3,0 4,3 8,9 17,9 22,0 28,0 -4,-7 -7,-3 0,-1 25,-8
Masukkan jumlah iterasi: 10

ALGORITMA BRUTE FORCE
Jumlah iterasi : 10 (ekivalensi iterasi Dnc)
Jumlah titik : 1025
Waktu eksekusi : 170.0 ms



Keterangan : Solusi yang dihasilkan kedua algoritma adalah sama. Dengan jumlah iterasi 10 namun jumlah titik bertambah 4 lebih banyak, dapat terlihat bahwa waktu eksekusi algoritma divide and conquer menjadi jauh bertambah. Jika dibandingkan dengan brute force, lebih cepat dengan selisih 60,62 ms.

3.3 Perbandingan Kompleksitas Algoritma Brute Force dengan Divide and Conquer

Dalam membandingkan algoritma *Divide and Conquer* dan *brute force* digunakan parameter yang sama yaitu jumlah iterasi. Pada algoritma *brute force*, diperlukan interpolasi linear antara semua pasangan titik berurutan (kurva Bézier linier), untuk mendapatkan titik-titik ini akan menggunakan parameter iterasi. Pada setiap iterasi, jumlah titik yang dihasilkan berlipat dua, sehingga total titik yang diperoleh adalah $2^k + 1$. Pada $k = 10$ banyak operasi yang dilakukan algoritma *brute force*

adalah 1025 operasi, dimana jika diplot akan memberikan kompleksitas yang tinggi dengan fungsi waktu $T(n) = O(2^k)$, dengan n adalah jumlah titik yang dihasilkan. Fungsi ini mengindikasikan bahwa waktu eksekusi meningkat secara eksponensial seiring dengan peningkatan jumlah iterasi.

Sedangkan pada algoritma *divide and conquer*, Setiap iterasi membagi titik-titik kontrol menjadi dua bagian, menghasilkan kompleksitas $O(\log n)$ dan proses pembuatan titik kontrol tengah memerlukan waktu $O(n)$, sehingga total kompleksitas waktu adalah $O(n \log n)$. Fungsi waktu algoritma *divide and conquer* dengan menggunakan teorema Master: $T(n) = aT(n/b) + f(n)$, di mana n adalah jumlah titik kontrol pada kurva Bézier, $a = 2$ karena dua subproblem dihasilkan pada setiap iterasi, $b = 2$ ukuran masalah berkurang menjadi setengahnya setiap rekursi, dan $f(n) = O(n)$ waktu yang diperlukan untuk membagi dan menggabungkan solusi subproblem. Dengan teorema Master, fungsi kompleksitas menjadi $T(n) = 2T(n/2) + O(n)$, sehingga kompleksitas waktu algoritma ini untuk pembentukan kurva Bézier adalah $O(n \log n)$, di mana n adalah jumlah titik kontrol pada kurva Bézier. Pada algoritma *divide and conquer*, fungsi rekursif untuk membagi titik kontrol menjadi dua bagian, kemudian melakukan penggabungan. Fungsi ini menunjukkan menunjukkan bahwa waktu eksekusi algoritma meningkat secara logaritmik seiring dengan peningkatan jumlah titik kontrol.

Algoritma *bruteforce* memiliki kompleksitas waktu yang eksponensial, yang berarti waktu eksekusi akan meningkat secara drastis seiring dengan peningkatan jumlah iterasi. Algoritma *Divide and Conquer*, meskipun memiliki kompleksitas waktu yang lebih rendah, tetapi tetap membutuhkan waktu yang signifikan untuk jumlah titik kontrol yang besar, ini karena kompleksitasnya bergantung pada jumlah titik kontrol. Oleh karena itu, algoritma *Divide and Conquer* lebih efisien daripada *bruteforce*, terutama untuk jumlah titik kontrol yang besar, karena kompleksitasnya lebih rendah dan pertumbuhannya lebih lambat seiring dengan peningkatan jumlah titik kontrol.

3.4 Implementasi Bonus

3.4.1 Implementasi Kurva untuk N Titik Kontrol

Untuk menghasilkan kurva Bézier yang dapat menerima masukan n buah titik kontrol, dilakukan modifikasi terhadap algoritma *divide and conquer* yang digunakan untuk kurva Bézier kuadratik. Dengan melihat pola bahwa dalam setiap dua titik akan dicari titik tengahnya, dan titik-titik tersebut beserta titik awal dan akhir dari masukan akan digunakan untuk rekursi dua bagian

kemudian digabungkan kembali, maka algoritma ini dapat diimplementasikan dengan cara sebagai berikut:

1. Kumpulkan semua titik kontrol dari masukan ke dalam sebuah array points.
2. Buat sebuah fungsi yang menerima masukan array points dan mengembalikan titik kontrol baru hasil titik tengah antardua titik. Fungsi ini memanfaatkan rekursi atas konsep pencarian titik tengah yang sama dengan pembentukan kurva Bézier kuadratik.
3. Gunakan fungsi tersebut untuk melakukan divide and conquer berdasarkan iterasi yang diberikan. Untuk itu, siapkan dua array guna menampung titik-titik yang akan diolah dalam rekursi menjadi dua bagian. Pada bagian kiri, kumpulkan titik awal, titik tengah, dan titik tengah lanjutan sebagai titik akhir sebagaimana dilakukan pada kurva Bézier kuadratik. Begitu juga dilakukan terhadap bagian kanan.
4. Lakukan rekursi fungsi yang sama untuk bagian kiri dahulu lalu dilanjut dengan rekursi untuk bagian kanan. Rekursi ini akan terus membagi titik ke titik menjadi bagian yang lebih kecil sesuai dengan iterasi masukan hingga iterasi mencapai 0 (selesai).
5. Gabungkan hasil dari kedua submasalah dengan cara mengkonkatenasi titik-titik hasil rekursi bagian kiri dengan bagian kanan.
6. Hasil penggabungan ini adalah titik-titik pada kurva Bézier yang dibentuk sesuai jumlah iterasinya.

3.4.2 Implementasi Visualisasi Pembuatan Kurva

Implementasi visualisasi pembuatan kurva pada kode di atas dilakukan menggunakan library Matplotlib dan Tkinter dalam bahasa pemrograman Python.

1. Tkinter GUI

Tkinter GUI digunakan untuk membuat antarmuka pengguna (GUI) yang terdiri dari dua halaman, yaitu halaman utama (HOME) dan halaman generator (GENERATOR). Pada halaman utama, terdapat tombol "Try Me" yang akan mengarahkan pengguna ke halaman generator. Di halaman generator, pengguna dapat memasukkan titik-titik kontrol, mengatur jumlah iterasi, serta melakukan operasi tambah titik, generate kurva, reset titik, dan menampilkan waktu generate.

2. Visualisasi Kurva Bézier

Visualisasi Kurva Bézier menggunakan algoritma Divide and Conquer, di mana kurva dibagi menjadi segmen-segmen kecil untuk setiap iterasi. Setiap segmen kurva ditentukan oleh titik-titik kontrol. Proses pembentukan kurva Bézier divisualisasikan menggunakan animasi Matplotlib. Pada setiap iterasi, kurva Bézier diperbarui dengan menambahkan lebih banyak titik kontrol di antara titik-titik kontrol yang sudah ada. Proses ini diulangi sebanyak iterasi yang dimasukkan oleh pengguna. Dalam visualisasi ini, terdapat dua nested list of array yang digunakan untuk menyimpan informasi titik-titik yang diperoleh setiap iterasi, yaitu `points` dan `all_points`. `Points` berisi array yang berisi kumpulan titik hasil proses Divide and Conquer, sementara `all_points` berisi array yang menyimpan semua titik kontrol di setiap iterasinya. Plot kurva dari `points` dan `all_points` dibedakan dengan tingkat opacity `all_points` yang hanya 50% agar kurva hasil Divide and Conquer (warna biru) dapat terlihat jelas. Selain itu, titik kontrol masukan juga di-plot dengan kurva garis putus-putus berwarna merah.

3. Interaksi Pengguna

Interaksi pengguna memungkinkan pengguna untuk memasukkan titik kontrol melalui GUI. Setelah memasukkan titik-titik kontrol, pengguna dapat mengatur jumlah iterasi untuk kurva Bézier. Ketika pengguna menekan tombol "Generate", kurva Bézier akan dibuat dan divisualisasikan sesuai dengan jumlah iterasi yang telah ditentukan. Pengguna juga dapat mereset titik-titik kontrol agar dapat memulai dari awal. Setelah proses pembuatan kurva Bézier selesai, waktu eksekusi akan diukur dan ditampilkan kepada pengguna, dimulai dari awal hingga akhir proses pembuatan kurva Bézier. Pesan error akan muncul jika pengguna tidak memasukkan titik kontrol atau jika jumlah titik kontrol kurang dari 3, dan jika pengguna tidak mengisi jumlah iterasi sebelum menekan tombol "Generate".

BAB IV LAMPIRAN

4.1 Link Repository

Github: https://github.com/NopalAul/Tucil2_13522072_13522074

4.2 Checklist

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.	✓	

REFERENSI

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-\(2022\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf)

<https://repository.unikom.ac.id/37037/1/BruteForce%28bagian%201%29.pdf>

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-\(2024\)-Bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian1.pdf)

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-\(2024\)-Bagian4.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian4.pdf)

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Aplikasi-Divide-and-Conquer-2020.pdf>