

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

Dipersiapkan oleh:

Pradipta Rafa Mahesa

13522162

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung
Jl. Ganesha 10, Bandung 4013

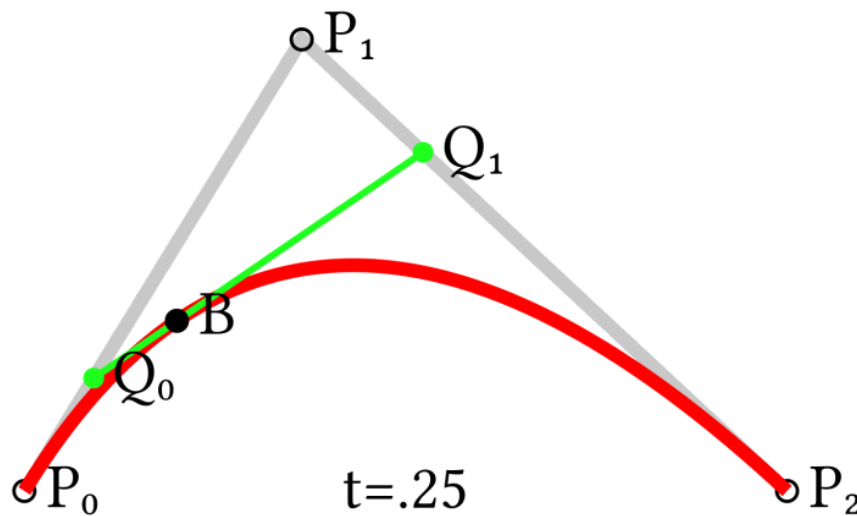
Daftar Isi

BAB I.....	3
Deskripsi Tugas.....	3
BAB 2.....	5
Algoritma Brute Force.....	5
3.1. Mapping Persoalan.....	5
3.2. Implementasi.....	6
BAB 3.....	7
Algoritma Divide and Conquer.....	7
4.1. Analisis.....	7
4.2. Implementasi.....	8
BAB 4.....	10
Source Code.....	10
4.1. main.py.....	10
4.2. brute_bezier.py.....	11
4.3. dnc_bezier.py.....	12
BAB 5.....	14
Test Case.....	14
5.1. Brute Force.....	14
5.2. Divide and Conquer.....	16
5.3. Analisis.....	18
Lampiran.....	19
Daftar Pustaka.....	20

BAB I

Deskripsi Tugas

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.

Ilustrasi kasus kurva bezier adalah sebagai berikut. terdapat tiga buah titik, P_0 , P_1 , dan P_2 , dengan titik P_1 menjadi titik kontrol antara

1. Buatlah sebuah titik baru Q_0 yang berada di tengah garis yang menghubungkan P_0 dan P_1 , serta titik Q_1 yang berada di tengah garis yang menghubungkan P_1 dan P_2 .
2. Hubungkan Q_0 dan Q_1 sehingga terbentuk sebuah garis baru.
3. Buatlah sebuah titik baru R_0 yang berada di tengah Q_0 dan Q_1 .
4. Buatlah sebuah garis yang menghubungkan $P_0 - R_0 - P_2$.

Melalui proses di atas, telah dilakukan 1 buah iterasi dan diperoleh sebuah “kurva” yang belum cukup mulus dengan aproksimasi 3 buah titik. Untuk membuat sebuah kurva yang lebih baik, perlu dilakukan iterasi lanjutan. Berikut adalah prosedurnya :

1. Buatlah beberapa titik baru, yaitu S0 yang berada di tengah P0 dan Q0, S1 yang berada di tengah Q0 dan R0, S2 yang berada di tengah R0 dan Q1, dan S3 yang berada di tengah Q1 dan P2.
2. Hubungkan S0 dengan S1 dan S2 dengan S3 sehingga terbentuk garis baru.
3. Buatlah dua buah titik baru, yaitu T0 yang berada di tengah S0 dan S1, serta T1 yang berada di tengah S2 dan S3.
4. Buatlah sebuah garis yang menghubungkan P0 - T0 - R0 - T1 - P2.

Melalui iterasi kedua akan tampak semakin mendekati sebuah kurva, dengan aproksimasi 5 buah titik. Anda dapat membuat visualisasi atau gambaran secara mandiri terkait hal ini sehingga dapat diamati dan diterka dengan jelas bahwa semakin banyak iterasi yang dilakukan, maka akan membentuk sebuah kurva yang tidak lain adalah kurva Bézier.

Spesifikasi dari Tugas Kecil 2 :

- Buatlah program sederhana dalam bahasa C++/Python/JavaScript/Go yang dapat membentuk sebuah kurva Bézier kuadratik dengan menggunakan algoritma titik tengah berbasis divide and conquer.
- Selain implementasi dalam algoritma divide and conquer, program juga diminta untuk diimplementasikan dalam algoritma brute force. Solusi ini ditujukan sebagai pembandingan dengan solusi sebelumnya.
- Tugas dapat dikerjakan individu atau berkelompok dengan anggota maksimal 2 orang (sangat disarankan). Boleh lintas kelas dan lintas kampus, tetapi tidak boleh sama dengan anggota kelompok pada tugas-tugas Strategi Algoritma sebelumnya.
- Input :

Format masukan dibebaskan, dengan catatan dijelaskan pada README dan laporan.

Komponen yang perlu menjadi masukan yaitu.

1. Tiga buah pasangan titik. Sebagai catatan, titik yang paling awal dimasukkan akan menjadi titik awal kurva, begitu juga dengan titik yang paling akhir.
 2. Jumlah iterasi yang ingin dilakukan.
- Output :

Berikut adalah luaran dari program yang diekspektasikan.

1. Hasil kurva Bézier yang terbentuk pada iterasi terkait. Kakas yang digunakan untuk visualisasi dibebaskan.
2. Waktu eksekusi program pembentukan kurva.

BAB 2

Algoritma Brute Force

3.1. Mapping Persoalan

Mengulas lebih jauh mengenai bagaimana sebuah kurva Bézier bisa terbentuk, misalkan diberikan dua buah titik P0 dan P1 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 Q0 yang berada pada garis yang dibentuk oleh P0 dan P1, 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 P0 ke P1. Misalnya ketika $t = 0.25$, maka B(t) adalah seperempat jalan dari titik P0 ke P1. sehingga seluruh rentang variasi nilai t dari 0 hingga 1 akan membuat persamaan B(t) membentuk sebuah garis lurus dari P0 ke P1. Misalkan selain dua titik sebelumnya ditambahkan sebuah titik baru, sebut saja P2, dengan P0 dan P2 sebagai titik kontrol awal dan akhir, dan P1 menjadi titik kontrol antara. Dengan menyatakan titik Q1 terletak diantara garis yang menghubungkan P1 dan P2, dan membentuk kurva Bézier linier yang berbeda dengan kurva letak Q0 berada, maka dapat dinyatakan sebuah titik baru, R0 yang berada diantara garis yang menghubungkan Q0 dan Q1 yang bergerak membentuk kurva Bézier kuadratik terhadap titik P0 dan P2. 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 Q0 dan Q1, maka diperoleh persamaan sebagai berikut.

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

3.2. Implementasi

No	Kode	Deskripsi
1	<pre>def bruteforce(t0 :list ,t1 : list,t2 : list,n : int): x = [t0[0]] y = [t0[1]] for i in range (1,n+1): t = i/n b0 = (1 - t) ** 2 b1 = 2*t*(1 - t) b2 = t ** 2 x.append(t0[0] * b0 + t1[0] * b1 + t2[0] * b2) y.append(t0[1] * b0 + t1[1] * b1 + t2[1] * b2) return x,y</pre>	<p>algoritma ini menerima 3 titik (t0,t1,t2) dan banyak iterasi (n) dan menghasilkan list x,y (koordinat dalam sumbu x dan y) yang awalnya dimasukkan titik awal (t0). Dalam loop sebanyak jumlah iterasi ditentukan t (fase) lalu memasukkannya ke dalam rumus dan didapatkan koordinat x dan y nya yang lalu ditambahkan ke dalam list x dan y.</p>

BAB 3

Algoritma Divide and Conquer

3.1. Analisis

Dari Ilustrasi Kasus :

Terdapat tiga buah titik, P0, P1, dan P2, dengan titik P1 menjadi titik kontrol antara

5. Buatlah sebuah titik baru Q0 yang berada di tengah garis yang menghubungkan P0 dan P1, serta titik Q1 yang berada di tengah garis yang menghubungkan P1 dan P2.
6. Hubungkan Q0 dan Q1 sehingga terbentuk sebuah garis baru.
7. Buatlah sebuah titik baru R0 yang berada di tengah Q0 dan Q1.
8. Buatlah sebuah garis yang menghubungkan P0 - R0 - P2.

Melalui proses di atas, telah dilakukan 1 buah iterasi dan diperoleh sebuah “kurva” yang belum cukup mulus dengan aproksimasi 3 buah titik. Untuk membuat sebuah kurva yang lebih baik, perlu dilakukan iterasi lanjutan. Berikut adalah prosedurnya :

5. Buatlah beberapa titik baru, yaitu S0 yang berada di tengah P0 dan Q0, S1 yang berada di tengah Q0 dan R0, S2 yang berada di tengah R0 dan Q1, dan S3 yang berada di tengah Q1 dan P2.
6. Hubungkan S0 dengan S1 dan S2 dengan S3 sehingga terbentuk garis baru.
7. Buatlah dua buah titik baru, yaitu T0 yang berada di tengah S0 dan S1, serta T1 yang berada di tengah S2 dan S3.
8. Buatlah sebuah garis yang menghubungkan P0 - T0 - R0 - T1 - P2.

Melalui iterasi kedua akan tampak semakin mendekati sebuah kurva, dengan aproksimasi 5 buah titik. Anda dapat membuat visualisasi atau gambaran secara mandiri terkait hal ini sehingga dapat diamati dan diterka dengan jelas bahwa semakin banyak iterasi yang dilakukan, maka akan membentuk sebuah kurva yang tidak lain adalah kurva Bézier.



notes :

- T0- T1 : titik tengah dari T0 dan T1
- (T0-T1-T2) : titik tengah dari (T0-T1) dan (T1-T2)

3.2. Implementasi

No	Kode	Deskripsi
1	<pre>def get_mid(t0,t1): t = [] t.append((t0[0]+t1[0])/2) t.append((t0[1]+t1[1])/2) #print("mid "+str(t0)+" , "+str(t1)+" : "+str(t)) return t</pre>	Fungsi untuk mencari titik tengah dari 2 buah titik
2	<pre>def berzier_curve_kuad(P0,P1,P2) : # t = [0,1] Q0 = get_mid(P0,P1) Q1 = get_mid(P1,P2) R0 = get_mid(Q0,Q1)</pre>	Fungsi untuk mencari titik tengah dari titik tengah

	<pre>return R0</pre>	
3	<pre>def div_conquer(t0: list,t1 : list,t2 : list,n : int,x : list,y : list): if n <= 1: titik = berzier_curve_kuad(t0,t1,t2) x.append(titik[0]) y.append(titik[1]) else: x,y = div_conquer(t0,get_mid(t0,t1),berzier_curve_kuad(t0,t 1,t2),n/2,x,y) x,y = div_conquer(berzier_curve_kuad(t0,t1,t2,),get_mid(t1, t2),t2,n/2,x,y) return x,y</pre>	<p>fungsi algoritma utama. menerima 3 buah titik, banyak iterasi dan list x dan y. Apabila banyak iterasi sisa <= 1 (basis) menghasilkan titik menggunakan fungsi berzier_curve_kuad, selain itu rekursif dan n dibagi 2</p>

BAB 4

Source Code

4.1. main.py

File utama untuk menjalankan program (mengimport time, numpy, brute_bezier.py, dnc_bezier.py, dan matplotlib)

No	Kode	Deskripsi
1	<pre>def main(): algorithm = main_menu() t0 = str_to_float_list(input("titik awal (x y): ").split()) t1 = str_to_float_list(input("titik kontrol (x y): ").split()) t2 = str_to_float_list(input("titik akhir (x y): ").split()) n = int(input("jumlah iterasi : ")) start = time.time() if algorithm == 1: x,y = bb.bruteforce(t0,t1,t2,n+1) else: x,y = dnc.div_conquer(t0,t1,t2,n,[t0[0]], [t0[1]]) runtime = time.time()-start print("runtime : " + str(runtime)) show_curve(x,y,t0,t1,t2) return 0</pre>	fungsi utama untuk menjalankan program, tempat menginput titik dan banyak iterasi.
2	<pre>def main_menu(): print("Pilih algoritma yang ingin digunakan") print("1). Brute Force") print("2). Divide and Conquer")</pre>	fungsi menu utama, tempat memilih algoritma untuk membuat kurva bezier

	<pre> mode = input("(1/2) : ") while(mode != "1" and mode != "2"): print("Pilih algoritma yang ingin digunakan") print("1). Brute Force") print("2). Divide and Conquer") mode = input("(1/2) : ") return mode </pre>	
3	<pre> def str_to_float_list(l): for i in range(len(l)) : l[i] = float(l[i]) return l </pre>	fungsi untuk mengubah str menjadi float dalam sebuah list
4	<pre> def show_curve(x,y,t0,t1,t2): plt.plot(x, y, label = "bezier curve") x1 = [t0[0],t1[0],t2[0]] y1 = [t0[1],t1[0],t2[0]] plt.scatter(x1,y1, color='red') if len(x) < 100: plt.scatter(x, y, color='red') plt.plot([t0[0], t1[0], t2[0]], [t0[1], t1[1], t2[1]], 'k--', label='Control Points') plt.legend() plt.show() </pre>	fungsi untuk menampilkan kurva bezier berdasarkan list x dan y menggunakan matplotlib

4.2. brute_bezier.py

No	Kode	Deskripsi
----	------	-----------

1	<pre>def bruteforce(t0 :list ,t1 : list,t2 : list,n : int): x = [t0[0]] y = [t0[1]] for i in range (1,n+1): t = i/n b0 = (1 - t) ** 2 b1 = 2*t*(1 - t) b2 = t ** 2 x.append(t0[0] * b0 + t1[0] * b1 + t2[0] * b2) y.append(t0[1] * b0 + t1[1] * b1 + t2[1] * b2) return x,y</pre>	<p>algoritma ini menerima 3 titik (t0,t1,t2) dan banyak iterasi (n) dan menghasilkan list x,y (koordinat dalam sumbu x dan y) yang awalnya dimasukkan titik awal (t0). Dalam loop sebanyak jumlah iterasi ditentukan t (fase) lalu memasukkannya ke dalam rumus dan didapatkan koordinat x dan y nya yang lalu ditambahkan ke dalam list x dan y.</p>
---	--	---

4.3. dnc_bezier.py

No	Kode	Deskripsi
1	<pre>def get_mid(t0,t1): t = [] t.append((t0[0]+t1[0])/2) t.append((t0[1]+t1[1])/2) #print("mid "+str(t0)+" , "+str(t1)+" : "+str(t)) return t</pre>	Fungsi untuk mencari titik tengah dari 2 buah titik
2	<pre>def berzier_curve_kuad(P0,P1,P2) : # t = [0,1] Q0 = get_mid(P0,P1) Q1 = get_mid(P1,P2) R0 = get_mid(Q0,Q1) return R0</pre>	Fungsi untuk mencari titik tengah dari titik tengah
3	<pre>def div_conquer(t0: list,t1 : list,t2 : list,n : int,x : list,y : list):</pre>	fungsi algoritma utama. menerima 3 buah titik, banyak iterasi dan list x dan

```

if n <= 1:
    titik = berzier_curve_kuad(t0,t1,t2)
    x.append(titik[0])
    y.append(titik[1])
else:
    x,y =
div_conquer(t0,get_mid(t0,t1),berzier_curve_kuad(t0,t
1,t2),n/2,x,y)
    x,y =
div_conquer(berzier_curve_kuad(t0,t1,t2,),get_mid(t1,
t2),t2,n/2,x,y)

return x,y

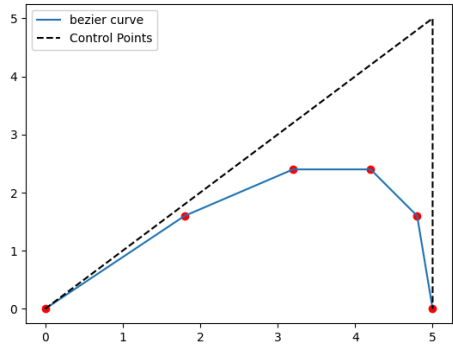
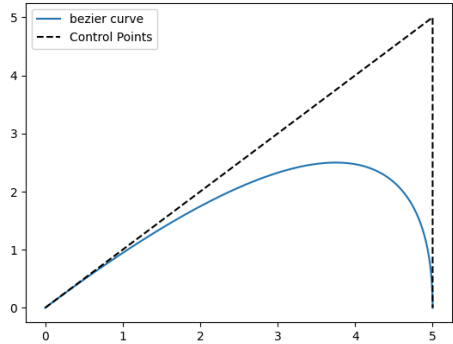
```

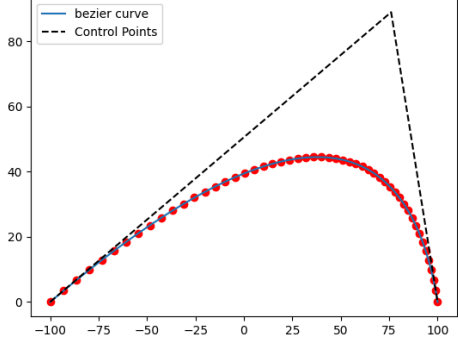
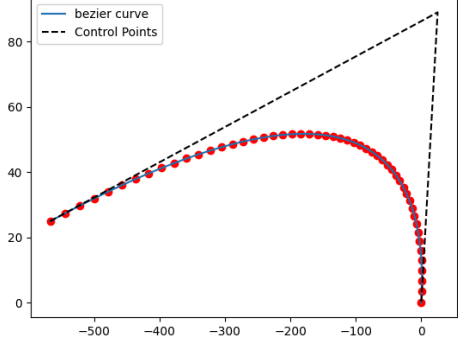
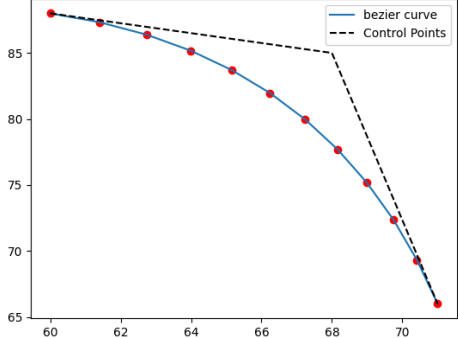
y. Apabila banyak iterasi
 sisa ≤ 1 (basis)
 menghasilkan titik
 menggunakan fungsi
 berzier_curve_kuad, selain
 itu rekursif dan n dibagi 2

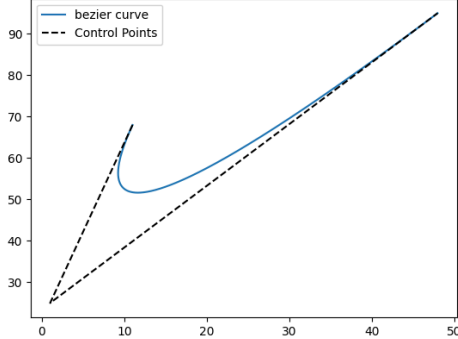
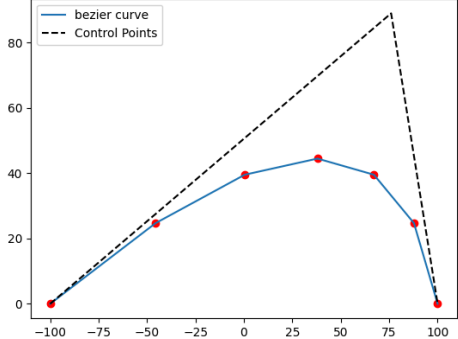
BAB 5

Test Case

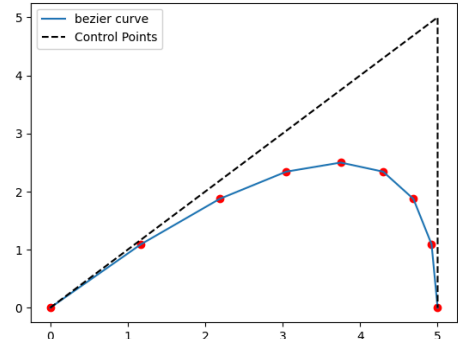
5.1. Brute Force

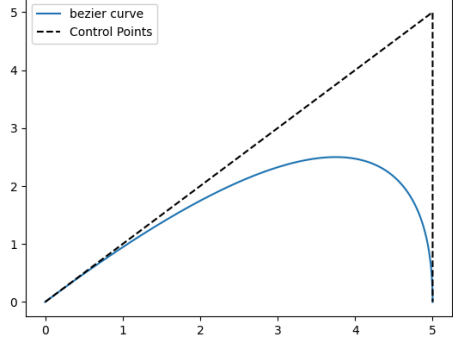
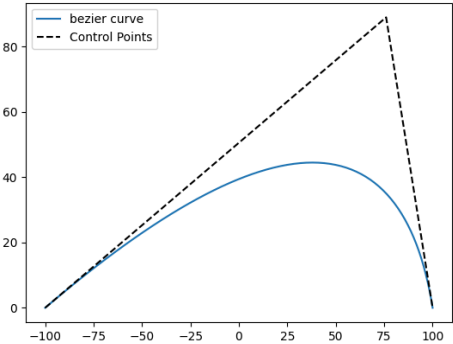
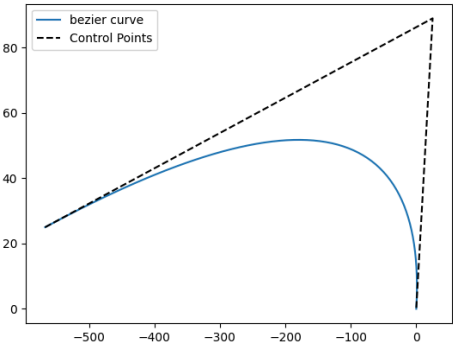
No	Input	Output	Runtime
1	<pre> Pilih algoritma yang ingin digunakan 1). Brute Force 2). Divide and Conquer (1/2) : 1 titik awal (x y): 0 0 titik kontrol (x y): 5 5 titik akhir (x y): 5 0 jumlah iterasi : 4 runtime : 0.0 ms </pre>		0.0 ms
2	<pre> Pilih algoritma yang ingin digunakan 1). Brute Force 2). Divide and Conquer (1/2) : 1 titik awal (x y): 0 0 titik kontrol (x y): 5 5 titik akhir (x y): 5 0 jumlah iterasi : 10000 runtime : 7.730007171630859 ms </pre>		7.7 ms

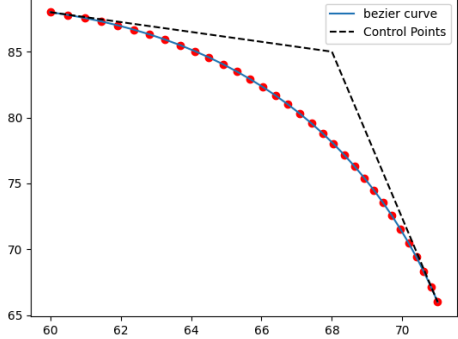
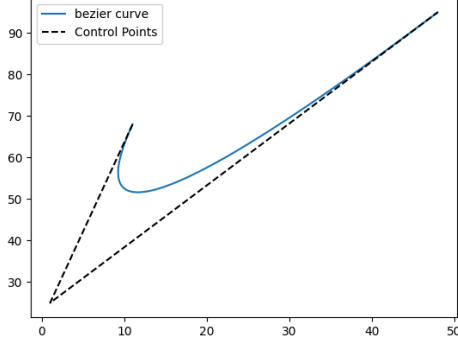
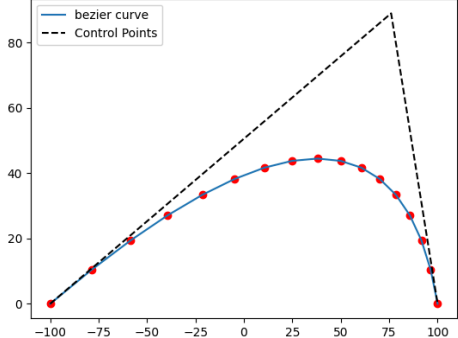
3	<p>Pilih algoritma yang ingin digunakan</p> <p>1). Brute Force</p> <p>2). Divide and Conquer</p> <p>(1/2) : 1</p> <p>titik awal (x y): -100 0</p> <p>titik kontrol (x y): 76 89</p> <p>titik akhir (x y): 100 0</p> <p>jumlah iterasi : 50</p> <p>runtime : 0.0 ms</p>		0 ms
4	<p>Pilih algoritma yang ingin digunakan</p> <p>1). Brute Force</p> <p>2). Divide and Conquer</p> <p>(1/2) : 1</p> <p>titik awal (x y): -567 25</p> <p>titik kontrol (x y): 25 89</p> <p>titik akhir (x y): 0 0</p> <p>jumlah iterasi : 50</p> <p>runtime : 0.0 ms</p>		0 ms
5	<p>Pilih algoritma yang ingin digunakan</p> <p>1). Brute Force</p> <p>2). Divide and Conquer</p> <p>(1/2) : 1</p> <p>titik awal (x y): 60 88</p> <p>titik kontrol (x y): 68 85</p> <p>titik akhir (x y): 71 66</p> <p>jumlah iterasi : 10</p> <p>runtime : 0.0 ms</p>		0 ms

6	<p>Pilih algoritma yang ingin digunakan</p> <p>1). Brute Force</p> <p>2). Divide and Conquer</p> <p>(1/2) : 1</p> <p>titik awal (x y): 11 68</p> <p>titik kontrol (x y): 1 25</p> <p>titik akhir (x y): 48 95</p> <p>jumlah iterasi : 100000</p> <p>runtime : 39.52312469482422 ms</p>		39.5 ms
7	<p>Pilih algoritma yang ingin digunakan</p> <p>1). Brute Force</p> <p>2). Divide and Conquer</p> <p>(1/2) : 1</p> <p>titik awal (x y): -100 0</p> <p>titik kontrol (x y): 76 89</p> <p>titik akhir (x y): 100 0</p> <p>jumlah iterasi : 5</p> <p>runtime : 0.0 ms</p>		0 ms

5.2. Divide and Conquer

No	Input	Output	Runtime
1	<p>Pilih algoritma yang ingin digunakan</p> <p>1). Brute Force</p> <p>2). Divide and Conquer</p> <p>(1/2) : 2</p> <p>titik awal (x y): 0 0</p> <p>titik kontrol (x y): 5 5</p> <p>titik akhir (x y): 5 0</p> <p>jumlah iterasi : 4</p> <p>runtime : 0.0 ms</p>		0.0 ms

2	<p>Pilih algoritma yang ingin digunakan</p> <p>1). Brute Force</p> <p>2). Divide and Conquer</p> <p>(1/2) : 2</p> <p>titik awal (x y): 0 0</p> <p>titik kontrol (x y): 5 5</p> <p>titik akhir (x y): 5 0</p> <p>jumlah iterasi : 10000</p> <p>runtime : 31.543254852294922 ms</p>		31.5 ms
3	<p>Pilih algoritma yang ingin digunakan</p> <p>1). Brute Force</p> <p>2). Divide and Conquer</p> <p>(1/2) : 2</p> <p>titik awal (x y): -100 0</p> <p>titik kontrol (x y): 76 89</p> <p>titik akhir (x y): 100 0</p> <p>jumlah iterasi : 50</p> <p>runtime : 0.0 ms</p>		0 ms
4	<p>Pilih algoritma yang ingin digunakan</p> <p>1). Brute Force</p> <p>2). Divide and Conquer</p> <p>(1/2) : 2</p> <p>titik awal (x y): -567 25</p> <p>titik kontrol (x y): 25 89</p> <p>titik akhir (x y): 0 0</p> <p>jumlah iterasi : 50</p> <p>runtime : 1.5065670013427734 ms</p>		1.5 ms

5	Pilih algoritma yang ingin digunakan 1). Brute Force 2). Divide and Conquer (1/2) : 2 titik awal (x y): 60 88 titik kontrol (x y): 68 85 titik akhir (x y): 71 66 jumlah iterasi : 10 runtime : 0.0 ms		0 ms
6	Pilih algoritma yang ingin digunakan 1). Brute Force 2). Divide and Conquer (1/2) : 2 titik awal (x y): 11 68 titik kontrol (x y): 1 25 titik akhir (x y): 48 95 jumlah iterasi : 100000 runtime : 226.2744903564453 ms		226.2 ms
7	Pilih algoritma yang ingin digunakan 1). Brute Force 2). Divide and Conquer (1/2) : 2 titik awal (x y): -100 0 titik kontrol (x y): 76 89 titik akhir (x y): 100 0 jumlah iterasi : 5 runtime : 0.0 ms		0 ms

5.3. Analisis

Berdasarkan hasil uji coba dari kedua algoritma dapat dilihat bahwa waktu yang diperlukan tidak beda jauh. Namun saat nilai iterasi tinggi algoritma brute force lebih cepat selesai. Hal ini dapat dilihat pada test case nomor 2,4, dan 6. Semakin tinggi nilai iterasi semakin jauh perbedaan antara waktu jalan algoritma brute force dan algoritma divide and conquer. Namun apabila dihitung kompleksitas dari kedua algoritma

adalah $O(n)$ dengan n adalah banyaknya iterasi. Hal tersebut terjadi karena faktor lain seperti memori alokasi dan pemanggilan fungsi lain.

Selain dari Perbedaan kecepatan menjalankan program, terdapat juga perbedaan kualitas kurva dalam iterasi rendah. Dari hasil uji coba nomor 1 dan 7 dapat dilihat pada iterasi rendah algoritma divide and conquer sudah terlihat lebih seperti sebuah kurva dibandingkan dengan algoritma brute force. Hal ini dikarenakan pada setiap iterasi, algoritma divide and conquer menghasilkan $2n-1$ titik dimana n adalah banyak iterasi sementara itu algoritma brute force hanya menghasilkan n buah titik.

Maka dari itu salah satu faktor lainnya dari mengapa algoritma divide and conquer berjalan lebih lama daripada algoritma brute force adalah banyaknya titik yang dihasilkan tidak setara, melainkan $n-1$ lebih banyak dibandingkan algoritma brute force.

Lampiran

Repository : [Github](#)

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

Daftar Pustaka

Munir, Rinaldi. "Algoritma Greedy." *Informatika*,
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf). Accessed 8 March 2024.

unila. "Apa itu Bot." *Himatro Unila*, <https://himatro.ee.unila.ac.id/apa-itu-bot/>. Accessed 9 March 2024.