

TUGAS KECIL 2
STRATEGI ALGORITMA

Membangun Kurva Bézier dengan Algoritma Titik Tengah berbasis Divide and Conquer



Disusun Oleh:
Benjamin Sihombing (13522054)
Evelyn Yosiana (13522083)

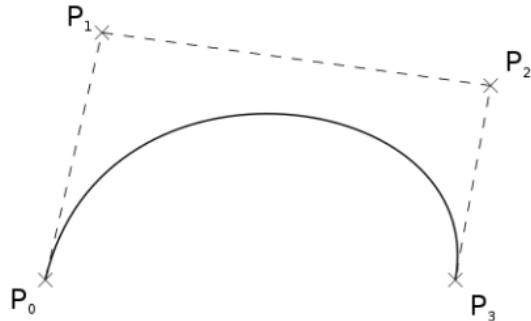
Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

DAFTAR ISI

DAFTAR ISI.....	2
BAB 1.....	3
BAB 2.....	6
BAB 3.....	10
BAB 4.....	12
BAB 5.....	24
BAB 6.....	25
BAB 7.....	32
DAFTAR PUSTAKA.....	33
LAMPIRAN.....	34
Github Repository.....	34
Checklist Keberhasilan Program.....	34

BAB 1

Deskripsi Masalah



Gambar 1.1 Kurva Bézier Kubik

(Sumber: https://id.wikipedia.org/wiki/Kurva_B%C3%A9zier)

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 realistik, 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 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.

Mengulas lebih jauh mengenai bagaimana sebuah kurva Bézier bisa terbentuk, misalkan diberikan dua buah titik P_0 dan P_1 yang menjadi titik kontrol, maka kurva Bézier yang terbentuk adalah sebuah garis lurus antara dua titik. Kurva ini disebut dengan kurva Bézier linier. Misalkan terdapat sebuah titik Q_0 yang berada pada garis yang dibentuk oleh P_0 dan P_1 , maka posisinya dapat dinyatakan dengan persamaan parametrik berikut.

$$Q_0 = B(t) = (1 - t)P_0 + tP_1, \quad t \in [0, 1]$$

dengan t dalam fungsi kurva Bézier linier menggambarkan seberapa jauh $B(t)$ dari P_0 ke P_1 . Misalnya ketika $t = 0.25$, maka $B(t)$ adalah seperempat jalan dari titik P_0 ke P_1 . sehingga seluruh rentang variasi nilai t dari 0 hingga 1 akan membuat persamaan $B(t)$ membentuk sebuah garis lurus dari P_0 ke P_1 .

Misalkan selain dua titik sebelumnya ditambahkan sebuah titik baru, sebut saja P_2 , dengan P_0 dan P_2 sebagai titik kontrol awal dan akhir, dan P_1 menjadi titik kontrol antara. Dengan menyatakan titik Q_1 terletak diantara garis yang menghubungkan P_1 dan P_2 , dan membentuk kurva Bézier linier yang berbeda dengan kurva letak Q_0 berada, maka dapat dinyatakan sebuah titik baru, R_0 yang berada diantara garis yang menghubungkan Q_0 dan Q_1 yang bergerak membentuk kurva Bézier kuadratik terhadap titik P_0 dan P_2 . Berikut adalah uraian persamaannya.

$$Q_0 = B(t) = (1 - t)P_0 + tP_1, \quad t \in [0, 1]$$

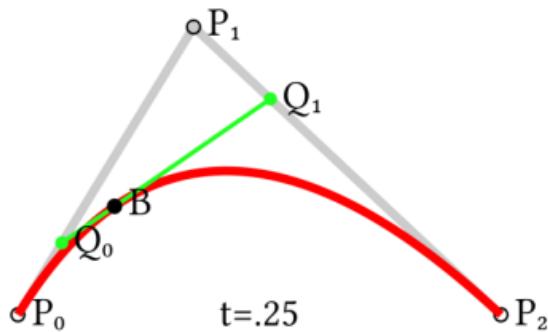
$$Q_1 = B(t) = (1 - t)P_1 + tP_2, \quad t \in [0, 1]$$

$$R_0 = B(t) = (1 - t)Q_0 + tQ_1, \quad t \in [0, 1]$$

dengan melakukan substitusi nilai Q_0 dan Q_1 , maka diperoleh persamaan sebagai berikut.

$$R_0 = B(t) = (1 - t)^2 P_0 + (1 - t)t P_1 + t^2 P_2, \quad t \in [0, 1]$$

Berikut adalah ilustrasi dari kasus diatas.



Gambar 1.2 Pembentukan Kurva Bézier Kuadratik

(Sumber: <https://simonhalliday.com/2017/02/15/quadratic-bezier-curve-demo/>)

Proses ini dapat juga diaplikasikan untuk jumlah titik yang lebih dari tiga, misalnya empat titik akan menghasilkan kurva Bézier kubik, lima titik akan menghasilkan kurva Bézier kuartik, dan seterusnya. Berikut adalah persamaan kurva Bézier kubik dan kuartik dengan menggunakan prosedur yang sama dengan yang sebelumnya.

$$S_0 = B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3, \quad t \in [0, 1]$$

$$T_0 = B(t) = (1-t)^4 P_0 + 4(1-t)^3 t P_1 + 6(1-t)^2 t^2 P_2 + 4(1-t)t^3 P_3 + t^4 P_4, \quad t \in [0, 1]$$

Tentu saja persamaan yang terbentuk sangat panjang dan akan semakin rumit seiring bertambahnya titik. Oleh sebab itu, dalam rangka melakukan efisiensi pembuatan kurva Bézier yang sangat berguna ini, maka Anda diminta untuk mengimplementasikan pembuatan kurva Bézier dengan algoritma titik tengah berbasis divide and conquer.

BAB 2

Analisis dan Implementasi

2.1. Analisis dan Implementasi dalam Algoritma Brute Force sebagai Algoritma Pembanding

2.1.1. Implementasi Algoritma *Brute Force*

Pada pembentukan kurva Bézier dengan algoritma Brute Force, koordinat y dapat dicari dengan memasukkan nilai x ke rumus kurva tersebut. Berikut beberapa rumus kurva Bézier.

Rumus kurva Bézier untuk 3 titik:

$$R_0 = B(t) = (1-t)^2 P_0 + (1-t)tP_1 + t^2P_2, \quad t \in [0, 1]$$

Rumus kurva Bézier untuk 4 titik:

$$S_0 = B(t) = (1-t)^3 P_0 + 3(1-t)^2tP_1 + 3(1-t)t^2P_2 + t^3P_3, \quad t \in [0, 1]$$

Rumus kurva Bézier untuk 5 titik:

$$T_0 = B(t) = (1-t)^4 P_0 + 4(1-t)^3tP_1 + 6(1-t)^2t^2P_2 + 4(1-t)t^3P_3 + t^4P_4, \quad t \in [0, 1]$$

Secara umum, rumus pembentukan kurva Bézier adalah sebagai berikut:

$$\mathbf{B}(t) = \sum_{i=0}^n \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i$$

Algoritma pembentukan kurva Bézier dengan metode brute force adalah sebagai berikut.

1. Buat list yang memuat nilai segitiga pascal dengan orde yang sama dengan banyak iterasi yang ingin dilakukan.
2. Buat list kosong untuk menampung hasil akhir.
3. Buat variabel t dengan nilai 0.
4. Lakukan iterasi sebanyak jumlah titik dengan langkah-langkah sebagai berikut.

- a. Buat variabel dengan nilai nol.
 - b. Buat sebuah variabel baru dengan nilai $(1 - t)$
 - c. Pangkatkan variabel pada langkah 4b dengan 1 dikurang n, dimana n merupakan iterasi yang sedang dijalankan.
 - d. Kalikan variabel pada langkah 4c dengan list pada langkah nomor 3 dengan indeks ke-n.
 - e. Kalikan variabel pada langkah 4d dengan titik ke-n.
 - f.Tambahkan nilai variabel pada langkah 4a dengan nilai variabel pada langkah 4e.
5. Setelah iterasi pada langkah 4 selesai dijalankan untuk satu nilai t, masukkan nilai variabel pada langkah 4f ke dalam list pada langkah ke-2.
 6. Tambahkan t dengan satu dibagi dua dipangkatkan banyak iterasi.
 7. Ulangi langkah nomor 4 sampai 6 hingga nilai t sama dengan satu.
 8. Lakukan langkah nomor 1 sampai 7 untuk nilai x dan nilai y.

2.1.2. Analisis Algoritma *Brute Force*

Pada fungsi `brute_force_bezier`, terdapat satu *while loop* yang akan melakukan pengulangan sebanyak 2^k dengan k adalah jumlah iterasi dan satu *for loop* yang akan melakukan pengulangan sebanyak n yaitu jumlah titik kontrol. Pada *for loop* terdapat 9 operasi aritmatik. Pada *while loop* terdapat 1 *for loop* sebelumnya, 1 operasi *append* dan 1 operasi aritmatik. Selain itu, fungsi juga menjalankan satu fungsi lain yaitu `pascals_triangle` untuk membuat segitiga pascal. Kompleksitas fungsi segitiga pascal adalah $g(n) = n + (n - 1) + (n - 2) + \dots + 1 = \frac{n(n+1)}{2}$. Maka total kompleksitas algoritma ini adalah

$$T(n, k) = \text{komp. pascal} + \text{operasi loop} + \text{operasi selain loop}$$

$$T(n, k) = \frac{n(n+1)}{2} + 2^k(n)9 + 2^k2 = \frac{n(n+1)}{2} + 2^k(9n + 2)$$

$$T(n, k) = O(2^k n)$$

2.2. Analisis dan Implementasi dalam Algoritma *Divide and Conquer*

2.2.1. Implementasi Algoritma *Divide and Conquer*

Pada pembuatan kurva Bézier menggunakan algoritma *divide and conquer* ini, kurva dibentuk dari titik-titik yang merupakan titik tengah dari suatu garis. Garis tersebut bisa terbentuk dari titik tengah dengan titik tengah lain, titik kontrol dengan titik kontrol, atau titik tengah dengan titik kontrol. Program ini sudah digeneralisasi order berapa pun. Berikut ini implementasi algoritma *divide and conquer*:

1. Program akan menerima input jumlah titik kontrol awal, jumlah iterasi, dan titik kontrol.
2. Setelah itu, program akan memastikan bahwa jumlah iterasi sudah tercapai atau tidak. Jika sudah mencapai jumlah iterasi, program akan berhenti dan mengeluarkan list berisi titik-titik yang membentuk kurva Bézier.
3. Jika program belum mencapai jumlah iterasi, list input akan dibagi (**divide**) menjadi beberapa list dasar sesuai dengan jumlah titik kontrolnya. Setiap list baru memiliki ukuran sebesar jumlah titik kontrol. Setiap elemen terakhir pada list baru pasti merupakan elemen pertama pada list baru selanjutnya. Misalnya, list [a,b,c,d,e,f,g] dengan titik kontrol 3 akan menjadi [a,b,c],[c,d,e], dan [e,f,g].
4. Setiap list baru akan mengalami proses iterasi dasar dan hasilnya akan digabungkan satu sama lain (**conquer**).

Berikut ini penjelasan proses iterasi dasar:

1. Proses akan menerima n sebagai pencacahan dan list. Nilai n awal adalah jumlah titik kontrol.
2. Jika n sama dengan 1, proses akan berhenti dan mengeluarkan hasil listnya.
3. Jika n tidak sama dengan 1, proses akan membentuk list baru yang berisi titik tengah antar titik-titik pada list awal.

- Selanjutnya, elemen pertama dan elemen akhir dari list awal akan digabungkan dengan pemanggilan proses ulang dengan input n-1 dan list baru (rekursif).

2.2.2. Analisis Algoritma *Divide and Conquer*

Pada algoritma ini akan dibentuk list berukuran 2^k . List tersebut akan diisi dengan sublist (**divide**). Setiap sublist akan diproses menggunakan iterasi dasar dengan kompleksitas $g(n) = \frac{n^2+n}{2} - 1$. Lalu, sublist-sublist tersebut akan digabungkan kembali menjadi satu list. Proses ini akan berulang sebanyak k kali. Setiap pengulangan proses, nilai k akan berkurang 1. Berikut ini adalah kompleksitas algoritma *divide and conquer* pada program ini:

$$T(n, k) = g(n), \text{ untuk } k = 1$$

$$T(n, k) = \text{komp. divide} + \text{komp. proses} + \text{komp. conquer}, \text{ untuk } k > 1$$

$$T(n, k) = 2^k nc + 2^k \left(\frac{n^2+n}{2} - 1 \right) + 2^k + T(n, k - 1), \text{ untuk } k > 1$$

Pada program ini nilai $c = 4$,

$$T(n, k) = 2^k 4n + 2^k \left(\frac{n^2+n}{2} - 1 \right) + 2^k + T(n, k - 1), \text{ untuk } k > 1$$

$$T(n, k) = 2^k \left(\frac{n^2+9n}{2} \right) + T(n, k - 1), \text{ untuk } k > 1$$

Maka,

$$T(n, k) = 1 \left(\frac{2^k - 1}{2^k} \right) \left(\frac{n^2+9n}{2} \right) = (2^k - 1) \left(\frac{n^2+9n}{2} \right) = O(2^k n^2)$$

BAB 3

Source Code

3.1. *Source Code Algoritma Brute Force*

Berikut ini algoritma *brute force* untuk pembentukan kurva Bézier

```
# Segitiga pascal buat konstantanya
def pascals_triangle(num_rows):
    triangle = []
    for i in range(num_rows+1):
        row = [1] * (i + 1)
        for j in range(1, i):
            row[j] = triangle[i - 1][j - 1] + triangle[i - 1][j]
        triangle.append(row)
    return triangle[len(triangle)-1]

# Brute force
def brute_force_bezier (num_of_point, list_of_point, itr):
    list_result = []
    idx = 0
    offset = 1 / (2**itr)
    constant = pascals_triangle(num_of_point-1)
    while idx <= 1:
        value = 0
        for i in range (len(constant)):
            value += constant[i] * ((1-idx)**(num_of_point-i-1)) * list_of_point[i] * (idx**i)
        list_result.append(value)
        idx += offset
    return list_result

# Brute force 3 titik
def brute_force_3titik (list_of_point, itr):
    list_result = []
    idx = 0
    offset = 1 / (2**itr)
    constant = [1, 2, 1]
    while idx <= 1:
        value = 0
        for i in range (3):
            value += constant[i] * ((1-idx)**(2-i)) * list_of_point[i] * (idx**i)
        list_result.append(value)
        idx += offset
    return list_result
```

Gambar 3.1 *Source Code Algoritma Brute Force*

3.2. Source Code Algoritma Divide and Conquer

Berikut ini algoritma *divide and conquer* untuk pembentukan kurva Bézier:

```
def midPoint(p1,p2):
    return [(p1[0]+p2[0])/2,(p1[1]+p2[1])/2]

def wide(i,list,f):
    temp = list
    temp.insert(0,i)
    temp.append(f)
    return temp

def base_iterate(n, list, listProses):
    if(n == 1):
        listProses.append([list[len(list)//2]])
        return list
    else:
        temp = []
        listProses.append(list)
        for i in range(n-1):
            temp.append(midPoint(list[i],list[i+1]))
        return wide(list[0],base_iterate(n-1,temp,listProses),list[-1])

def connect(list1,list2):
    if(len(list1) == 0):
        return list2
    else:
        list2.pop(0)
        return list1 + (list2)
```

Gambar 3.2 Source Code Algoritma Fungsi Tambahan untuk *Divide & Conquer*

```
def general_iterate(n, iterate, count, list, listProses):
    if(iterate == count):
        return list
    else:
        result = []
        for i in range(2**count):
            temp = []
            for j in range(n):
                temp.append(list[(n-1)*i+j])
            temp = base_iterate(n,temp,listProses)
            result = connect(result,temp)
        return general_iterate(n,iterate,count+1,result,listProses)
```

Gambar 3.3 Source Code Algoritma *Divide & Conquer*

```
def take_result_point(list, n):
    result = []
    for i in range(len(list)):
        if(i%(n-1) == 0):
            result.append(list[i])
    return result
```

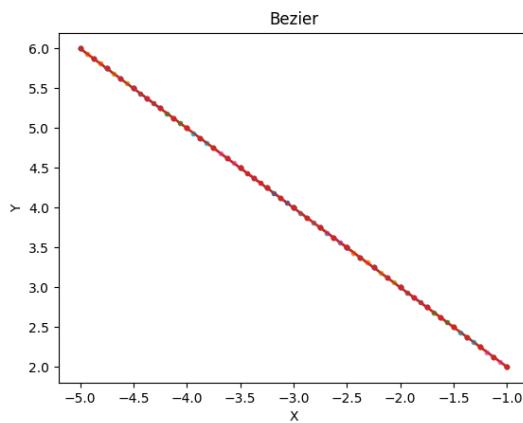
Gambar 3.4 Source Code Algoritma untuk Mengembalikan Titik Kurva Bézier

BAB 4

Testing

Test Case	Divide and Conquer	Brute Force
TC 1 3 titik 5 iterasi	<pre>Selamat datang di generator kurva berzier! Banyak titik yang dapat dipilih: 1. 3 titik 2. n titik Silakan masukkan pilihan: 1 Silakan masukkan banyaknya iterasi: 5 Silahkan masukkan koodinat x1 dan y1 dipisahkan spasi (dalam float) tanpa spasi di akhir: -1 2 Silahkan masukkan koodinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: -3 4 Silahkan masukkan koodinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir: -5 6 Metode penyelesaian: 1. Divide and Conquer 2. Brute Force Silakan masukkan pilihan: 2</pre> <pre>Hasil dengan metode divide and conquer: [-1.0, 2.0] [-1.125, 2.125] [-1.25, 2.25] [-1.375, 2.375] [-1.5, 2.5] [-1.625, 2.625] [-1.75, 2.75] [-1.875, 2.875] [-2.0, 3.0] [-2.125, 3.125] [-2.25, 3.25] [-2.375, 3.375] [-2.5, 3.5] [-2.625, 3.625] [-2.75, 3.75] [-2.875, 3.875] [-3.0, 4.0] [-3.125, 4.125] [-3.25, 4.25] [-3.375, 4.375] [-3.5, 4.5] [-3.625, 4.625] [-3.75, 4.75] [-3.875, 4.875] [-4.0, 5.0] [-4.125, 5.125] [-4.25, 5.25] [-4.375, 5.375] [-4.5, 5.5] [-4.625, 5.625] [-4.75, 5.75] [-4.875, 5.875] [-5.0, 6.0]</pre> <p>Silakan masukkan nama file gambar: test1 Gambar kurva berhasil disimpan Runtime: 32.00125694274902 ms</p>	

```
Hasil dengan metode brute force:  
[-1.0, 2.0]  
[-1.125, 2.125]  
[-1.25, 2.25]  
[-1.375, 2.375]  
[-1.5, 2.5]  
[-1.625, 2.625]  
[-1.75, 2.75]  
[-1.875, 2.875]  
[-2.0, 3.0]  
[-2.125, 3.125]  
[-2.25, 3.25]  
[-2.375, 3.375]  
[-2.5, 3.5]  
[-2.625, 3.625]  
[-2.75, 3.75]  
[-2.875, 3.875]  
[-3.0, 4.0]  
[-3.125, 4.125]  
[-3.25, 4.25]  
[-3.375, 4.375]  
[-3.5, 4.5]  
[-3.625, 4.625]  
[-3.75, 4.75]  
[-3.875, 4.875]  
[-4.0, 5.0]  
[-4.125, 5.125]  
[-4.25, 5.25]  
[-4.375, 5.375]  
[-4.5, 5.5]  
[-4.625, 5.625]  
[-4.75, 5.75]  
[-4.875, 5.875]  
[-5.0, 6.0]  
Runtime: 25.16961097717285 ms
```



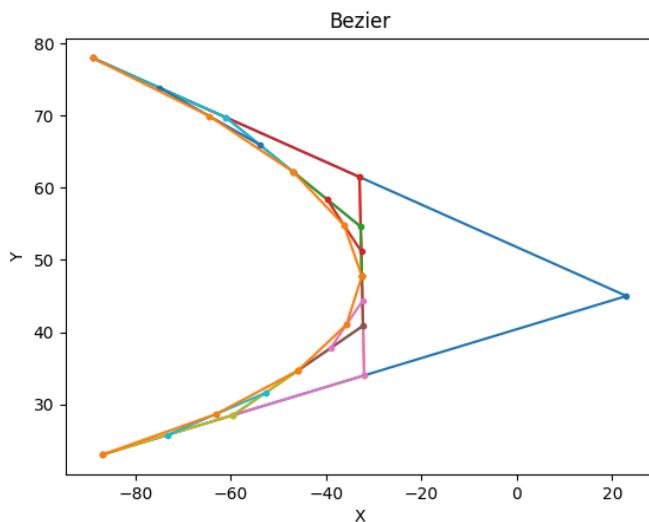
Runtime: 32.00 ms

Runtime: 25.16 ms

TC 2
3 titik
3 iterasi

```
Selamat datang di generator kurva berzier!
Banyak titik yang dapat dipilih: 1. 3 titik 2. n titik
Silakan masukkan pilihan: 1
Silakan masukkan banyaknya iterasi: 3
Silahkan masukkan koodinat x1 dan y1 dipisahkan spasi (dalam float) tanpa spasi di akhir: -89 78
Silahkan masukkan koodinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: 23 45
Silahkan masukkan koodinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir: -87 23
Metode penyelesaian: 1. Divide and Conquer 2. Brute Force
Silakan masukkan pilihan: 1
Hasil dengan metode divide and conquer:
[-89.0, 78.0]
[-64.46875, 69.921875]
[-46.875, 62.1875]
[-36.21875, 54.796875]
[-32.5, 47.75]
[-35.71875, 41.046875]
[-45.875, 34.6875]
[-62.96875, 28.671875]
[-87.0, 23.0]
Silakan masukkan nama file gambar: text2new
Gambar kurva berhasil disimpan
Runtime: 0.7684230804443359 ms
```

```
Selamat datang di generator kurva berzier!
Banyak titik yang dapat dipilih: 1. 3 titik 2. n titik
Silakan masukkan pilihan: 1
Silakan masukkan banyaknya iterasi: 3
Silahkan masukkan koodinat x1 dan y1 dipisahkan spasi (dalam float) tanpa spasi di akhir: -89 78
Silahkan masukkan koodinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: 23 45
Silahkan masukkan koodinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir: -87 23
Metode penyelesaian: 1. Divide and Conquer 2. Brute Force
Silakan masukkan pilihan: 2
Hasil dengan metode brute force:
[-89.0, 78.0]
[-64.46875, 69.921875]
[-46.875, 62.1875]
[-36.21875, 54.796875]
[-32.5, 47.75]
[-35.71875, 41.046875]
[-45.875, 34.6875]
[-62.96875, 28.671875]
[-87.0, 23.0]
Runtime: 8.00943374633789 ms
```



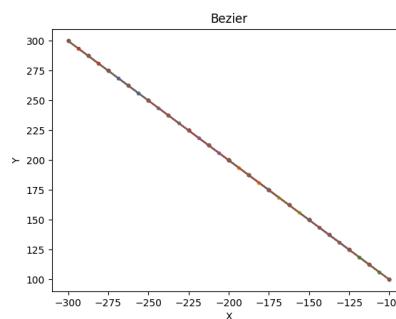
Runtime: 0.77 ms

Runtime: 8.00 ms

TC 3
3 titik
4 iterasi

```
Selamat datang di generator kurva berzier!
Banyak titik yang dapat dipilih: 1. 3 titik 2. n titik
Silakan masukkan pilihan: 1
Silakan masukkan banyaknya iterasi: 4
Silahkan masukkan koodinat x1 dan y1 dipisahkan spasi (dalam float) tanpa spasi di akhir: -100 100
Silahkan masukkan koodinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: -200 200
Silahkan masukkan koodinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir: -300 300
Metode penyelesaian: 1. Divide and Conquer 2. Brute Force
Silakan masukkan pilihan: 1
Hasil dengan metode divide and conquer:
[-100.0, 100.0]
[-112.5, 112.5]
[-125.0, 125.0]
[-137.5, 137.5]
[-150.0, 150.0]
[-162.5, 162.5]
[-175.0, 175.0]
[-187.5, 187.5]
[-200.0, 200.0]
[-212.5, 212.5]
[-225.0, 225.0]
[-237.5, 237.5]
[-250.0, 250.0]
[-262.5, 262.5]
[-275.0, 275.0]
[-287.5, 287.5]
[-300.0, 300.0]
Silakan masukkan nama file gambar: test3
Gambar kurva berhasil disimpan
Runtime: 3.008127212524414 ms
```

```
Selamat datang di generator kurva berzier!
Banyak titik yang dapat dipilih: 1. 3 titik 2. n titik
Silakan masukkan pilihan: 1
Silakan masukkan banyaknya iterasi: 4
Silahkan masukkan koodinat x1 dan y1 dipisahkan spasi (dalam float) tanpa spasi di akhir: -100 100
Silahkan masukkan koodinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: -200 200
Silahkan masukkan koodinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir: -300 300
Metode penyelesaian: 1. Divide and Conquer 2. Brute Force
Silakan masukkan pilihan: 2
Hasil dengan metode brute force:
[-100.0, 100.0]
[-112.5, 112.5]
[-125.0, 125.0]
[-137.5, 137.5]
[-150.0, 150.0]
[-162.5, 162.5]
[-175.0, 175.0]
[-187.5, 187.5]
[-200.0, 200.0]
[-212.5, 212.5]
[-225.0, 225.0]
[-237.5, 237.5]
[-250.0, 250.0]
[-262.5, 262.5]
[-275.0, 275.0]
[-287.5, 287.5]
[-300.0, 300.0]
Runtime: 1.9867420196533203 ms
```



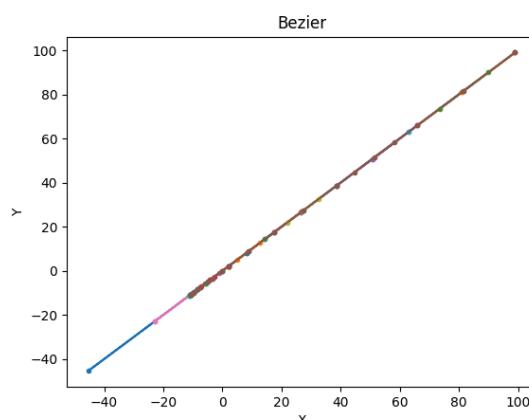
Runtime: 3.00 ms

Runtime: 1.99 ms

TC 4
3 titik
4 iterasi

```
Selamat datang di generator kurva berzier!
Banyak titik yang dapat dipilih: 1. 3 titik 2. n titik
Silakan masukkan pilihan: 1
Silakan masukkan banyaknya iterasi: 4
Silahkan masukkan koordinat x1 dan y1 dipisahkan spasi (dalam float) tanpa spasi di akhir: 98.98 98.98
Silahkan masukkan koordinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: -45.45 -45.45
Silahkan masukkan koordinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir: 0 0
Metode penyelesaian: 1. Divide and Conquer 2. Brute Force
Silakan masukkan pilihan: 1
Hasil dengan metode divide and conquer:
[98.98, 98.98]
[81.66796875, 81.66796875]
[65.839375, 65.839375]
[51.49421875, 51.49421875]
[38.6325, 38.6325]
[27.25421875, 27.25421875]
[17.359375, 17.359375]
[8.94796875000001, 8.94796875000001]
[2.01999999999996, 2.01999999999996]
[-3.42453125000001, -3.42453125000001]
[-7.38562500000001, -7.38562500000001]
[-9.86328125, -9.86328125]
[-10.85750000000002, -10.85750000000002]
[-10.36828125, -10.36828125]
[-8.395625, -8.395625]
[-4.93953125, -4.93953125]
[0.0, 0.0]
Silakan masukkan nama file gambar: test4
Gambar kurva berhasil disimpan
Runtime: 10.999917984008789 ms
```

```
Selamat datang di generator kurva berzier!
Banyak titik yang dapat dipilih: 1. 3 titik 2. n titik
Silakan masukkan pilihan: 1
Silakan masukkan banyaknya iterasi: 4
Silahkan masukkan koordinat x1 dan y1 dipisahkan spasi (dalam float) tanpa spasi di akhir: 98.98 98.98
Silahkan masukkan koordinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: -45.45 -45.45
Silahkan masukkan koordinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir: 0 0
Metode penyelesaian: 1. Divide and Conquer 2. Brute Force
Silakan masukkan pilihan: 2
Hasil dengan metode brute force:
[98.98, 98.98]
[81.66796875, 81.66796875]
[65.839375, 65.839375]
[51.49421874999995, 51.49421874999995]
[38.6325, 38.6325]
[27.25421875, 27.25421875]
[17.359375, 17.359375]
[8.94796875000001, 8.94796875000001]
[2.01999999999996, 2.01999999999996]
[-3.42453125000001, -3.42453125000001]
[-7.38562500000003, -7.38562500000003]
[-9.86328125, -9.86328125]
[-10.85750000000002, -10.85750000000002]
[-10.36828125, -10.36828125]
[-8.395625, -8.395625]
[-4.93953125, -4.93953125]
[0.0, 0.0]
Runtime: 1.9979476928710938 ms
```



Runtime: 10.99 ms

Runtime: 1.99 ms

TC 5
3 titik
8 iterasi

Selamat datang di generator kurva berzier!
Banyak titik yang dapat dipilih: 1. 3 titik 2. n titik
Silakan masukkan pilihan: 1
Silakan masukkan banyaknya iterasi: 8
Silahkan masukkan koodinat x1 dan y1 dipisahkan spasi (dalam float) tanpa spasi di akhir: 1 2
Silahkan masukkan koodinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: 3 4
Silahkan masukkan koodinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir: 5 6
Metode penyelesaian: 1. Divide and Conquer 2. Brute Force
Silakan masukkan pilihan: 1
Hasil dengan metode divide and conquer:
[1.0, 2.0]
[1.015625, 2.015625]
[1.03125, 2.03125]
[1.046875, 2.046875]
[1.0625, 2.0625]
[1.078125, 2.078125]
[1.09375, 2.09375]
[1.109375, 2.109375]
[1.125, 2.125]
[1.140625, 2.140625]
[1.15625, 2.15625]
[1.171875, 2.171875]
[1.1875, 2.1875]
[1.203125, 2.203125]
[1.21875, 2.21875]
[1.234375, 2.234375]
[1.25, 2.25]
[1.265625, 2.265625]
[1.28125, 2.28125]
[1.296875, 2.296875]

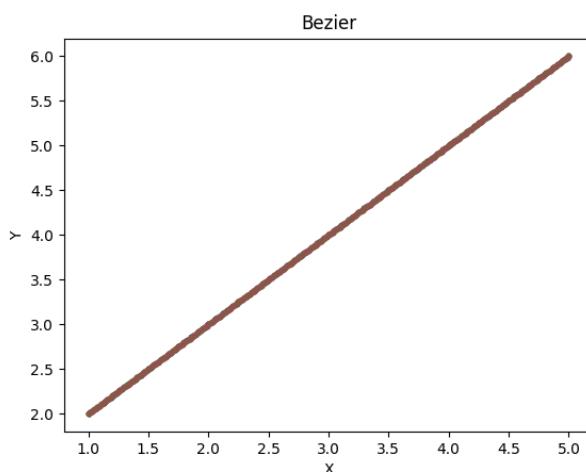
[4.6875, 5.6875]
[4.703125, 5.703125]
[4.71875, 5.71875]
[4.734375, 5.734375]
[4.75, 5.75]
[4.765625, 5.765625]
[4.78125, 5.78125]
[4.796875, 5.796875]
[4.8125, 5.8125]
[4.828125, 5.828125]
[4.84375, 5.84375]
[4.859375, 5.859375]
[4.875, 5.875]
[4.890625, 5.890625]
[4.90625, 5.90625]
[4.921875, 5.921875]
[4.9375, 5.9375]
[4.953125, 5.953125]
[4.96875, 5.96875]
[4.984375, 5.984375]
[5.0, 6.0]
Silakan masukkan nama file gambar: test5
Gambar kurva berhasil disimpan
Runtime: 29.992103576660156 ms

```

Selamat datang di generator kurva berzier!
Banyak titik yang dapat dipilih: 1. 3 titik 2. n titik
Silakan masukkan pilihan: 1
Silakan masukkan banyaknya iterasi: 8
Silahkan masukkan koodinat x1 dan y1 dipisahkan spasi (dalam float) tanpa spasi di akhir: 1 2
Silahkan masukkan koodinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: 3 4
Silahkan masukkan koodinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir: 5 6
Metode penyelesaian: 1. Divide and Conquer 2. Brute Force
Silakan masukkan pilihan: 2
Hasil dengan metode brute force:
[1.0, 2.0]
[1.015625, 2.015625]
[1.03125, 2.03125]
[1.046875, 2.046875]
[1.0625, 2.0625]
[1.078125, 2.078125]
[1.09375, 2.09375]
[1.109375, 2.109375]
[1.125, 2.125]
[1.140625, 2.140625]
[1.15625, 2.15625]
[1.171875, 2.171875]
[1.1875, 2.1875]
[1.203125, 2.203125]
[1.21875, 2.21875]
[1.234375, 2.234375]
[1.25, 2.25]
[1.265625, 2.265625]

[4.734375, 5.734375]
[4.75, 5.75]
[4.765625, 5.765625]
[4.78125, 5.78125]
[4.796875, 5.796875]
[4.8125, 5.8125]
[4.828125, 5.828125]
[4.84375, 5.84375]
[4.859375, 5.859375]
[4.875, 5.875]
[4.890625, 5.890625]
[4.90625, 5.90625]
[4.921875, 5.921875]
[4.9375, 5.9375]
[4.953125, 5.953125]
[4.96875, 5.96875]
[4.984375, 5.984375]
[5.0, 6.0]
Runtime: 30.971765518188477 ms

```

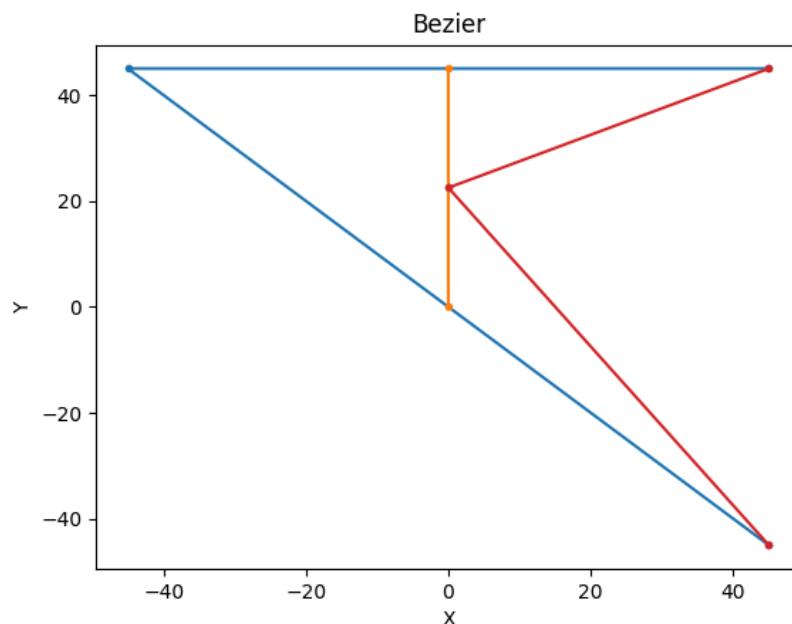


Runtime: 29.99 ms

Runtime: 30.97 ms

TC 6
3 titik
1 iterasi

```
Selamat datang di generator kurva berzier!
Banyak titik yang dapat dipilih: 1. 3 titik 2. n titik
Silakan masukkan pilihan: 1
Silakan masukkan banyaknya iterasi: 1
Silahkan masukkan kordinat x1 dan y1 dipisahkan spasi (dalam float) tanpa spasi di akhir: 45 -45
Silahkan masukkan kordinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: -45 45
Silahkan masukkan kordinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir: 45 45
Metode penyelesaian: 1. Divide and Conquer 2. Brute Force
Silakan masukkan pilihan: 1
Hasil dengan metode divide and conquer:
[45.0, -45.0]
[0.0, 22.5]
[45.0, 45.0]
Silakan masukkan nama file gambar: test6
Gambar kurva berhasil disimpan
Runtime: 1.0001659393310547 ms
PS C:\EVELYN\SEMESTER 4\STIMA\Tucil2Fix\Tucil2_13522054_13522083> python -u "c:\EVELYN\SEMESTER 4\STIMA\bezier.py"
Selamat datang di generator kurva berzier!
Banyak titik yang dapat dipilih: 1. 3 titik 2. n titik
Silakan masukkan pilihan: 1
Silakan masukkan banyaknya iterasi: 1
Silahkan masukkan kordinat x1 dan y1 dipisahkan spasi (dalam float) tanpa spasi di akhir: 45 -45
Silahkan masukkan kordinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: -45 45
Silahkan masukkan kordinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir: 45 45
Metode penyelesaian: 1. Divide and Conquer 2. Brute Force
Silakan masukkan pilihan: 2
Hasil dengan metode brute force:
[45.0, -45.0]
[0.0, 22.5]
[45.0, 45.0]
Runtime: 0.9889602661132812 ms
```



Runtime: 1.00 ms

Runtime: 0.9 ms

TC 7
Input
Handling
3 titik
1 iterasi

```
selamat datang di generator kurva berzier!
Banyak titik yang dapat dipilih: 1. 3 titik 2. n titik
Silakan masukkan pilihan: 3
Input invalid!
Silakan masukkan pilihan: 1
Silakan masukkan banyaknya iterasi: p
Input invalid!
Silakan masukkan banyaknya iterasi: satu
Input invalid!
Silakan masukkan banyaknya iterasi: 1
Silahkan masukkan kordinat x1 dan y1 dipisahkan spasi (dalam float) tanpa spasi di akhir: 45.45
Input invalid!
Silahkan masukkan kordinat x1 dan y1 dipisahkan spasi (dalam float) tanpa spasi di akhir: 45 45
Silahkan masukkan kordinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: p p
Input invalid!
Silahkan masukkan kordinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: 45 p
Input invalid!
Silahkan masukkan kordinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: -99 -99.99
Silahkan masukkan kordinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir:
Input invalid!
Silahkan masukkan kordinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir: 1;
Input invalid!
Silahkan masukkan kordinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir: 99 99
Metode penyelesaian: 1. Divide and Conquer 2. Brute Force
Silakan masukkan pilihan: 3
Input invalid!
Silakan masukkan pilihan: p
Input invalid!
Silakan masukkan pilihan: 1
Hasil dengan metode divide and conquer:
[45.0, 45.0]
[-13.5, -13.994999999999997]
[99.0, 99.0]
Silakan masukkan nama file gambar: test1
File sudah adal Silakan masukkan nama file gambar: test2
Gambar kurva berhasil disimpan
Runtime: 0.9951591491699219 ms
```

TC 8
GUI
5 titik
5 iterasi



Runtime: 28.3575057983 ms

Runtime: 11.246442794799 ms

TC 9 GUI 3 titik 5 iterasi	<p>Number of Points: 3 Number of Iterations: 5 List of Points: -1000, -800 -71, -655 122, 78 Solve Format input: x1,y1 x2,y2 x3,y3... Each point is split by a space Each x and y is split by a comma</p> <p>Bezier Graphic</p> <p>(122.0, 78.0)</p> <p>(-1000.0, -800.0)</p> <p>Points info</p> <pre>[[[-1000.0, -800.0], [-942.65625, -790.36328125], [-886.75, -779.578125], [-832.28125, -767.64453125], [-779.25, -754.5625], [-717.5, -708.4571875], [-656.25, -671.92578125], [-708.4571875], [-581.5, -490.75], [-535.5625, -671.92578125], [-491.25, -651.953125], [-446.25125, -630.83203125], [-406.75, -605.625], [-366.65625, -585.14453125], [-328.0, -560.578125], [-290.25, -521.84375], [-251.5, -483.65625], [-212.75, -445.46875], [-174.0, -407.28125], [-134.25, -369.09375], [-94.5, -330.875], [-54.75, -292.65625], [-14.0, -254.4375], [-71.25, -216.225], [-126.25, -187.0125], [-181.5, -157.79375], [-235.75, -128.575], [-290.0, -99.35625], [-344.25, -69.1375], [-398.5, -39.91875], [-452.75, -9.699375], [-507.0, -0.379375], [-561.25, 38.91875], [-615.5, 78.0], [-669.75, 117.11875], [-724.0, 156.225], [-778.25, 195.3375], [-832.5, 234.44375], [-886.75, 273.55125], [-941.0, 312.65625], [-995.25, 351.76375], [-1049.5, 390.875], [-1000.0, -800.0]]</pre> <p>Runtime: 0.0 ms</p> <p>OK</p> <p>Code</p> <p>Code</p>	
	Runtime: 20.2960 ms	Runtime: 12.947 ms
TC 10 GUI 7 titik 5 iterasi	<p>Number of Points: 7 Number of Iterations: 5 List of Points: -8, -8 -1, -4 2, 2 6, 8 10, 0 9, -3 0, 1 Solve Format input: x1,y1 x2,y2 x3,y3... Each point is split by a space Each x and y is split by a comma</p> <p>Bezier Graphic</p> <p>(0, 1)</p> <p>(-8.0, -8.0)</p> <p>Points info</p> <pre>[[[-8.0, -8.0], [-6.74312815445881, -7.2208731350553], [-5.49666666666667, -6.88333333333333], [-4.2500000000000, -6.54999999999999], [-3.00333333333333, -6.21666666666667], [-1.75666666666667, -5.88333333333333], [-0.5100000000000, -5.5500000000000], [1.73333333333333, -5.21666666666667], [3.48666666666667, -4.88333333333333], [5.2400000000000, -4.5500000000000], [7.00333333333333, -4.21666666666667], [8.76666666666667, -3.88333333333333], [10.5300000000000, -3.5500000000000], [12.2933333333333, -3.21666666666667], [14.0566666666667, -2.88333333333333], [15.8200000000000, -2.5500000000000], [17.5833333333333, -2.21666666666667], [19.3466666666667, -1.88333333333333], [21.1100000000000, -1.5500000000000], [22.8733333333333, -1.21666666666667], [24.6366666666667, -0.88333333333333], [26.4000000000000, -0.5500000000000], [28.1633333333333, -0.21666666666667], [29.9266666666667, 0.11666666666667], [31.6899999999999, 0.4500000000000], [33.4533333333333, 0.78333333333333], [35.2166666666667, 1.11666666666667], [37.0000000000000, 1.4500000000000], [38.7833333333333, 1.78333333333333], [40.5466666666667, 2.11666666666667], [42.3000000000000, 2.4500000000000], [44.0533333333333, 2.78333333333333], [45.8166666666667, 3.11666666666667], [47.5799999999999, 3.4500000000000], [49.3433333333333, 3.78333333333333], [51.1066666666667, 4.11666666666667], [52.8700000000000, 4.4500000000000], [54.6333333333333, 4.78333333333333], [56.3966666666667, 5.11666666666667], [58.1600000000000, 5.4500000000000], [60.0233333333333, 5.78333333333333], [61.7866666666667, 6.11666666666667], [63.5500000000000, 6.4500000000000], [65.3133333333333, 6.78333333333333], [67.0766666666667, 7.11666666666667], [68.8400000000000, 7.4500000000000], [70.6033333333333, 7.78333333333333], [72.3666666666667, 8.11666666666667], [74.1300000000000, 8.4500000000000], [75.8933333333333, 8.78333333333333], [77.6566666666667, 9.11666666666667], [79.4199999999999, 9.4500000000000], [81.1833333333333, 9.78333333333333], [82.9466666666667, 10.1166666666667], [84.7100000000000, 10.4500000000000], [86.4733333333333, 10.78333333333333], [88.2366666666667, 11.11666666666667], [90.0000000000000, 11.4500000000000], [91.7633333333333, 11.78333333333333], [93.5266666666667, 12.11666666666667], [95.2899999999999, 12.4500000000000], [97.0533333333333, 12.78333333333333], [98.8166666666667, 13.11666666666667], [100.5800000000000, 13.4500000000000], [102.3433333333333, 13.78333333333333], [104.1066666666667, 14.11666666666667], [105.8700000000000, 14.4500000000000], [107.6333333333333, 14.78333333333333], [109.3966666666667, 15.11666666666667], [111.1600000000000, 15.4500000000000], [112.9233333333333, 15.78333333333333], [114.6866666666667, 16.11666666666667], [116.4500000000000, 16.4500000000000], [118.2133333333333, 16.78333333333333], [120.0000000000000, 17.11666666666667], [121.7633333333333, 17.4500000000000], [123.5266666666667, 17.78333333333333], [125.2899999999999, 18.11666666666667], [127.0533333333333, 18.4500000000000], [128.8166666666667, 18.78333333333333], [130.5800000000000, 19.11666666666667], [132.3433333333333, 19.4500000000000], [134.1066666666667, 19.78333333333333], [135.8700000000000, 20.11666666666667], [137.6333333333333, 20.4500000000000], [139.3966666666667, 20.78333333333333], [141.1600000000000, 21.11666666666667], [142.9233333333333, 21.4500000000000], [144.6866666666667, 21.78333333333333], [146.4500000000000, 22.11666666666667], [148.2133333333333, 22.4500000000000], [150.0000000000000, 22.78333333333333], [151.7633333333333, 23.11666666666667], [153.5266666666667, 23.4500000000000], [155.2899999999999, 23.78333333333333], [157.0533333333333, 24.11666666666667], [158.8166666666667, 24.4500000000000], [160.5800000000000, 24.78333333333333], [162.3433333333333, 25.11666666666667], [164.1066666666667, 25.4500000000000], [165.8700000000000, 25.78333333333333], [167.6333333333333, 26.11666666666667], [169.3966666666667, 26.4500000000000], [171.1600000000000, 26.78333333333333], [172.9233333333333, 27.11666666666667], [174.6866666666667, 27.4500000000000], [176.4500000000000, 27.78333333333333], [178.2133333333333, 28.11666666666667], [180.0000000000000, 28.4500000000000], [181.7633333333333, 28.78333333333333], [183.5266666666667, 29.11666666666667], [185.2899999999999, 29.4500000000000], [187.0533333333333, 29.78333333333333], [188.8166666666667, 30.11666666666667], [190.5800000000000, 30.4500000000000], [192.3433333333333, 30.78333333333333], [194.1066666666667, 31.11666666666667], [195.8700000000000, 31.4500000000000], [197.6333333333333, 31.78333333333333], [199.3966666666667, 32.11666666666667], [201.1600000000000, 32.4500000000000], [202.9233333333333, 32.78333333333333], [204.6866666666667, 33.11666666666667], [206.4500000000000, 33.4500000000000], [208.2133333333333, 33.78333333333333], [210.0000000000000, 34.11666666666667], [211.7633333333333, 34.4500000000000], [213.5266666666667, 34.78333333333333], [215.2899999999999, 35.11666666666667], [217.0533333333333, 35.4500000000000], [218.8166666666667, 35.78333333333333], [220.5800000000000, 36.11666666666667], [222.3433333333333, 36.4500000000000], [224.1066666666667, 36.78333333333333], [225.8700000000000, 37.11666666666667], [227.6333333333333, 37.4500000000000], [229.3966666666667, 37.78333333333333], [231.1600000000000, 38.11666666666667], [232.9233333333333, 38.4500000000000], [234.6866666666667, 38.78333333333333], [236.4500000000000, 39.11666666666667], [238.2133333333333, 39.4500000000000], [240.0000000000000, 39.78333333333333], [241.7633333333333, 40.11666666666667], [243.5266666666667, 40.4500000000000], [245.2899999999999, 40.78333333333333], [247.0533333333333, 41.11666666666667], [248.8166666666667, 41.4500000000000], [250.5800000000000, 41.78333333333333], [252.3433333333333, 42.11666666666667], [254.1066666666667, 42.4500000000000], [255.8700000000000, 42.78333333333333], [257.6333333333333, 43.11666666666667], [259.3966666666667, 43.4500000000000], [261.1600000000000, 43.78333333333333], [262.9233333333333, 44.11666666666667], [264.6866666666667, 44.4500000000000], [266.4500000000000, 44.78333333333333], [268.2133333333333, 45.11666666666667], [270.0000000000000, 45.4500000000000], [271.7633333333333, 45.78333333333333], [273.5266666666667, 46.11666666666667], [275.2899999999999, 46.4500000000000], [277.0533333333333, 46.78333333333333], [278.8166666666667, 47.11666666666667], [280.5800000000000, 47.4500000000000], [282.3433333333333, 47.78333333333333], [284.1066666666667, 48.11666666666667], [285.8700000000000, 48.4500000000000], [287.6333333333333, 48.78333333333333], [289.3966666666667, 49.11666666666667], [291.1600000000000, 49.4500000000000], [292.9233333333333, 49.78333333333333], [294.6866666666667, 50.11666666666667], [296.4500000000000, 50.4500000000000], [298.2133333333333, 50.78333333333333], [300.0000000000000, 51.11666666666667], [301.7633333333333, 51.4500000000000], [303.5266666666667, 51.78333333333333], [305.2899999999999, 52.11666666666667], [307.0533333333333, 52.4500000000000], [308.8166666666667, 52.78333333333333], [310.5800000000000, 53.11666666666667], [312.3433333333333, 53.4500000000000], [314.1066666666667, 53.78333333333333], [315.8700000000000, 54.11666666666667], [317.6333333333333, 54.4500000000000], [319.3966666666667, 54.78333333333333], [321.1600000000000, 55.11666666666667], [322.9233333333333, 55.4500000000000], [324.6866666666667, 55.78333333333333], [326.4500000000000, 56.11666666666667], [328.2133333333333, 56.4500000000000], [330.0000000000000, 56.78333333333333], [331.7633333333333, 57.11666666666667], [333.5266666666667, 57.4500000000000], [335.2899999999999, 57.78333333333333], [337.0533333333333, 58.11666666666667], [338.8166666666667, 58.4500000000000], [340.5800000000000, 58.78333333333333], [342.3433333333333, 59.11666666666667], [344.1066666666667, 59.4500000000000], [345.8700000000000, 59.78333333333333], [347.6333333333333, 60.11666666666667], [349.3966666666667, 60.4500000000000], [351.1600000000000, 60.78333333333333], [352.9233333333333, 61.11666666666667], [354.6866666666667, 61.4500000000000], [356.4500000000000, 61.78333333333333], [358.2133333333333, 62.11666666666667], [360.0000000000000, 62.4500000000000], [361.7633333333333, 62.78333333333333], [363.5266666666667, 63.11666666666667], [365.2899999999999, 63.4500000000000], [367.0533333333333, 63.78333333333333], [368.8166666666667, 64.11666666666667], [370.5800000000000, 64.4500000000000], [372.3433333333333, 64.78333333333333], [374.1066666666667, 65.11666666666667], [375.8700000000000, 65.4500000000000], [377.6333333333333, 65.78333333333333], [379.3966666666667, 66.11666666666667], [381.1600000000000, 66.4500000000000], [382.9233333333333, 66.78333333333333], [384.6866666666667, 67.11666666666667], [386.4500000000000, 67.4500000000000], [388.2133333333333, 67.78333333333333], [390.0000000000000, 68.11666666666667], [391.7633333333333, 68.4500000000000], [393.5266666666667, 68.78333333333333], [395.2899999999999, 69.11666666666667], [397.0533333333</pre>	

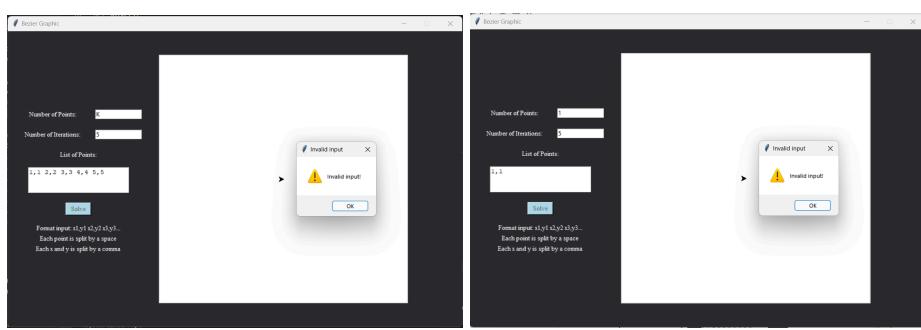
TC 12
GUI
3 titik
8 iterasi



Runtime: 98.248004913 ms

Runtime: 63.57288360595703 ms

TC 13
Input
handling
GUI



TC 14 5 titik 3 iterasi	<p>Selamat datang di generator kurva berzier!</p> <p>Banyak titik yang dapat dipilih: 1. 3 titik 2. n titik</p> <p>Silakan masukkan pilihan: 2</p> <p>Silakan masukkan banyaknya iterasi: 3</p> <p>Silakan masukkan banyaknya banyak pasangan titik: 5</p> <p>Silahkan masukkan koodinat x1 dan y1 dipisahkan spasi (dalam float) tanpa spasi di akhir: 1 1</p> <p>Silahkan masukkan koodinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: 2 2</p> <p>Silahkan masukkan koodinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir: 3 3</p> <p>Silahkan masukkan koodinat x4 dan y4 dipisahkan spasi (dalam float) tanpa spasi di akhir: 4 4</p> <p>Silahkan masukkan koodinat x5 dan y5 dipisahkan spasi (dalam float) tanpa spasi di akhir: 5 5</p> <p>Metode penyelesaian: 1. Divide and Conquer 2. Brute Force</p> <p>Silakan masukkan pilihan: 1</p> <p>Hasil dengan metode divide and conquer:</p> <ul style="list-style-type: none"> [1.0, 1.0] [1.5, 1.5] [2.0, 2.0] [2.5, 2.5] [3.0, 3.0] [3.5, 3.5] [4.0, 4.0] [4.5, 4.5] [5.0, 5.0] <p>Silakan masukkan nama file gambar: test14</p> <p>Gambar kurva berhasil disimpan</p> <p>Runtime: 10.998010635375977 ms</p>	
	Runtime: 10.998 ms	Runtime: 0.0 ms
TC 15 5 titik 2 iterasi	<p>Selamat datang di generator kurva berzier!</p> <p>Banyak titik yang dapat dipilih: 1. 3 titik 2. n titik</p> <p>Silakan masukkan pilihan: 2</p> <p>Silakan masukkan banyaknya iterasi: 2</p> <p>Silakan masukkan banyaknya banyak pasangan titik: 8</p> <p>Silahkan masukkan koodinat x1 dan y1 dipisahkan spasi (dalam float) tanpa spasi di akhir: 1 1</p> <p>Silahkan masukkan koodinat x2 dan y2 dipisahkan spasi (dalam float) tanpa spasi di akhir: 2 2</p> <p>Silahkan masukkan koodinat x3 dan y3 dipisahkan spasi (dalam float) tanpa spasi di akhir: 3 3</p> <p>Silahkan masukkan koodinat x4 dan y4 dipisahkan spasi (dalam float) tanpa spasi di akhir: 4 4</p> <p>Silahkan masukkan koodinat x5 dan y5 dipisahkan spasi (dalam float) tanpa spasi di akhir: 5 5</p> <p>Silahkan masukkan koodinat x6 dan y6 dipisahkan spasi (dalam float) tanpa spasi di akhir: 6 6</p> <p>Silahkan masukkan koodinat x7 dan y7 dipisahkan spasi (dalam float) tanpa spasi di akhir: 7 7</p> <p>Silahkan masukkan koodinat x8 dan y8 dipisahkan spasi (dalam float) tanpa spasi di akhir: 8 8</p> <p>Metode penyelesaian: 1. Divide and Conquer 2. Brute Force</p> <p>Silakan masukkan pilihan: 1</p> <p>Hasil dengan metode divide and conquer:</p> <ul style="list-style-type: none"> [1.0, 1.0] [2.75, 2.75] [4.5, 4.5] [6.25, 6.25] [8.0, 8.0] <p>Silakan masukkan nama file gambar: test15</p> <p>Gambar kurva berhasil disimpan</p> <p>Runtime: 8.999824523925781 ms</p>	
	Runtime: 8.9998 ms	Runtime: 0.0 ms

BAB 5

Pembahasan

Pada program ini, algoritma *brute force* memiliki kompleksitas $O(2^k n)$ dan algoritma *divide and conquer* memiliki kompleksitas $O(2^k n^2)$ dengan k adalah banyaknya iterasi dan n adalah banyak titik kontrol. Algoritma *divide and conquer* pada program ini memiliki kompleksitas yang lebih tinggi karena pada iterasi dasar memiliki kompleksitas yang cukup tinggi yaitu $g(n) = \frac{n^2+n}{2} - 1$. Selain itu, pada implementasi aslinya, algoritma *divide and conquer* memiliki beberapa operasi tambahan untuk melakukan visualisasi kurva. Hal ini tentu akan memiliki kompleksitas algoritmanya. Maka dari itu, algoritma *brute force* lebih efisien daripada algoritma *divide and conquer*. Hal ini juga terbukti dari uji coba yang telah dilakukan. Dari 13 uji coba komparasi yang dilakukan, algoritma *brute force* memiliki durasi yang lebih singkat daripada algoritma *divide and conquer* pada 11 uji coba.

BAB 6

Penjelasan Bonus

6.1. Generalisasi

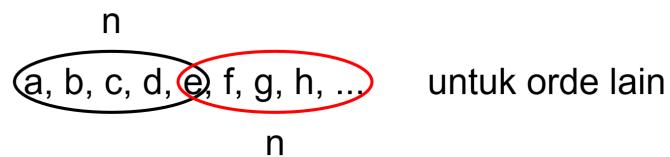
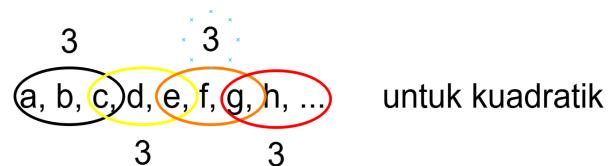
Perbedaan antara kurva Bézier kuadratik dan kurva Bézier orde lainnya terletak pada iterasi dasar dan pembagian titik-titik kontrol untuk dilakukan proses iterasi dasarnya. Proses iterasi dasar untuk kuadratik perlu 2 kali proses perhitungan list titik tengah. Sedangkan, proses iterasi dasar untuk orde lainnya perlu (jumlah titik kontrol - 1) kali proses perhitungan list titik tengah. Berikut ilustrasi iterasi dasarnya

$$[a, b, c] \xrightarrow[1]{\text{proses}} [\overline{ab}, \overline{bc}] \xrightarrow[2]{\text{proses}} [\overline{\overline{ab}bc}], \text{ iterasi dasar untuk kuadratik}$$

$$[u_1, u_2, u_3, \dots, u_{n-2}, u_{n-1}, u_n] \xrightarrow[1]{\text{proses}} [\overline{u_1u_2}, \overline{u_2u_3}, \dots, \overline{u_{n-1}u_n}] \xrightarrow[2]{\text{proses}} \dots$$

$$\xrightarrow[n-1]{\text{proses}} [\overline{\overline{u_1u_2\dots u_{n-1}u_n}}], \text{ iterasi dasar untuk lainnya}$$

Pembagian titik kontrol pada kurva Bézier kuadratik dan kurva Bézier orde lain terletak pada jumlah titik kontrol yang diambil. Titik kontrol yang diambil untuk kurva Bézier kuadratik adalah 3. Untuk orde lain, titik kontrol yang diambil adalah sebanyak titik kontrol awal kurva tersebut. Berikut ilustrasinya,



Program ini telah melakukan generalisasi. Jadi, algoritma pembuatan kurva Bézier kuadratik tidak diimplementasikan khusus. Namun, pembuatan kurva Bézier kuadratik sudah tergabung pada algoritma pembuatan kurva Bézier general.

6.2. GUI

6.2.1. Source Code

```
import tkinter as tk
import turtle
import app
import copy
import time

running = False
global multiplier

def solve():
    global running
    if running:
        return
    running = True
    solve_button.config(state=tk.DISABLED)
    pass
    t.clear()
    num_of_iteration = num_iterations_entry.get()
    num_of_point = num_points_entry.get()
    list_of_point = list_of_point_entry.get("1.0", "end")
    list_of_point = list_of_point.split(" ")
    try:
        for i in range(len(list_of_point)):
            point = list_of_point[i].split(",")
            list_of_point[i] = [float(xy) for xy in point]
    num_of_iteration = int(num_of_iteration)
    num_of_point = int(num_of_point)
    if num_of_iteration <= 0 or num_of_point <= 2:
        raise ValueError
    if num_of_point != len(list_of_point):
        raise ValueError
    except ValueError:
        tk.messagebox.showwarning(title="Invalid input",
message="Invalid input!")
        running = False
        solve_button.config(state=tk.NORMAL)
        return
    list_step = []
    start = time.time()
    result_dnc = app.general_iterate(num_of_point,
num_of_iteration, 0, list_of_point, list_step)
    result_dnc = app.take_result_point(result_dnc,
num_of_point)
    end = time.time()
```

```

        runtime = (end - start)*1000
        draw_bezier(list_step)
        draw_bezier_final(result_dnc)
        # result_dnc_str = '\n'.join(result_dnc)
        message = f'{result_dnc}\nRuntime: {runtime} ms'
        tk.messagebox.showinfo(title="Points info",
message=message)
        running = False
        solve_button.config(state=tk.NORMAL)

def add_scrollbar(content):
    scrollbar_frame = tk.Frame(left_frame)
    scrollbar_frame.grid(row=7, column=0, columnspan=2,
rowspan=3, pady=5)
    scrollbar = tk.Scrollbar(scrollbar_frame,
orient="vertical", bg="white", fg="black", font=("Palatino",
10))
    scrollbar.grid(row=0, column=1, sticky="ns")
    text_widget = tk.Text(scrollbar_frame, width=25,
height=5, yscrollcommand=scrollbar.set)
    text_widget.grid(row=0, column=0)
    scrollbar.config(command=text_widget.yview)
    for item in content:
        text_widget.insert(tk.END, item + "\n")

def draw_bezier(points):
    t.penup()
    max_value = 0
    for sublist in points:
        for point in sublist:
            for xy in point:
                max_value= max(max_value, abs(xy))
    global multiplier
    multiplier = 220 / max_value
    if (multiplier > 1):
        multiplier = 220 // max_value

    scaling_points = []
    for sublist1 in points:
        multiplied_sublist1 = []
        for sublist2 in sublist1:
            multiplied_sublist2 = []
            for value in sublist2:
                multiplied_sublist2.append(value *
multiplier)
            multiplied_sublist1.append(multiplied_sublist2)
        scaling_points.append(multiplied_sublist1)
    draw_axes(multiplier)
    draw_curve(points, scaling_points)

def draw_curve(points, scaling_points):
    t.color("#219EBC")
    for sublist in scaling_points:
        t.penup()
        t.goto(sublist[0][0], sublist[0][1])
        t.pendown()
        for point in sublist[1:]:
            t.goto(point[0], point[1])

```

```

        t.penup()
    return

def draw_bezier_final(points):
    scaling_points = copy.deepcopy(points)
    for i in range(len(points)):
        scaling_points[i][0] *= multiplier
        scaling_points[i][1] *= multiplier
    draw_curve_final(points, scaling_points)

def draw_curve_final(points, scaling_points):
    t.penup()
    t.goto(scaling_points[0][0], scaling_points[0][1])
    t.color("black")
    t.pendown()
    for point in scaling_points:
        t.goto(point[0], point[1])
        if (point == scaling_points[0]):
            if (points[0][0] < 0):
                t.write(f"({points[0][0]}, {points[0][1]}), "
align="left")
            else:
                t.write(f"({points[0][0]}, {points[0][1]}), "
align="right")
            if (point == scaling_points[-1]):
                if (points[-1][0] < 0):
                    t.write(f"({points[-1][0]}, "
{points[-1][1]}), align="left")
                else:
                    t.write(f"({points[-1][0]}, "
{points[-1][1]}), align="right")

def draw_axes(multiplier):
    t.color("lightgrey")
    t.penup()
    t.goto(-canvas.winfo_width() / 2, 0)
    t.pendown()
    t.goto(canvas.winfo_width() / 2, 0)
    t.penup()
    t.goto(0, -canvas.winfo_height() / 2)
    t.pendown()
    t.goto(0, canvas.winfo_height() / 2)
    t.penup()

    x = -200
    while x < 201:
        t.goto(x, -5)
        t.pendown()
        t.goto(x, 5)
        t.penup()
        t.goto(x, -20)
        t.write(str(round(x / multiplier, 1)),
align="center")
        x += 50

    y = -200
    while y < 201:
        t.goto(-5, y)

```

```

        t.pendown()
        t.goto(5, y)
        t.penup()
        t.goto(-20, y)
        t.write(str(round((y / multiplier), 1)),
align="right")
        y += 50

    t.penup()

# Initialization
root = tk.Tk()
root.title("Bezier Graphic")
root.configure(bg="#2C2B30")

# Fix size
root.geometry("920x600")
root.resizable(False, False)

# Left side
left_frame = tk.Frame(root, bg="#2C2B30", padx=10, pady=10)
left_frame.pack(side=tk.LEFT)

# Right side
right_frame = tk.Frame(root, bg="#2C2B30", padx=10, pady=10)
right_frame.pack(side=tk.LEFT)

left_frame.grid_columnconfigure(1, weight=1)
right_frame.grid_columnconfigure(0, weight=2)

# Num of point
num_points_label = tk.Label(left_frame, text="Number of Points:",
bg="#2C2B30", fg="white", font=("Palatino", 10))
num_points_label.grid(row=0, column=0, pady=10, padx=30)
num_points_entry = tk.Entry(left_frame, width=15)
num_points_entry.grid(row=0, column=1, padx=5, pady=5)

# Num of iteration
num_iterations_label = tk.Label(left_frame, text="Number of Iterations:",
bg="#2C2B30", fg="white", font=("Palatino", 10))
num_iterations_label.grid(row=1, column=0, pady=10)
num_iterations_entry = tk.Entry(left_frame, width=15)
num_iterations_entry.grid(row=1, column=1, padx=5, pady=5)

# List of point
list_of_point = tk.Label(left_frame, text="List of Points:",
bg="#2C2B30", fg="white", font=("Palatino", 10))
list_of_point.grid(row=2, column=0, pady=10, columnspan=2)
list_of_point_entry = tk.Text(left_frame, width=25,
height=3)
list_of_point_entry.grid(row=3, column=0, columnspan=2,
pady=5)

# Solve button
solve_button = tk.Button(left_frame, text="Solve",
command=solve, bg="lightblue", font=("Palatino", 10),
width=6, height=1)

```

```

solve_button.grid(row=6, columnspan=2, pady=15)

label1 = tk.Label(left_frame, text="Format input: x1,y1  
x2,y2 x3,y3...", bg="#2C2B30", fg="white", font=("Palatino", 10))
label2 = tk.Label(left_frame, text="Each point is split by a  
space", bg="#2C2B30", fg="white", font=("Palatino", 10))
label3 = tk.Label(left_frame, text="Each x and y is split by  
a comma", bg="#2C2B30", fg="white", font=("Palatino", 10))
label1.grid(row=7, columnspan=2)
label2.grid(row=8, columnspan=2)
label3.grid(row=9, columnspan=2)

# Turtle canvas
canvas_frame = tk.Frame(right_frame, width=600, height=600,
bg="#2C2B30")
canvas_frame.pack(padx=10, pady=10)
canvas = tk.Canvas(canvas_frame, width=500, height=500,
bg="#2C2B30")
canvas.pack()
screen = turtle.TurtleScreen(canvas)
t = turtle.RawTurtle(screen)

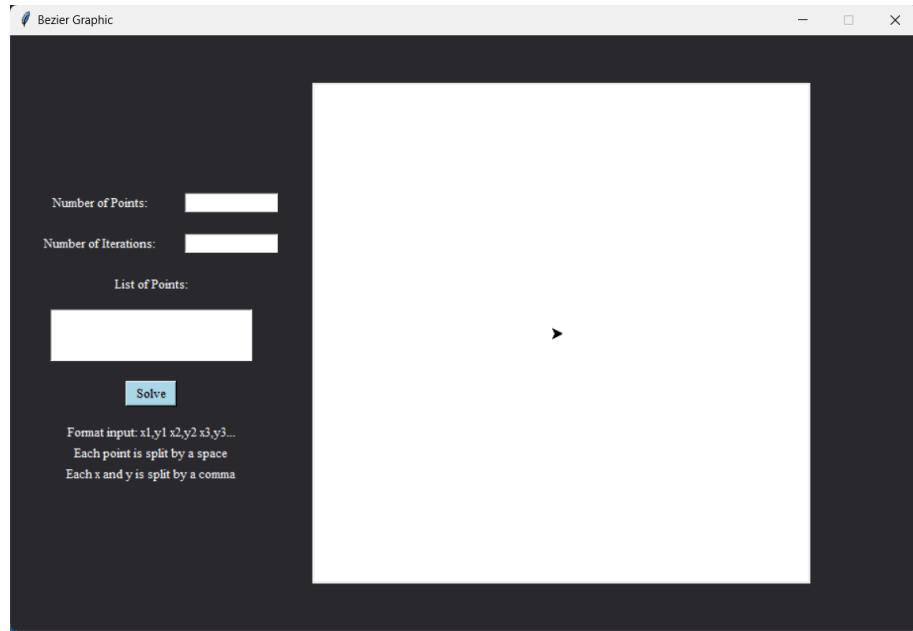
root.mainloop()

```

6.2.2. Penjelasan

GUI menggunakan Tkinter dan turtle python. Tkinter digunakan untuk membuat antarmuka pengguna, sedangkan turtle digunakan untuk menampilkan gambar.

6.2.3. Cara Penggunaan



1. Input banyaknya point (number of point), banyaknya iterasi (number of iteration) dan list pasangan titik. Ada beberapa ketentuan input:
 - Number of point harus bilangan bulat lebih dari 2;
 - Number of iteration harus bilangan bulat lebih dari 0;
 - Format pasangan titik: $x_1, y_1 \ x_2, y_2 \ x_3, y_3, \dots$, dst, dimana setiap pasangan titik dipisahkan oleh spasi dan tiap x dan y dipisahkan oleh tanda koma. Tidak boleh ada karakter tambahan di akhir (spasi, enter, titik, koma, dll).
 - Jumlah pasangan titik harus sama dengan number of point.

Jika tidak semua ketentuan terpenuhi, maka program akan menampilkan pesan error.

2. Klik solve button. Setelah solve button diklik dan program belum selesai, maka solve button akan berada dalam mode *disabled*. Setelah program selesai, program akan menampilkan pesan berisi titik-titik hasil dan waktu.
3. Jika ingin melakukan penghitungan ulang, masukkan input setelah program sebelumnya selesai, kemudian klik solve. Program akan menghitung dan menggambar ulang.

BAB 7

Kesimpulan

Berdasarkan program yang kami buat, dapat disimpulkan bahwa program berjalan dengan baik dan lancar. Program dapat menerima masukan dari *command line interface* (CLI) dan *Graphical User Interface* (GUI). Program dapat melakukan penghitungan dengan metode *brute force* dan *divide and conquer*. Program dapat mengatasi kesalahan masukan dan akan meminta pengguna memasukkan ulang.

DAFTAR PUSTAKA

- Chavan, A. (2024, January 28). Structures in Python - the startup - medium. Medium.
<https://medium.com/swlh/structures-in-python-ed199411b3e1>
- Munir, Rinaldi. 2024. Algoritma Divide and Conquer bagian 1. Bandung: Institut Teknologi Bandung.[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)
- Munir, Rinaldi. 2024. Algoritma Divide and Conquer bagian 2. Bandung: Institut Teknologi Bandung.[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-\(2024\)-Bagian2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian2.pdf)
- Munir, Rinaldi. 2024. Algoritma Divide and Conquer bagian 3. Bandung: Institut Teknologi Bandung.[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-\(2024\)-Bagian3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian3.pdf)
- Munir, Rinaldi. 2024. Algoritma Divide and Conquer bagian 4. Bandung: Institut Teknologi Bandung.[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)

LAMPIRAN

Github Repository

https://github.com/evelynnn04/Tucil2_13522054_13522083.git

Checklist Keberhasilan Program

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