

## **LAPORAN TUGAS KECIL 2**

**IF2211 STRATEGI ALGORITMA**

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



Disusun oleh:

Ahmad Thoriq Saputra (13522141)  
Rafif Ardhinto Ichwantoro (13522159)

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

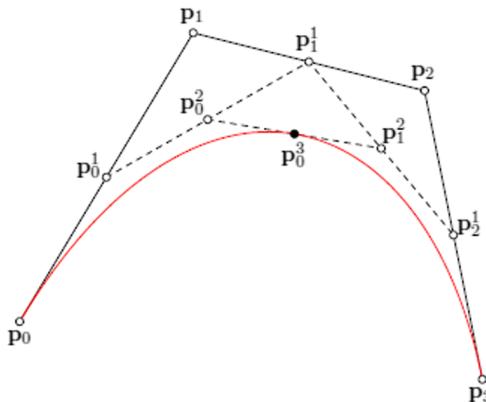
**BANDUNG  
2024**

## DAFTAR ISI

<b>DAFTAR ISI.....</b>	<b>2</b>
<b>BAB I</b>	
<b>DESKRIPSI MASALAH.....</b>	<b>3</b>
<b>BAB II</b>	
<b>TEORI SINGKAT.....</b>	<b>5</b>
<b>BAB III</b>	
<b>ANALISIS DAN IMPLEMENTASI ALGORITMA.....</b>	<b>6</b>
3.1 Implementasi Algoritma Brute force.....	6
3.2 Implementasi Algoritma Divide and conquer.....	6
3.3 Analisis Kompleksitas Waktu Algoritma.....	7
<b>BAB IV</b>	
<b>PENJELASAN PROGRAM.....</b>	<b>8</b>
4.1 Program Divide and Conquer.....	8
4.2 Program Brute Force.....	11
<b>BAB V</b>	
<b>EKSPERIMEN.....</b>	<b>12</b>
5.1 Algoritma Divide and Conquer.....	12
5.2 Algoritma Brute Force.....	16
<b>BAB VI</b>	
<b>PROGRAM BONUS.....</b>	<b>21</b>
6.1 Membuat kurva dengan N titik kontrol.....	21
6.2 Visualisasi kurva pada GUI tkinter.....	21
<b>BAB VII</b>	
<b>KESIMPULAN.....</b>	<b>26</b>
<b>DAFTAR PUSTAKA.....</b>	<b>27</b>
<b>LAMPIRAN.....</b>	<b>28</b>

## BAB I

### DESKRIPSI MASALAH



**Gambar 1.** Pembentukan Kurva Bézier

(Sumber:[https://www.researchgate.net/figure/Interpolation-of-cubic-Bezier-curve-using-four-control-points\\_fig2\\_349801462](https://www.researchgate.net/figure/Interpolation-of-cubic-Bezier-curve-using-four-control-points_fig2_349801462))

Kurva Bézier adalah salah satu kurva halus yang sering digunakan dalam berbagai aplikasi seperti desain grafis, animasi, dan manufaktur. Kurva ini dibentuk dengan menghubungkan serangkaian titik kontrol yang menentukan bentuk dan arah kurva. Keuntungan utama dari penggunaan kurva Bézier adalah kemudahan dalam mengubah dan memanipulasi desain, sehingga menghasilkan hasil akhir yang presisi dan sesuai dengan kebutuhan.

Sebuah kurva Bézier didefinisikan oleh setidaknya dua titik kontrol, dengan titik kontrol pertama dan terakhir selalu menjadi ujung dari kurva. Titik kontrol di antara kedua titik tersebut (jika ada) umumnya tidak terletak pada kurva yang terbentuk. Pada gambaran umum, titik-titik kontrol ini dinyatakan sebagai  $P_0$  sampai  $P_n$ , dengan  $n$  disebut sebagai order kurva. Contohnya, untuk kurva linier, ordernya adalah 1, sedangkan untuk kurva kuadratik, ordernya adalah 2, dan seterusnya.

Untuk membentuk sebuah kurva Bézier, misalnya sebuah kurva linier antara dua titik  $P_0$  dan  $P_1$ , parameter  $t$  digunakan untuk menentukan seberapa jauh titik pada kurva tersebut dari  $P_0$  ke  $P_1$ . Misalnya, ketika  $t = 0.25$ , titik pada kurva adalah seperempat jalan dari  $P_0$  ke  $P_1$ . Proses yang sama berlaku untuk pembentukan kurva Bézier dengan order yang lebih tinggi, seperti kuadratik atau kubik.

Untuk menghasilkan kurva Bézier dengan algoritma yang efisien, dapat digunakan pendekatan titik tengah berbasis divide and conquer. Dengan pendekatan ini, pembentukan kurva

Bézier dapat dibagi menjadi langkah-langkah yang lebih kecil, sehingga memudahkan dalam mengelola kompleksitasnya. Hal ini sangat penting terutama ketika jumlah titik kontrol yang digunakan semakin banyak, karena persamaan yang terbentuk dapat menjadi sangat panjang dan rumit.

Dengan implementasi algoritma titik tengah berbasis divide and conquer, pembuatan kurva Bézier dapat dilakukan dengan lebih efisien dan cepat, sehingga memungkinkan untuk digunakan dalam berbagai aplikasi yang memerlukan kurva yang halus dan presisi.

## BAB II

### TEORI SINGKAT

Algoritma Divide and Conquer adalah sebuah paradigma algoritma yang efektif dalam menyelesaikan masalah kompleks dengan membagi masalah tersebut menjadi submasalah yang lebih kecil, menyelesaikan sub masalah tersebut secara rekursif, dan menggabungkan solusi dari sub masalah tersebut untuk mendapatkan solusi akhir dari masalah asli. Paradigma ini terdiri dari tiga langkah utama: *Divide*, *Conquer*, dan *Combine*.

Langkah pertama, *Divide*, melibatkan pembagian masalah asli menjadi submasalah yang lebih kecil. Proses ini umumnya dilakukan secara rekursif hingga tidak ada lagi submasalah yang dapat dibagi lebih lanjut. Setiap submasalah harus mewakili bagian dari masalah asli.

Langkah kedua, *Conquer*, adalah tahap di mana submasalah diselesaikan secara rekursif. Pada tahap ini, sub masalah dianggap "selesai" secara independen.

Langkah terakhir, *Combine*, melibatkan penggabungan solusi dari sub masalah yang telah diselesaikan menjadi solusi akhir dari masalah asli. Tahap ini dilakukan secara rekursif hingga solusi akhir diperoleh.

Algoritma Divide and Conquer sangat berguna dalam mengurangi kompleksitas waktu karena memungkinkan penyelesaian masalah yang lebih besar dengan cara yang efisien. Contohnya adalah algoritma-algoritma seperti *Quicksort*, *Merge Sort*, dan *Strassen's Algorithm*, yang memanfaatkan pendekatan ini untuk meningkatkan efisiensi perhitungan mereka. Penting untuk memilih antara algoritma Divide and Conquer dan Dynamic Programming berdasarkan seberapa sering submasalah yang sama dievaluasi. Jika submasalah sering dievaluasi, Dynamic Programming atau Memoization mungkin lebih cocok. Namun, jika submasalah tidak dievaluasi banyak kali, maka Divide and Conquer adalah pilihan yang lebih baik.

## **BAB III**

### **ANALISIS DAN IMPLEMENTASI ALGORITMA**

#### **3.1 Implementasi Algoritma Brute force**

1. Program menerima inputan beberapa titik
2. Program menerima inputan jumlah iterasi
3. Penentuan jumlah titik dilakukan dengan fungsi dari library python yaitu linspace untuk membuat sebuah array dengan rentang tertentu, dalam kasus ini rentang yang digunakan 0 sampai 1 dan t sebagai inisialisasi. Menggunakan rumus jumlah titik  $2^n+1$  dengan n jumlah iterasi.
4. Membuat array 2D sebagai titik curve berukuran sesuai dengan jumlah titik yang telah ditentukan.
5. Proses pembuatan titik diawali dengan rumus  $(1-t[i])*P[j] + t*P[j+1]$ , P sebagai titik dan t merupakan distribusi nilai rentang t.setiap nilai disimpan dalam array baru dan terpisah untuk absis dan ordinat. Misalkan hasil tersebut disimpan dalam array Q maka langkah selanjutnya menggunakan rumus  $(1-t[i])*(Qx[j]) + t[i]*(Qx[j+1])$  proses tersebut terus dilakukan sampai tersisa 1 nilai yang merupakan titik kurva bezier. Iterasi i merupakan proses perpindahan titik kurva satu ke titik kurva lainnya.

#### **3.2 Implementasi Algoritma Divide and conquer**

1. Program menerima inputan beberapa titik
2. Program menerima inputan jumlah iterasi
3. Fungsi kurva bezier memiliki parameter array of titik, iterasi dan isleft yang menandakan bagian kanan atau kiri.
4. Proses rekursif dilakukan diawali dengan mencari titik tengah antara garis lurus titik koordinat lalu titik tersebut disimpan dalam sebuah array sebagai titik Q0 dan Q1
5. Dari titik tengah yang didapatkan proses sebelumnya dicari titik tengahnya lalu disimpan dalam array sebagai titik b
6. Menggabungkan 2 array proses sebelumnya dengan algoritma meletakan elemen Q0 dan Q1 pada indeks genap dan titik b pada indeks ganjil

7. Membagi array tersebut menjadi 2 bagian yaitu left dan right, dalam left dilakukan penambahan elemen P0 pada head array, pada bagian right ditambahkan Pn-1, pada bagian tail array dengan n jumlah titik.
8. Fungsi mengembalikan pemanggilan fungsi dengan parameter array left ditambah fungsi dengan parameter right dengan pengurangan iterasi -1
9. Basis fungsi berkondisi ketika iterasi = 0, akan mengembalikan array dari parameter fungsi dengan pengambilan elemen dengan indeks ganjil. Mencegah duplikasi titik, pada bagian fungsi right proses iterasi pengambilan elemen dimulai pada indeks 2.

### 3.3 Analisis Kompleksitas Waktu Algoritma

Dalam analisis kompleksitas waktu Algoritma-Algoritma tersebut, kita dapat mempertimbangkan beberapa faktor utama yang mempengaruhi kinerja mereka. Algoritma dengan implementasi divide and conquer menggunakan pendekatan rekursif dengan pembagian daerah untuk menghitung kurva Bezier, sementara Algoritma brute force menggunakan rumus Bezier langsung tanpa rekursi.

Pada kasus Algoritma Divide and conquer ini membagi kurva menjadi dua bagian dengan setiap iterasi, yang menghasilkan logaritma basis dua ( $\log_2(n)$ ) panggilan fungsi rekursif, di mana n adalah jumlah iterasi. Pada setiap iterasi melakukan perintah yang sama sehingga dalam notasi O kompleksitas waktunya  $O(n * \log_2(n))$ .

Pada kasus implementasi dengan algoritma Brute force menggunakan rumus matematika Bezier secara langsung untuk menghitung titik-titik pada kurva Bezier, tanpa membagi kurva menjadi bagian-bagian. Operasi utama dalam Algoritma ini adalah perhitungan titik-titik pada kurva Bezier, yang memakan waktu tetap untuk setiap iterasi. Oleh karena itu, kompleksitas waktu Algoritma ini dapat dianggap sebagai  $O(n)$ , di mana n adalah jumlah iterasi.

Dari kompleksitas waktu yang didapatkan dapat disimpulkan penyelesaian pembentukan kurva bezier lebih optimal menggunakan Algoritma brute force.

## BAB IV

### PENJELASAN PROGRAM

#### 4.1 Program *Divide and Conquer*

```
● ● ●

def midpoint(p0, p1):
    return [(p0[0]+p1[0])/2, (p0[1]+p1[1])/2]

def bezier_curve_dnc(points, iterations, isleft):
    if iterations == 0 and isleft == 0:
        points = [points[i] for i in range(len(points)) if i % 2 == 0]
        return points
    if iterations == 0 and isleft == 1:
        points = [points[i] for i in range(1,len(points)) if i % 2 == 0]
        return points
    else:
        new_points = []
        for i in range(len(points)-1):
            new_points.append(midpoint(points[i], points[i+1]))
        midpoints = []
        for i in range(len(new_points)-1):
            midpoints.append(midpoint(new_points[i], new_points[i+1]))
        new_array = []
        for i in range(len(new_points) + len(midpoints)):
            if i%2 == 0:
                new_array.append(new_points[int(i/2)])
            else:
                new_array.append(midpoints[int(i/2)])
        left = [points[0]]
        for i in range(len(new_array)//2+1):
            left.append(new_array[i])
        if iterations == 1 and isleft == 1:
            left.pop(0)
            left.pop(0)
        right = []
        for i in range(len(new_array)//2, len(new_array)):
            right.append(new_array[i])
        right.append(points[-1])

    return bezier_curve_dnc(left, iterations-1,0) + bezier_curve_dnc(right, iterations-1,1)
```

Gambar 2. Snippet file divideandconquer.py

Penjelasan fungsi:

1. Midpoint dengan parameter 2 titik berfungsi untuk mencari titik tengah antara dua titik.
2. bezier\_curve\_dnc dengan parameter titik, jumlah iterasi dan isleft sebagai penanda kurva bagian kanan atau kiri. Fungsi menjalankan proses pembentukan kurva bezier dengan cara DNC,

## 1. Program *Divide and Conquer* Genap

```
● ● ●

def midpoint(p0, p1):
    return [(p0[0]+p1[0])/2, (p0[1]+p1[1])/2]

def bezier_curve_dncG(points,iterations, isleft,iter):
    if iterations == 0 and isleft == 0:
        points = [points[i] for i in range(len(points)) if i % 2 == 0]
        return points
    elif iterations == 0 and isleft == 1:
        points = [points[i] for i in range(2,len(points)) if i % 2 == 0]
        return points
    else:
        new_points = []
        for i in range(len(points)-1):
            new_points.append(midpoint(points[i], points[i+1]))
        midpoints = []
        for i in range(len(new_points)-1):
            midpoints.append(midpoint(new_points[i], new_points[i+1]))
        new_array = []
        print('iterasi ke', iter)
        for i in range(len(new_points) + len(midpoints)):
            if i%2 == 0:
                new_array.append(new_points[int(i/2)])
            else:
                new_array.append(midpoints[int(i/2)])
        left = []
        if isleft == 0:
            left = [points[0]]
        for i in range(len(new_array)//2+2):
            left.append(new_array[i])
        if isleft == 1:
            left.pop(0)
        right = []

        for i in range(len(new_array)//2-1, len(new_array)):
            right.append(new_array[i])
        if iter >0:
            right.pop(0)
        right.append(points[-1])

    return bezier_curve_dncG(left, iterations-1,0,iter+1) + bezier_curve_dncG(right, iterations-1,1,iter+1)
```

Gambar 3. Snippet file divideandconquergenap.py

Penjelasan fungsi:

1. Midpoint dengan parameter 2 titik berfungsi untuk mencari titik tengah antara dua titik.
2. bezier\_curve\_dncG dengan parameter titik, jumlah iterasi, isleft sebagai penanda kurva bagian kanan atau kiri dan iter menyatakan proses iterasi ke-iter pada awal program

diinisialisasi 0. Fungsi menjalankan proses pembentukan kurva bezier yang memiliki titik kontrol berjumlah genap dengan cara DNC,

#### 4.2 Program *Brute Force*

```
import numpy as np

def bezier_curve_bf(points, iterations):
    iterations = pow(2, iterations)+1
    t = np.linspace(0, 1, iterations)
    t = t.tolist()
    curve = [[0 for _ in range(2)] for _ in range(iterations)]

    for i in range(iterations):
        Qx = []
        Qy = []
        for j in range(len(points) - 1):
            tempx = (1-t[i])*points[j][0] + t[i]*points[j+1]
            tempy = (1-t[i])*points[j][1] + t[i]*points[j+1]
            Qx.append(tempx)
            Qy.append(tempy)
        while len(Qx) > 1:
            Rx = []
            Ry = []
            for j in range(len(Qx) - 1):
                tempx = (1-t[i])*(Qx[j]) + t[i]*(Qx[j+1])
                tempy = (1-t[i])*(Qy[j]) + t[i]*(Qy[j+1])
                Rx.append(tempx)
                Ry.append(tempy)
            Qx = Rx
            Qy = Ry
        curve[i][0] = Qx[0]
        curve[i][1] = Qy[0]
    curve = np.array(curve)
    return curve
```

Gambar 4. Snippet file bruteForce.py

Penjelasan Fungsi:

1. `bezier_curve_bf` dengan parameter titik dan jumlah iterasi. Fungsi menjalankan proses pembentukan kurva Bezier dengan cara brute force.
2. `np.linspace` dengan parameter nilai rentang dan jumlah iterasi. Fungsi membuat array dengan elemen yang berisi distribusi titik dari nilai rentang dan iterasi menentukan banyaknya distribusi.

## BAB V

### EKSPERIMEN

#### 5.1 Algoritma *Divide and Conquer*

Input	Output
<p>Set iteration</p> <p>1</p> <p>Add Points</p> <p>Point 1: 1 4 Delete</p> <p>Point 2: 4 1 Delete</p> <p>Point 3: 6 -6 Delete</p> <p>Add Point</p> <p>Submit</p>	<p>Divide and Conquer Iterations: 1</p> <p>Bezier Curve</p> <p>(1, 0, 4, 0)</p> <p>(4, 0, 1, 0)</p> <p>(6, 0, -6, 0)</p> <p>created in 0.0 seconds</p>

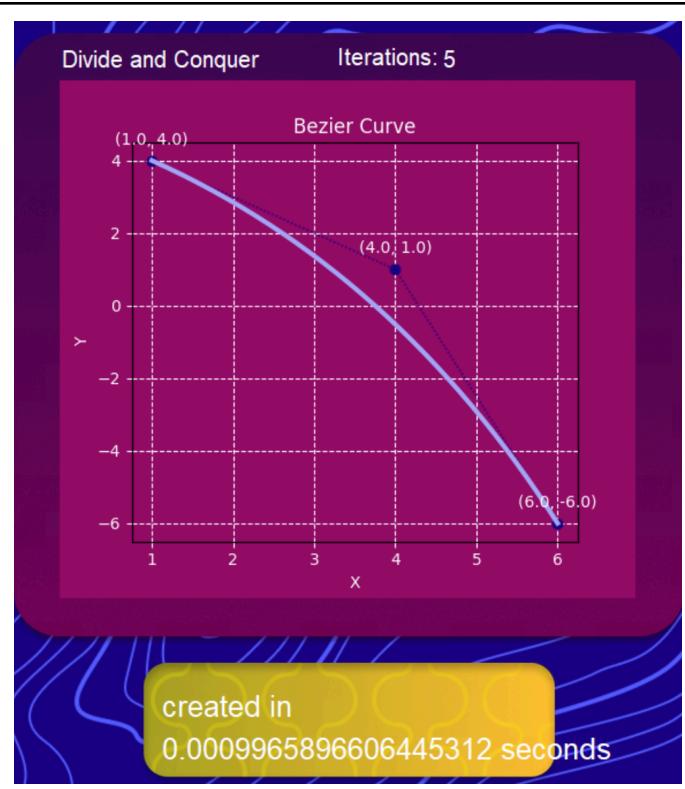
5

Add Points

Point 1:   Delete

Point 2:   Delete

Point 3:   Delete



Set iteration

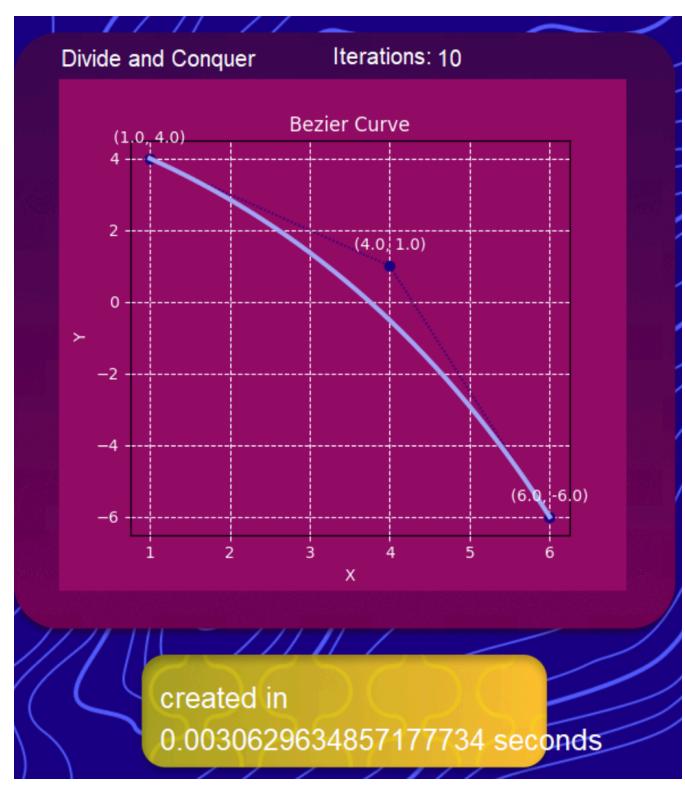
10

Add Points

Point 1:   Delete

Point 2:   Delete

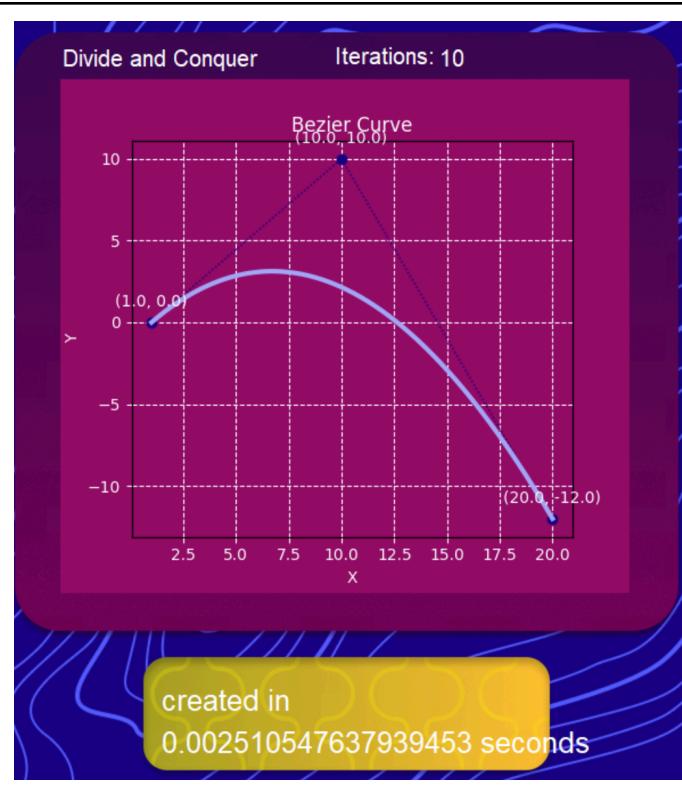
Point 3:   Delete



### Set iteration

Add Points

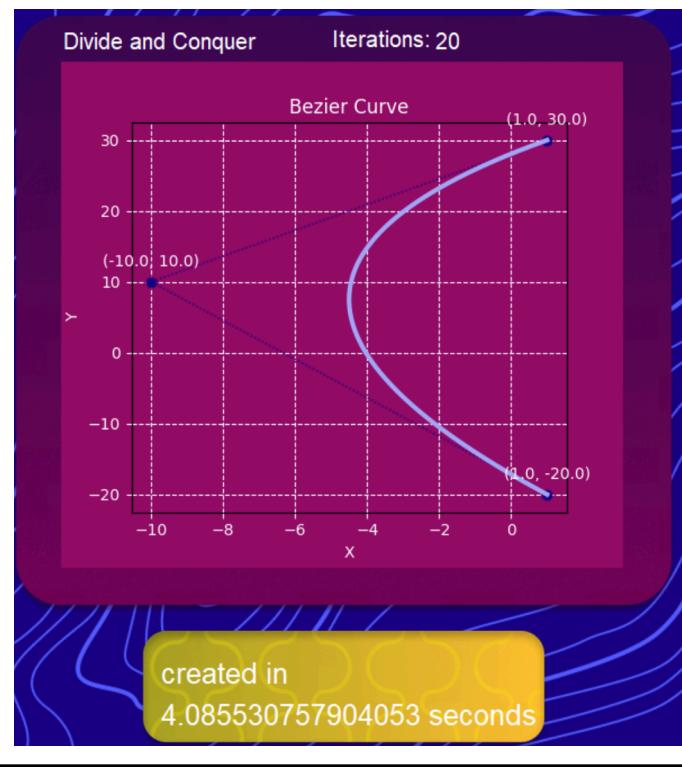
Point 1:	1	0	<input type="button" value="Delete"/>
Point 2:	10	10	<input type="button" value="Delete"/>
Point 5:	20	-12	<input type="button" value="Delete"/>



### Set iteration

Add Points

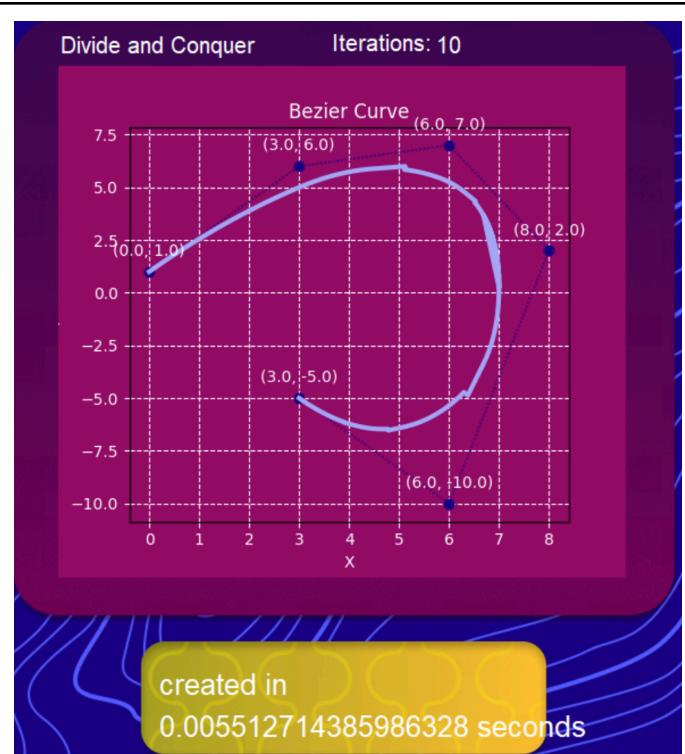
Point 1:	1	30	<input type="button" value="Delete"/>
Point 2:	-10	10	<input type="button" value="Delete"/>
Point 5:	1	-20	<input type="button" value="Delete"/>



### Set iteration

Add Points

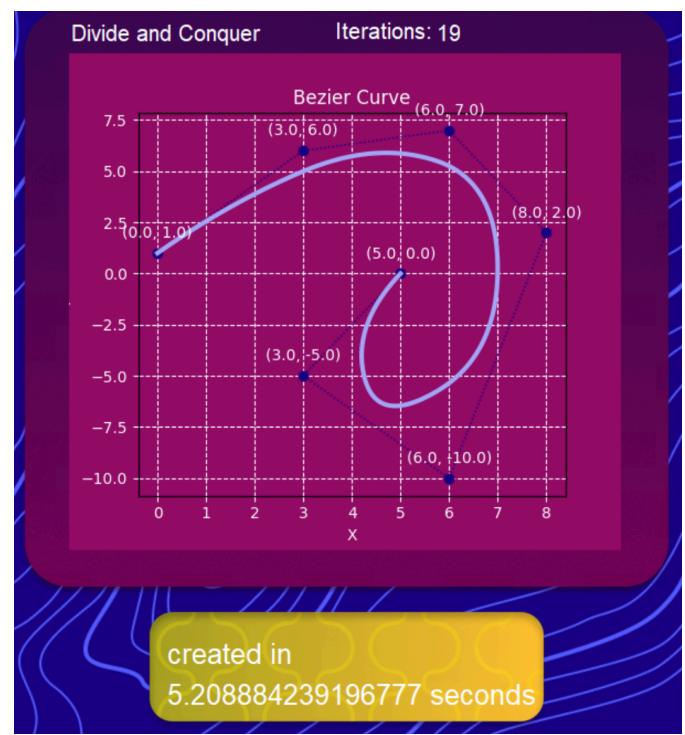
Point 1:	0	1	Delete
Point 2:	3	6	Delete
Point 3:	6	7	Delete
Point 4:	8	2	Delete
Point 5:	6	-10	Delete
Point 6:	3	-5	Delete

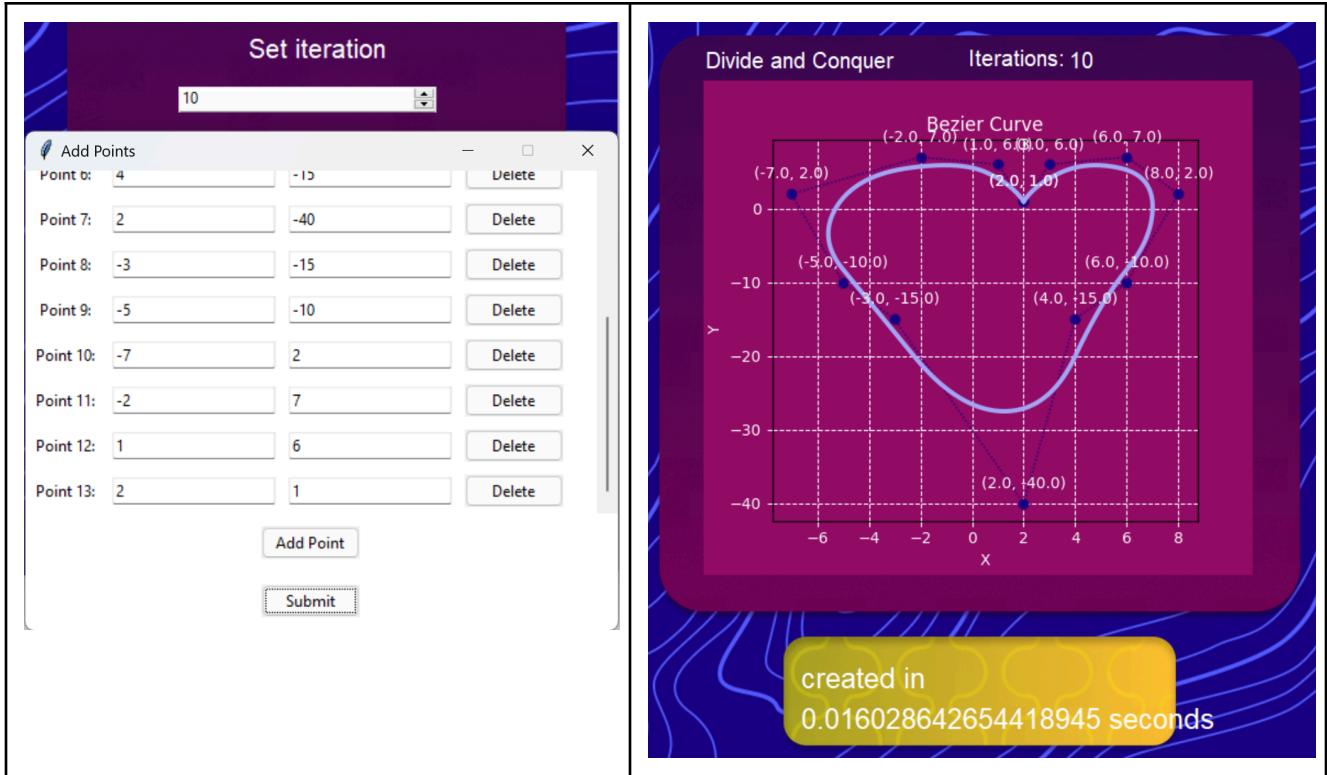


### Set iteration

Add Points

Point 1:	0	1	Delete
Point 2:	3	6	Delete
Point 3:	6	7	Delete
Point 4:	8	2	Delete
Point 5:	6	-10	Delete
Point 6:	3	-5	Delete
Point 7:	5	0	Delete





## 5.2 Algoritma Brute Force

Input	Output
-------	--------

### Set iteration

1

Add Points

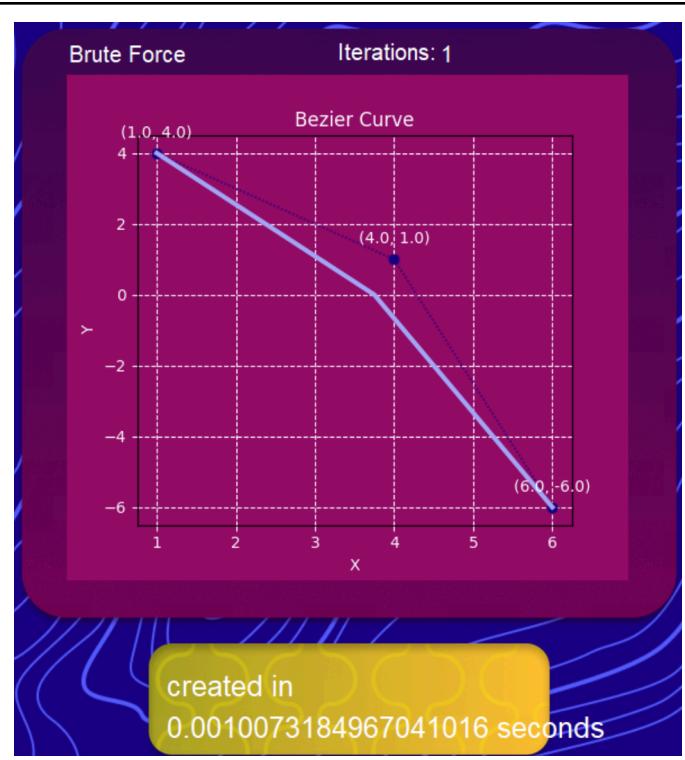
Point 1: 1 4 Delete

Point 2: 4 1 Delete

Point 3: 6 -6 Delete

Add Point

Submit



### Set iteration

5

Add Points

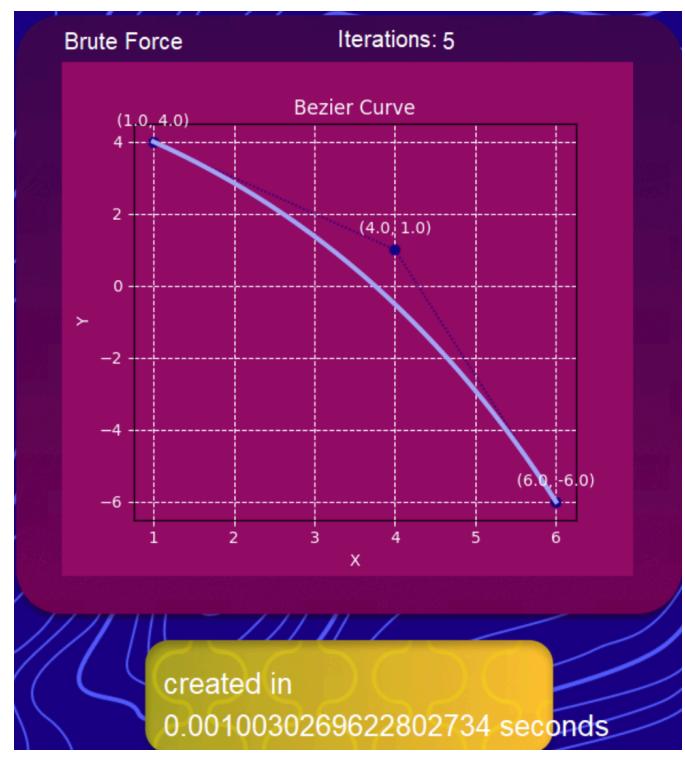
Point 1: 1 4 Delete

Point 2: 4 1 Delete

Point 3: 6 -6 Delete

Add Point

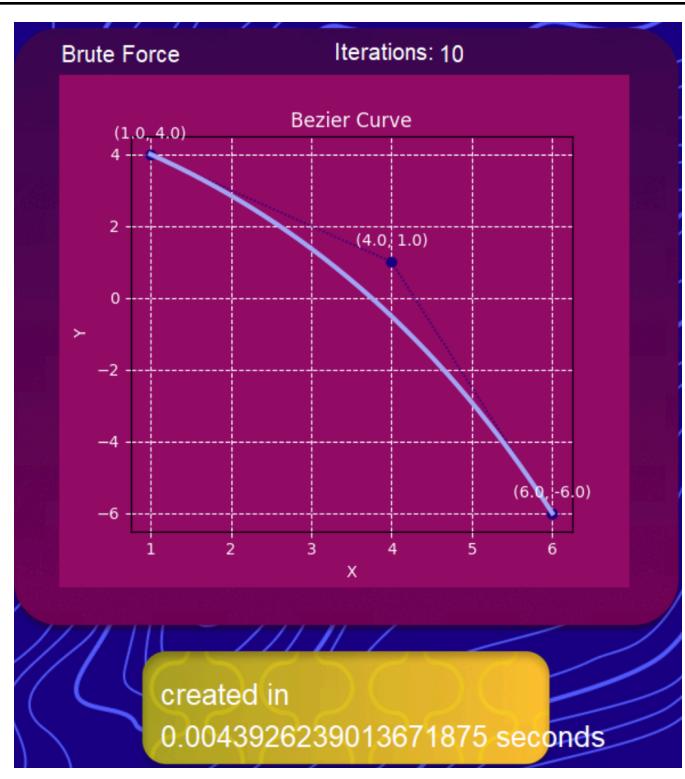
Submit



### Set iteration

Add Points

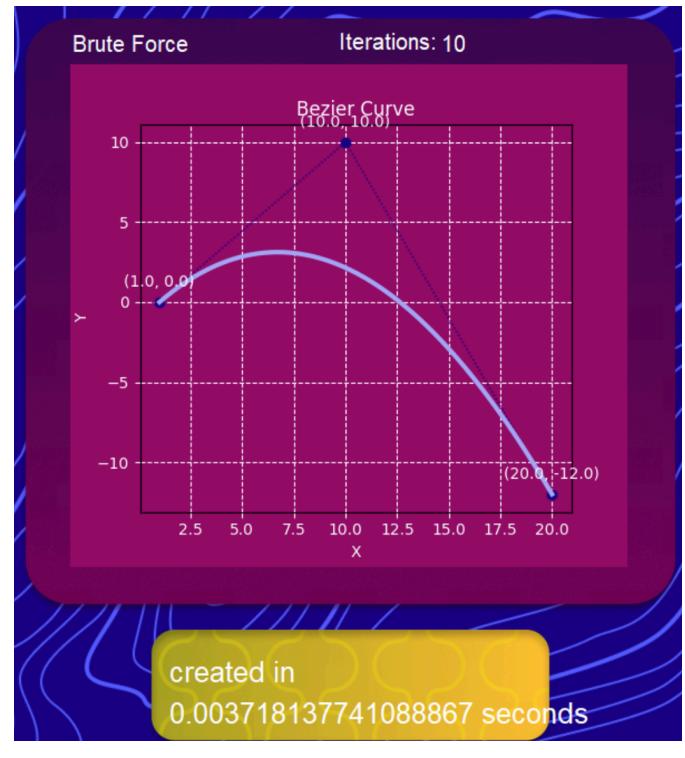
Point 1:	<input type="text" value="1"/>	<input type="text" value="4"/>	<input type="button" value="Delete"/>
Point 2:	<input type="text" value="4"/>	<input type="text" value="1"/>	<input type="button" value="Delete"/>
Point 3:	<input type="text" value="6"/>	<input type="text" value="-6"/>	<input type="button" value="Delete"/>

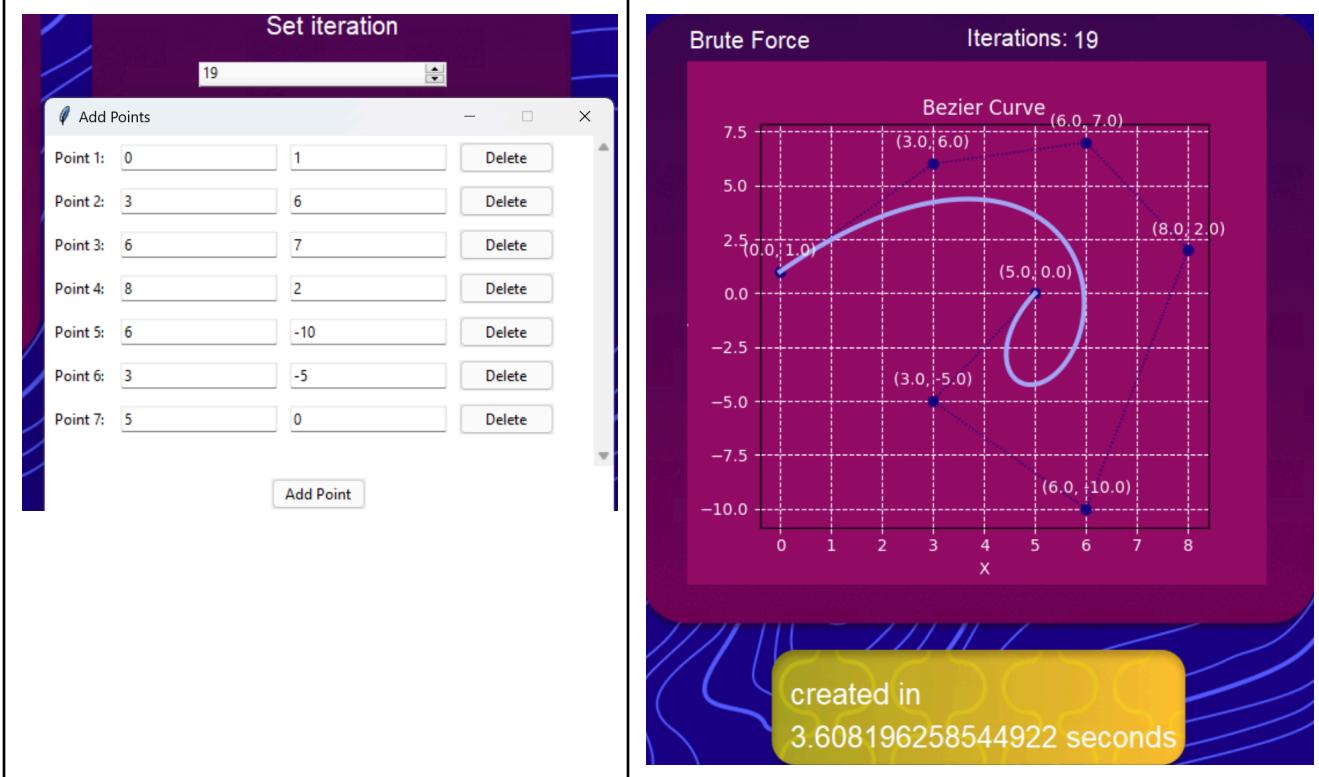
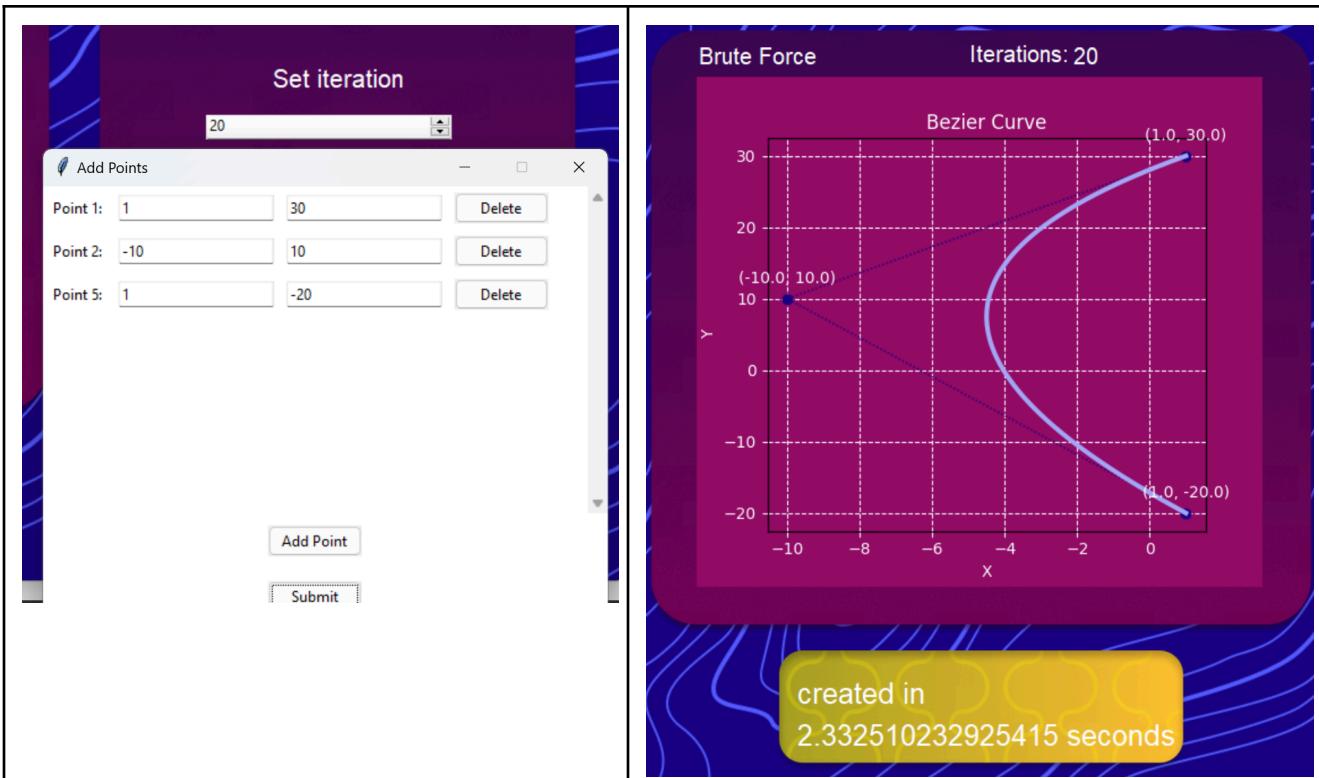


### Set iteration

Add Points

Point 1:	<input type="text" value="1"/>	<input type="text" value="0"/>	<input type="button" value="Delete"/>
Point 2:	<input type="text" value="10"/>	<input type="text" value="10"/>	<input type="button" value="Delete"/>
Point 5:	<input type="text" value="20"/>	<input type="text" value="-12"/>	<input type="button" value="Delete"/>

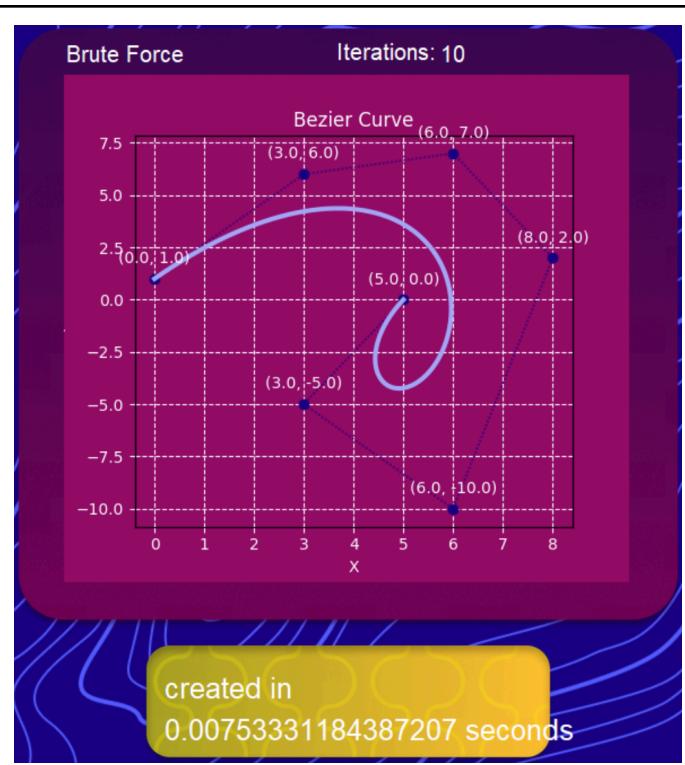




### Set iteration

Add Points

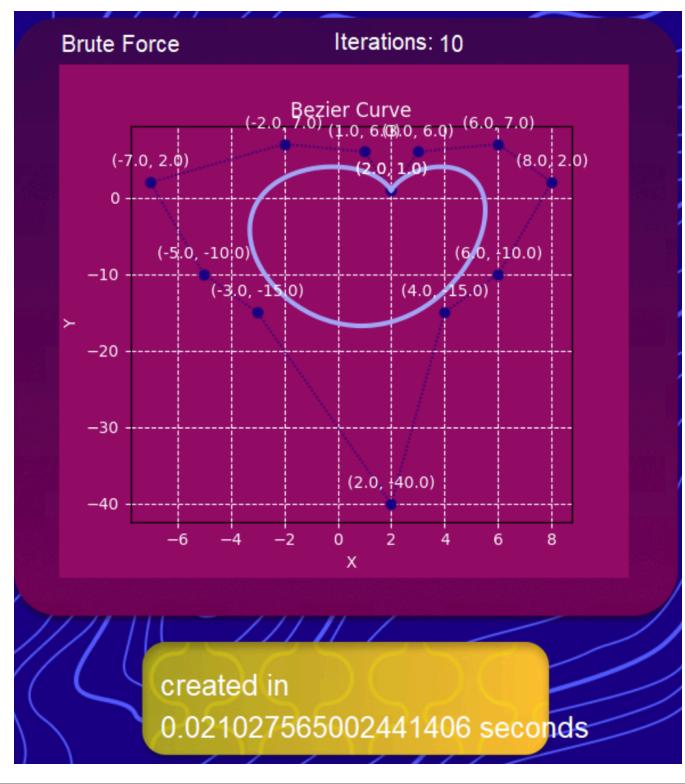
Point 1:	0	1	Delete
Point 2:	3	6	Delete
Point 3:	6	7	Delete
Point 4:	8	2	Delete
Point 5:	6	-10	Delete
Point 6:	3	-5	Delete
Point 7:	5	0	Delete



### Set iteration

Add Points

Point 0:	4	-15	Delete
Point 7:	2	-40	Delete
Point 8:	-3	-15	Delete
Point 9:	-5	-10	Delete
Point 10:	-7	2	Delete
Point 11:	-2	7	Delete
Point 12:	1	6	Delete
Point 13:	2	1	Delete



## **BAB VI**

### **PROGRAM BONUS**

#### **6.1 Membuat kurva dengan N titik kontrol**

Pada kedua program, baik divide and conquer maupun brute force, keduanya dapat menerima input jumlah titik kontrol yang lebih dari 3. Namun, implementasi program pada pendekatan brute force tampak lebih mudah saat jumlah titik kontrol melebihi 3. Ini disebabkan oleh sifat algoritma brute force yang sederhana dan langsung, di mana penggunaan semua kemungkinan kombinasi titik kontrol dapat dilakukan dengan relatif mudah.

Di sisi lain, algoritma divide and conquer menemui beberapa kendala ketika jumlah titik kontrol melebihi 3. Meskipun algoritma ini efisien dalam menangani jumlah titik kontrol yang besar, namun pendekatan divide and conquer membutuhkan langkah-langkah tambahan untuk mengatasi masalah ketika titik kontrol melebihi 3. Kompleksitas tambahan ini dapat melibatkan proses pemecahan submasalah dan penggabungan solusi yang memerlukan perhitungan lebih lanjut, yang mungkin memperkenalkan potensi kesalahan atau kompleksitas tambahan dalam implementasi.

Oleh karena itu, ketika menangani kurva dengan lebih dari 3 titik kontrol, algoritma divide and conquer dapat menimbulkan beberapa tantangan yang tidak ditemui pada pendekatan brute force. Ini mungkin termasuk penyesuaian tambahan dalam kode untuk menangani kasus khusus ketika jumlah titik kontrol melebihi batas tertentu, atau bahkan memerlukan strategi khusus untuk mengelola kompleksitas tambahan yang mungkin timbul. Meskipun algoritma divide and conquer biasanya lebih efisien secara komputasional, tantangan dalam mengatasi masalah dengan lebih dari 3 titik kontrol menunjukkan bahwa tidak ada satu pendekatan pun yang sempurna dalam semua situasi, dan pilihan antara kedua pendekatan tersebut harus mempertimbangkan kebutuhan spesifik dari permasalahan yang dihadapi.

#### **6.2 Visualisasi kurva pada GUI tkinter**

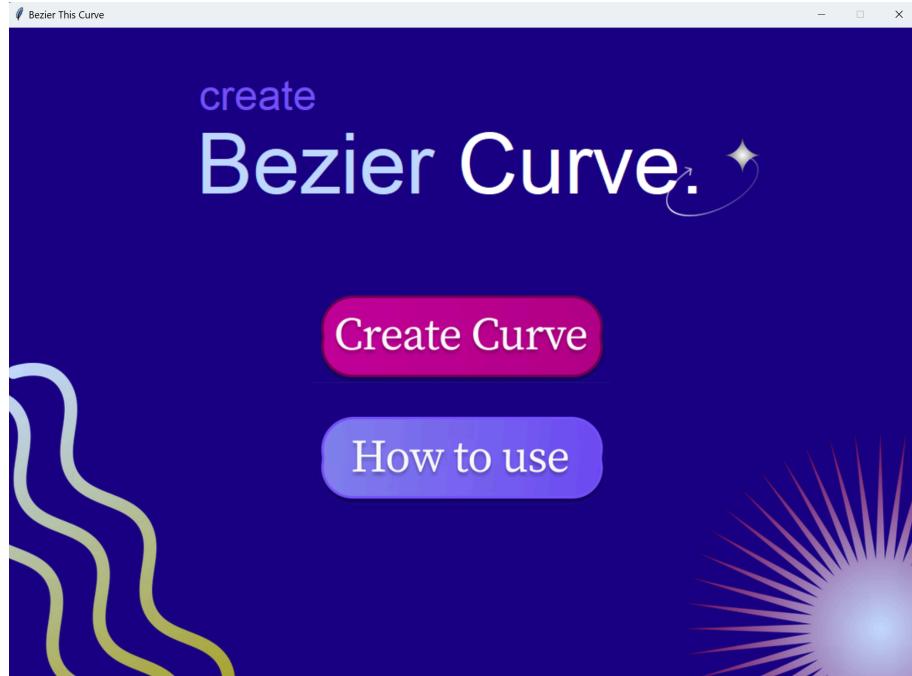
Pada program ini, GUI yang digunakan dibangun menggunakan modul Tkinter pada bahasa Python. Struktur GUI terdiri dari tiga frame atau halaman, yaitu homepage, how to use page, dan create curve page. Pada halaman create curve, pengguna diberikan opsi untuk menampilkan animasi yang memperlihatkan langkah-langkah pembentukan kurva. Opsi ini

memungkinkan pengguna untuk memvisualisasikan proses pembentukan kurva dengan lebih jelas.

Khusus untuk algoritma divide and conquer, animasi dijalankan dengan menampilkan hasil dari setiap iterasi mulai dari iterasi 0 hingga iterasi yang diinginkan. Dengan menggunakan fitur animasi, pengguna dapat melihat secara langsung bagaimana kurva terbentuk dari titik-titik kontrol yang diberikan. Animasi ini memberikan pemahaman yang lebih baik tentang cara kerja algoritma divide and conquer dalam memproses dan menyusun titik-titik kontrol menjadi kurva yang tepat.

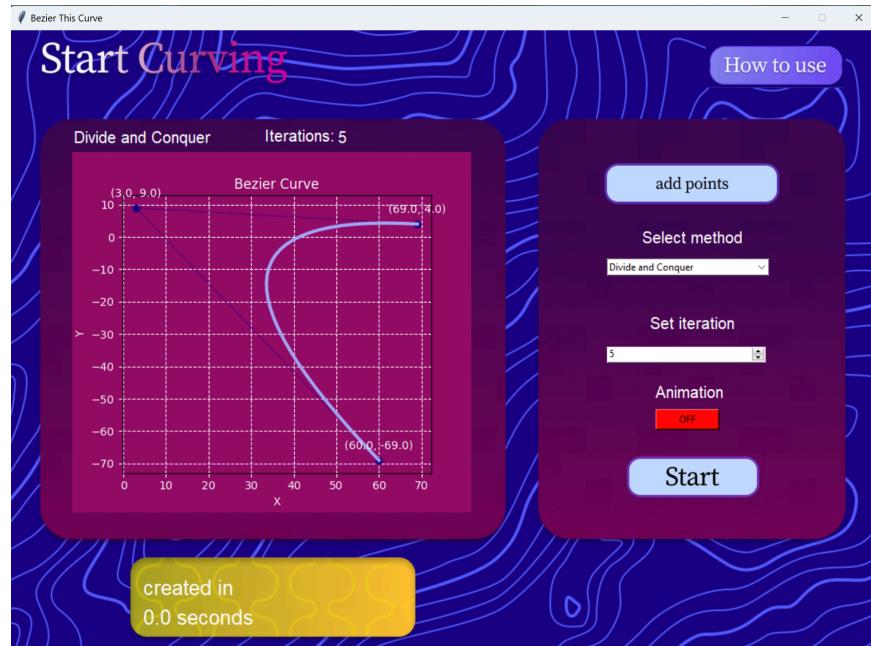
Sementara untuk algoritma brute force, prosesnya dimulai dengan menjalankan fungsinya terlebih dahulu. Setelah itu, titik-titik kontrol secara berurutan digunakan untuk membentuk kurva dengan pergerakan dari titik awal ke titik akhir. Meskipun tidak menggunakan animasi secara khusus seperti pada algoritma divide and conquer, namun hasil pergerakan titik-titik ini tetap dapat memberikan gambaran visual tentang bagaimana kurva terbentuk secara bertahap.

Untuk mengimplementasikan animasi, program menggunakan fitur bawaan dari library Matplotlib yang disebut FuncAnimation. Dengan memanfaatkan fitur ini, pengguna dapat melihat perubahan kurva secara real-time selama proses pembentukan. Ini memberikan pengalaman interaktif yang lebih menarik dan memudahkan pengguna dalam memahami konsep dan mekanisme yang terlibat dalam pembentukan kurva menggunakan kedua pendekatan algoritma tersebut.

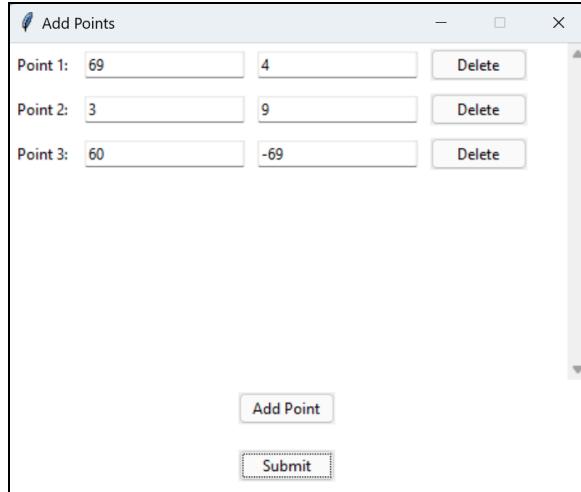


**Gambar 5.** *Homepage GUI*

Pada halaman utama (homepage), terdapat dua tombol navigasi. Pertama, tombol yang mengarah ke laman create curve, tempat pengguna dapat melakukan pengaturan kurva. Kedua, tombol untuk menuju ke how to use page, yang memberikan petunjuk penggunaan program.



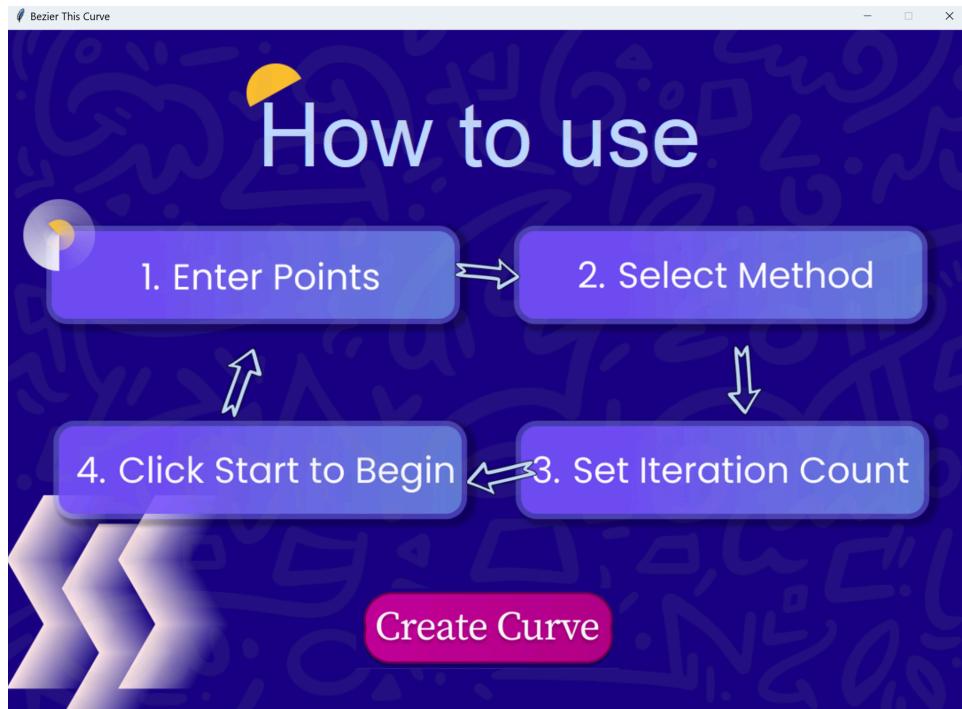
**Gambar 6.** *Create Curve Page*



**Gambar 7.** Window Penambahan Titik Control

Halaman create curve memungkinkan pengguna untuk mengatur titik kontrol, memilih metode (divide and conquer atau brute force), menentukan jumlah iterasi, serta menampilkan visualisasi grafik menggunakan Matplotlib. Di sini, terdapat elemen-elemen berikut:

- Input Titik Kontrol: Pengguna dapat mengisi titik kontrol yang diperlukan untuk pembentukan kurva.
- Pilihan Metode: Pengguna diberikan opsi untuk memilih metode algoritma yang akan digunakan, yaitu divide and conquer atau brute force.
- Set Iterasi: Pengguna dapat menentukan jumlah iterasi yang diinginkan.
- Tombol Menuju How to Use Page: Tombol ini memungkinkan pengguna untuk kembali ke halaman yang memberikan petunjuk penggunaan program.
- Tombol Animasi: Tombol untuk menghidupkan atau mematikan animasi yang menampilkan proses pembentukan kurva.
- Tombol Menjalankan Program: Tombol untuk menjalankan program dengan konfigurasi yang telah ditentukan.
- Visualisasi Grafik: Tempat untuk menampilkan visualisasi grafik menggunakan Matplotlib, yang akan memperlihatkan kurva yang dihasilkan sesuai dengan konfigurasi yang telah dipilih.



**Gambar 8.** How to use Page

Pada halaman how to use, pengguna diberikan langkah-langkah untuk menjalankan program. Di sini, terdapat elemen-elemen berikut:

- Langkah-langkah: Penjelasan tentang cara menggunakan program, termasuk cara mengisi titik kontrol, memilih metode, menentukan jumlah iterasi, dan menjalankan program.
- Tombol Menuju Create Curve Page: Tombol yang memungkinkan pengguna untuk kembali ke halaman create curve dan melakukan pengaturan lebih lanjut sebelum menjalankan program.

## **BAB VII**

### **KESIMPULAN**

Kesimpulan dari perbandingan penyelesaian kurva bezier dengan algoritma divide and conquer dan algoritma brute force yang telah dilakukan bentuk kurva yang didapatkan sama namun proses pembentukannya berbeda, divide and conquer dilakukan banyak pembagian pembentukan, sedangkan brute force dilakukan terurut secara linier. Dalam penelitian kami terdapat perbedaan bentuk kurva antara algoritma brute force dan divide and conquer jika memasukan input titik kontrol berjumlah genap. Dalam percobaan perbandingan kami, dari segi waktu jika dilakukan iterasi dibawah 19 didapatkan algoritma divide and conquer lebih cepat sedikit sedangkan di atas 19 iterasi didapatkan pembentukan kurva dengan algoritma brute force lebih cepat dibanding algoritma divide and conquer. Jadi kami simpulkan dalam pembentukan kurva bezier lebih efektif menggunakan algoritma brute force jika dilakukan banyak iterasi.

## **DAFTAR PUSTAKA**

"Introduction to Divide and Conquer Algorithm - GeeksforGeeks." GeeksforGeeks, <https://www.geeksforgeeks.org/introduction-to-divide-and-conquer-algorithm-data-structure-and-algorithm-tutorials/?ref=lbp>. Diakses pada tanggal 18 Maret 2024, pukul 19:00.

Bandara, Ravimal. "Midpoint Algorithm: Divide and Conquer Method for Drawing Circle - CodeProject." CodeProject, 9 Juli 2011, <https://www.codeproject.com/articles/223159/midpoint-algorithm-divide-and-conquer-method-for-d>. Diakses pada tanggal 17 Maret 2024, pukul 8:00.

"Divide and Conquer Algorithm in Data Structure - Programiz." Programiz, <https://www.programiz.com/dsa/divide-and-conquer>. Diakses pada tanggal 18 Maret 2024, pukul 20:00.

## LAMPIRAN

1. Github Link: [https://github.com/thoriqsaputra/Tucil2\\_13522141\\_13522159.git](https://github.com/thoriqsaputra/Tucil2_13522141_13522159.git)
- 2.

Poin	Ya	Tidak
Program berhasil dijalankan.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Program dapat melakukan visualisasi kurva Bézier.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Solusi yang diberikan program optimal.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<b>[Bonus]</b> Program dapat membuat kurva untuk $n$ titik kontrol.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<b>[Bonus]</b> Program dapat melakukan visualisasi proses pembuatan kurva.	<input checked="" type="checkbox"/>	<input type="checkbox"/>