LAPORAN TUGAS KECIL II
IF2211 STRATEGI ALGORITMA
"Membangun Kurva Bézier dengan Algoritma Titik Tengah berbasis Divide and Conquer"

Dosen:

Ir. Rila Mandala, M. Eng, Ph. D.

Monterico Adrian, S. T., M. T.

Kelompok 80:

13522131 Owen Tobias Sinurat

13522155 Axel Santadi Warih

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER II TAHUN 2023/2024

DAFTAR ISI

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa kami ucapkan atas keberhasilan dalam menyelesaikan Tugas Kecil II IF2211 Strategi Algoritma yang berjudul "Membangun Kurva *Bézier* dengan Algoritma Titik Tengah berbasis *Divide and Conquer*" ini.

Laporan ini merupakan dokumentasi dan penjelasan atas program dan algoritma yang telah kami implementasikan. Kami berusaha untuk merancang dan mengimplementasikan strategi *divide and conquer* ini sedemikian rupa sehingga kurva dapat dibentuk dengan lebih efisien.

Pengembangan algoritma dan penyelesaian tugas kecil ini tidak lepas dari bimbingan dan arahan yang diberikan oleh dosen pengampu mata kuliah Strategi Algoritma. Kami juga ingin mengucapkan terima kasih kepada rekan-rekan kami yang telah memberikan masukan, saran, dan inspirasi selama proses pengembangan algoritma ini. Selain itu, kami juga mengapresiasi dukungan yang diberikan oleh asisten dan semua pihak yang telah membantu, baik secara langsung maupun tidak langsung, dalam penyelesaian tugas kecil ini. Kami berharap bahwa *output* dari tugas kecil ini dapat memberikan kontribusi bagi pengembangan ilmu pengetahuan, khususnya dalam bidang strategi algoritma.

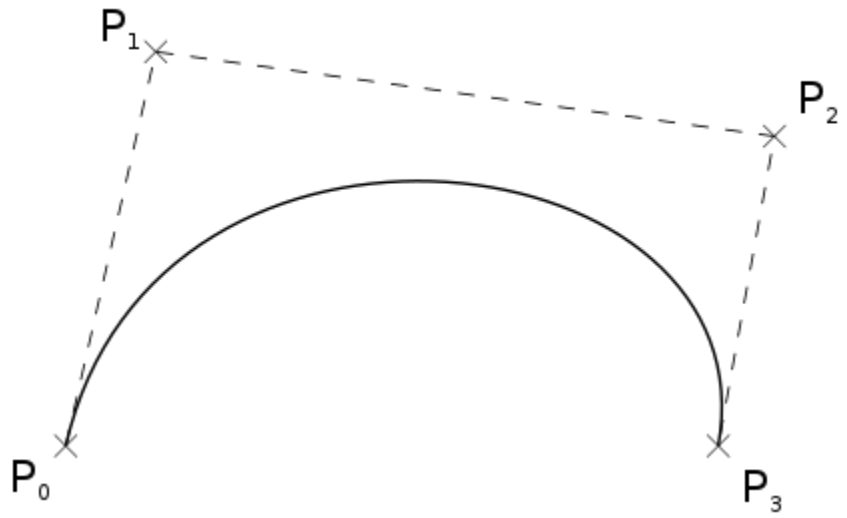
Kami menyadari bahwa tugas kecil ini masih jauh dari sempurna. Oleh karena itu, kami sangat terbuka untuk menerima kritik dan saran yang konstruktif demi perbaikan di masa yang akan datang. Akhir kata, semoga tugas kecil ini dapat bermanfaat bagi semua pihak yang berkepentingan.

Sumedang, 14 Maret 2024,

Kelompok 80.

BAB I
DESKRIPSI TUGAS

1.1. Teori Dasar



(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 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 P0 sampai Pn , 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 P0, sedangkan titik kontrol terakhir adalah P3. Titik kontrol P1 dan P2 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 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.

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

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.

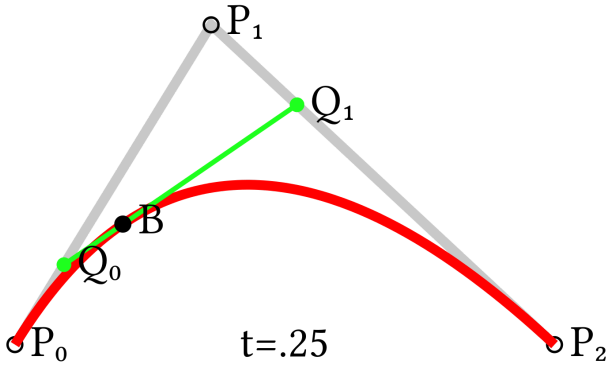
Misalkan selain dua titik sebelumnya ditambahkan sebuah titik baru, sebut saja P2, dengan P0 dan P2 sebagai titik kontrol awal dan akhir, dan P1 menjadi titik kontrol antara. Dengan menyatakan titik Q1 terletak diantara garis yang menghubungkan P1 dan P2, dan membentuk kurva Bézier linier yang berbeda 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.

$$Q_0 = B(t) = (1-t)P_0 + tP_1, \qquad t \in [0,1]$$
$$Q_1 = B(t) = (1-t)P_1 + tP_2, \qquad t \in [0,1]$$
$$R_0 = B(t) = (1-t)Q_0 + tQ_1, \qquad t \in [0,1]$$

dengan melakukan substitusi nilai Q0 dan Q1, maka diperoleh persamaan sebagai berikut.

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

Berikut adalah ilustrasi dari kasus diatas.



Gambar 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.

$$S_0 = B(t) = (1-t)^3P_0 + 3(1-t)^2tP_1 + 3(1-t)t^2P_2 + t^3P_3, \qquad t \in [0,1]$$
$$T_0 = B(t) = (1-t)^4P_0 + 4(1-t)^3tP_1 + 6(1-t)^2t^2P_2 + 4(1-t)t^3P_3 + t^4P_4, \qquad t \in [0,1]$$

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.

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.

1.2. Ilustrasi Kasus

Idenya cukup sederhana, relatif mirip dengan pembahasan sebelumnya, dan dilakukan secara iteratif. Misalkan terdapat tiga buah titik, P0, P1, dan P2, dengan titik P1 menjadi titik kontrol antara, maka:

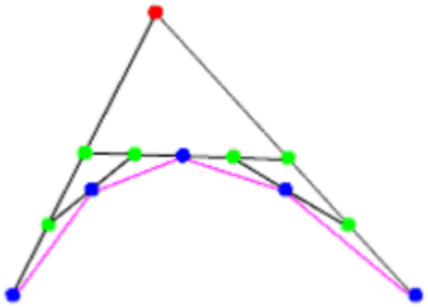
a. Buatlah sebuah titik baru Q0 yang berada di tengah garis yang menghubungkan P0 dan P1, serta titik Q1 yang berada di tengah garis yang menghubungkan P1 dan P2.
b. Hubungkan Q0 dan Q1 sehingga terbentuk sebuah garis baru.
c. Buatlah sebuah titik baru R0 yang berada di tengah Q0 dan Q1.
d. Buatlah sebuah 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.

a. 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.
b. Hubungkan S0 dengan S1 dan S2 dengan S3 sehingga terbentuk garis baru.

c. Buatlah dua buah titik baru, yaitu T0 yang berada di tengah S0 dan S1, serta T1 yang berada di tengah S2 dan S3.
d. 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.



Gambar 3. Hasil pembentukan Kurva Bézier Kuadratik dengan divide and conquer setelah iterasi ke-2.

BAB II
ANALISIS DAN IMPLEMENTASI ALGORITMA

2.1 Analisis dan Implementasi Algoritma *Brute Force*

Algoritma *brute force* pada pembentukan kurva *Bezier* akan mengiterasi setiap titik untuk mendapatkan hasil akhirnya. Hal ini berpengaruh terhadap waktu pemrosesannya yang menjadi cukup lama apabila iterasi yang dilakukan banyak.

Berikut adalah tahapan algoritma *brute force* untuk pembentukan kurva *Bezier*:

1. Diberikan 3 titik dengan salah satunya adalah titik kontrol, dan diberikan juga jumlah iterasi yang ingin dijalani.
2. Titik diproses menjadi titik-titik baru berdasarkan rumus berikut.

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

Contohnya x dan y pada titik P_0 akan dikali dengan $(1\text{-}t)^2$ sehingga akan didapat x dan y yang baru.

2.2 Analisis dan Implementasi Algoritma *Divide and Conquer*

Proses algoritma *divide and conquer* dimulai dengan membagi kurva menjadi segmen-segmen yang lebih kecil, kemudian menggabungkannya secara iteratif untuk mendapatkan kurva Bézier final. Pada algoritma divide and conquer ini, kami menggunakan rumus sebagai berikut.

$$\mathbf{B}(t) = \mathbf{B}_{\mathbf{P}_0\mathbf{P}_1\ldots\mathbf{P}_n}(t) = (1-t)\mathbf{B}_{\mathbf{P}_0\mathbf{P}_1\ldots\mathbf{P}_{n-1}}(t) + t\mathbf{B}_{\mathbf{P}_1\mathbf{P}_2\ldots\mathbf{P}_n}(t)$$

Untuk proses kalkulasinya diilustrasikan sebagai berikut.

A. Iterasi Pertama:

   a. Buat titik-titik tengah, Q0 dan Q1, antara titik-titik kontrol P0-P1 dan P1-P2.

   b. Hubungkan Q0 dan Q1 membentuk garis baru.

   c. Buat titik tengah baru, R0, di antara Q0 dan Q1.

   d. Hubungkan titik-titik kontrol awal dan akhir, P0 - R0 - P2.

B. Iterasi Lanjutan:

   a. Buat titik-titik tengah tambahan, S0, S1, S2, dan S3, antara titik-titik yang telah dibuat pada iterasi sebelumnya.

   b. Hubungkan titik-titik tengah yang baru terbentuk untuk membentuk garis baru.

   c. Buat titik-titik tengah baru, T0 dan T1, antara titik-titik yang baru terbentuk.

   d. Hubungkan semua titik untuk membentuk kurva Bézier baru.

2.3 *Source Code* Algoritma *Brute Force*

bruteforce.py

```
import math


def bezier_bf(positions, t):
    x = 0
    y = 0
    n = len(positions) - 1
    for i in range(len(positions)):
        coeff = math.comb(n, i) * ((1 - t) ** (n - i)) *
(t**i)
        x += coeff * positions[i][0]
```

```
        y += coeff * positions[i][1]
    return [x, y]
```

2.4 *Source Code* Algoritma *Divide and Conquer*

divideandconquer.py

```
def bezier_dnc(points, t):
    if len(points) == 1:
        return points[0]
    else:
        left_points = [point for point in points[:-1]]
        right_points = [point for point in points[1:]]
         return (1 - t) * bezier_dnc(left_points, t) + t
* bezier_dnc(right_points, t)
```

2.5 Implementasi Bonus

A. Bonus 1

Kami mengimplementasikan bonus pertama generalisasi algoritma untuk menerima n buah titik kontrol. Generalisasi algoritma kami lakukan dengan mengganti rumus kurva Bezier kuadratik dengan rumus general kurva Bezier, yaitu:

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

untuk algoritma *brute force*, dan

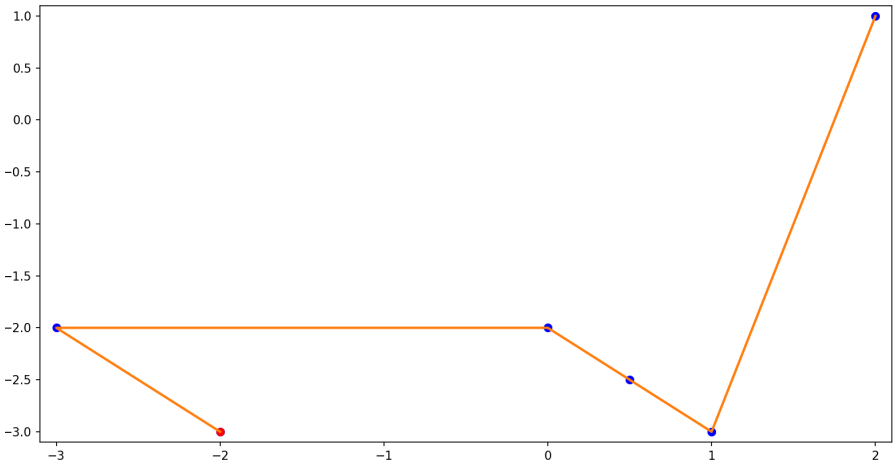
$$\mathbf{B}(t) = \mathbf{B}_{\mathbf{P}_0\mathbf{P}_1\ldots\mathbf{P}_n}(t) = (1-t)\mathbf{B}_{\mathbf{P}_0\mathbf{P}_1\ldots\mathbf{P}_{n-1}}(t) + t\mathbf{B}_{\mathbf{P}_1\mathbf{P}_2\ldots\mathbf{P}_n}(t)$$

untuk algoritma *divide and conquer.*

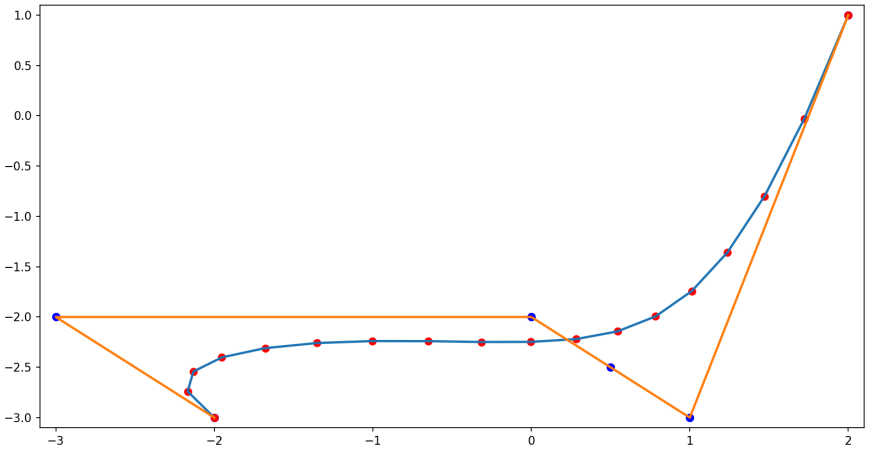
B. Bonus 2

Kami mengimplementasikan bonus kedua yaitu visualisasi proses pembentukan kurva. Proses visualisasi pembentukan dibagi menjadi beberapa tahapan:

1. Digambarkan titik kontrol dan garis penghubungnya sesuai urutan masukan titik kontrol.



2. Kurva digambarkan secara bertahap dengan *increment* 1 titik.



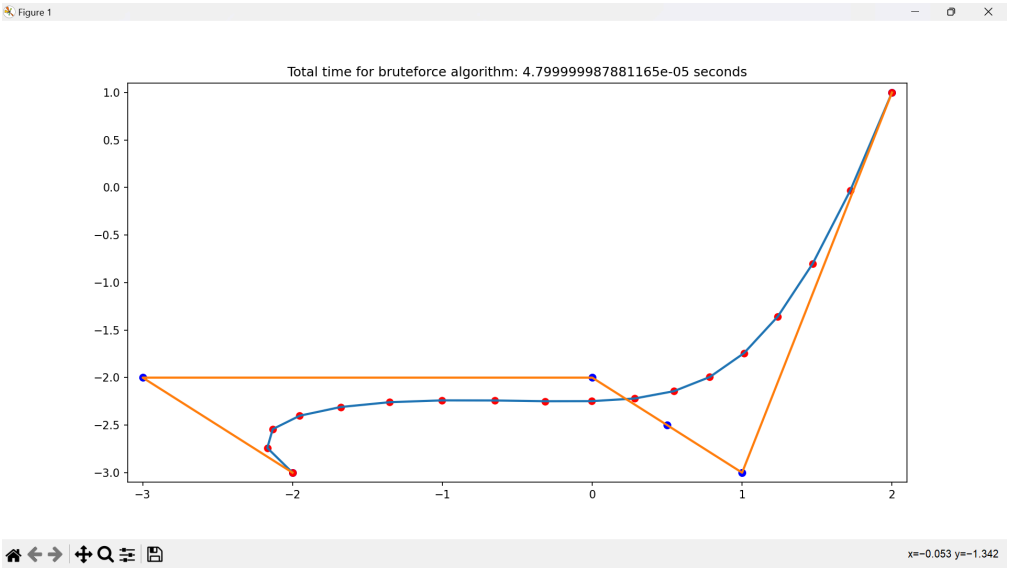
BAB III
PENGUJIAN DAN HASIL

3.1 Pengujian Algoritma Brute Force

1. Pengujian 1

   Titik kontrol : (-2, -3), (-3, -2), (0, -2), (0.5, -2.5), (1, -3), (2, 1).

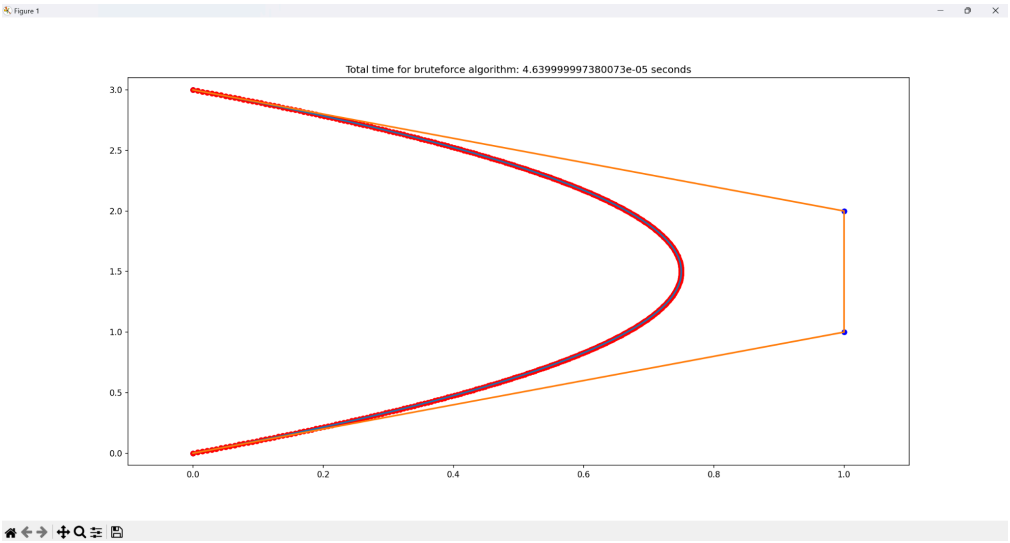
   Waktu : $4.79999\ x\ 10^{-5}$ detik



2. Pengujian 2

   Titik kontrol : (0,0), (1,1), (1,2), (0,3)

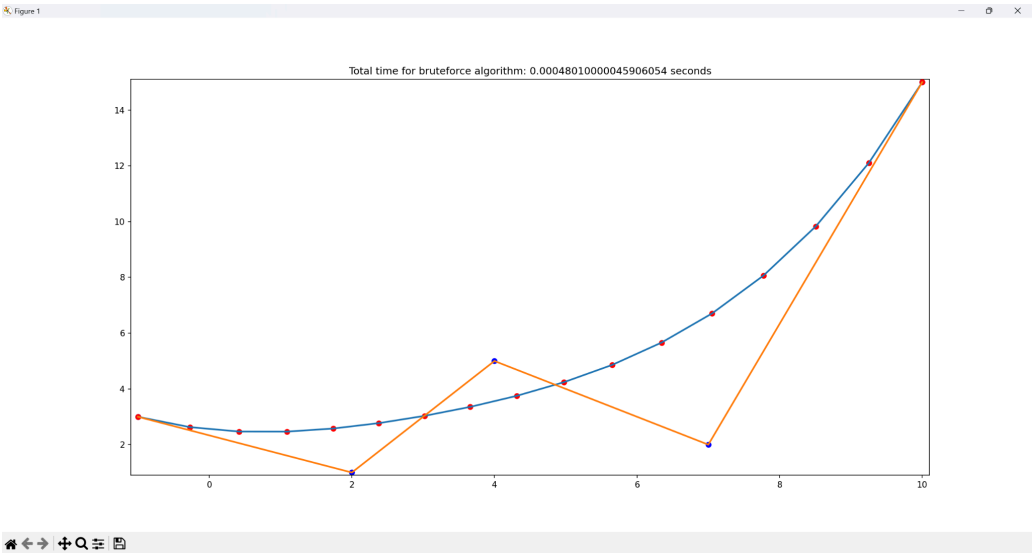
   Waktu : $4.639999 \; x \; 10^{-5}$ detik



3. Pengujian 3

   Titik kontrol : (-1,3), (2,1), (4,5), (7,2), (10,15)

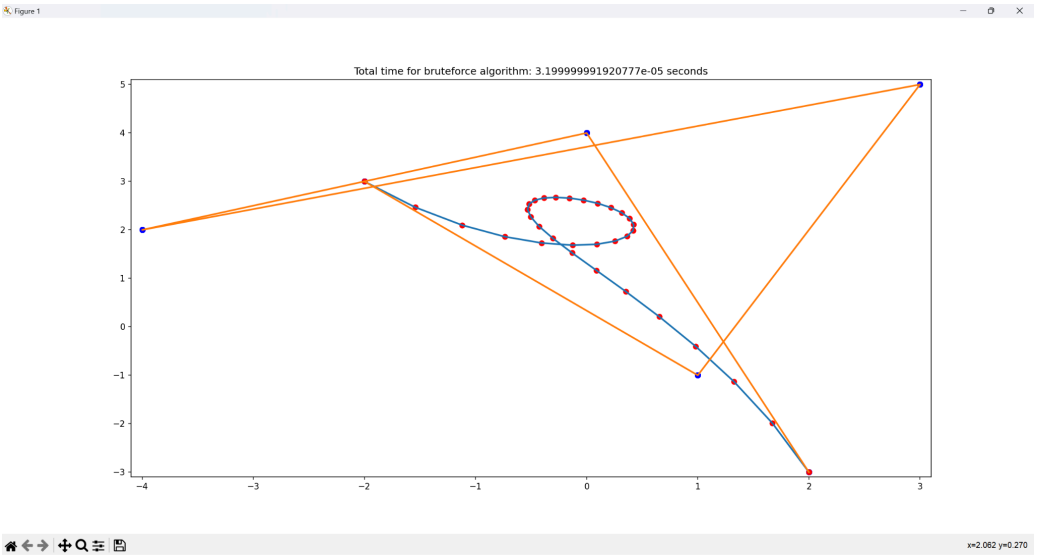
   Waktu : 0.00048 detik



4. Pengujian 4

   Titik Kontrol : (-2,3), (1,-1), (3,5), (-4,2), (0,4), (2,-3)

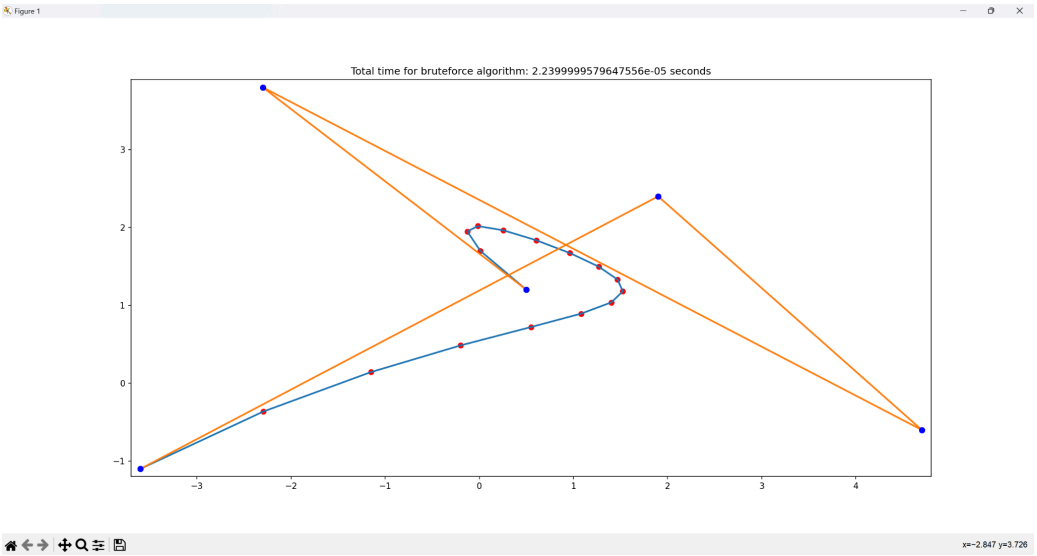
   Waktu : 3.199999 x 10^{-5} detik



5. Pengujian 5

   Titik Kontrol : (0.5, 1.2), (-2.3, 3.8), (4.7, -0.6), (1.9, 2.4), (-3.6, -1.1)

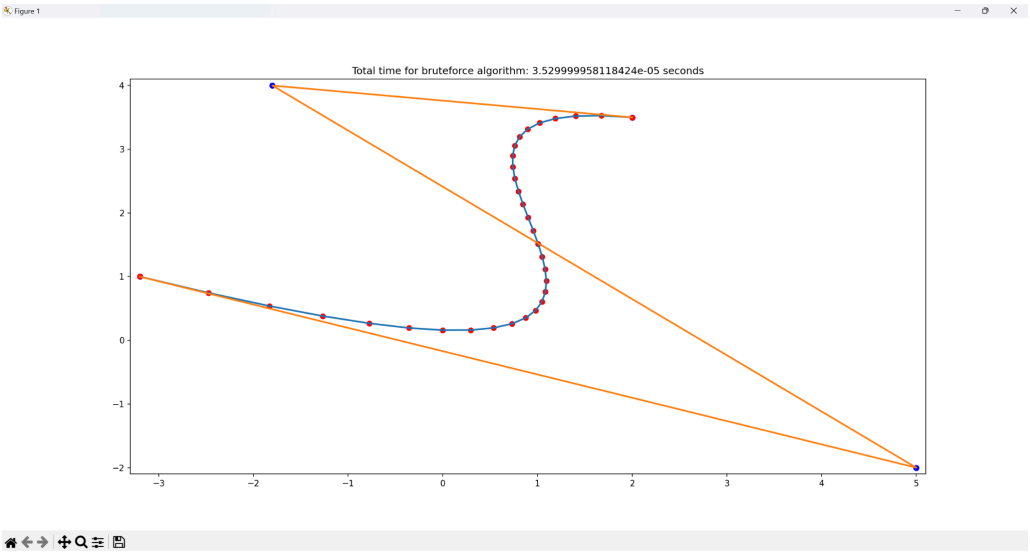
   Waktu : 2.239999 x 10^{-5} detik



6. Pengujian 6

   Titik Kontrol : (2, 3.5), (-1.8, 4), (5, -2), (-3.2, 1)

   Waktu : $3.529999\ x\ 10^{-5}$ detik

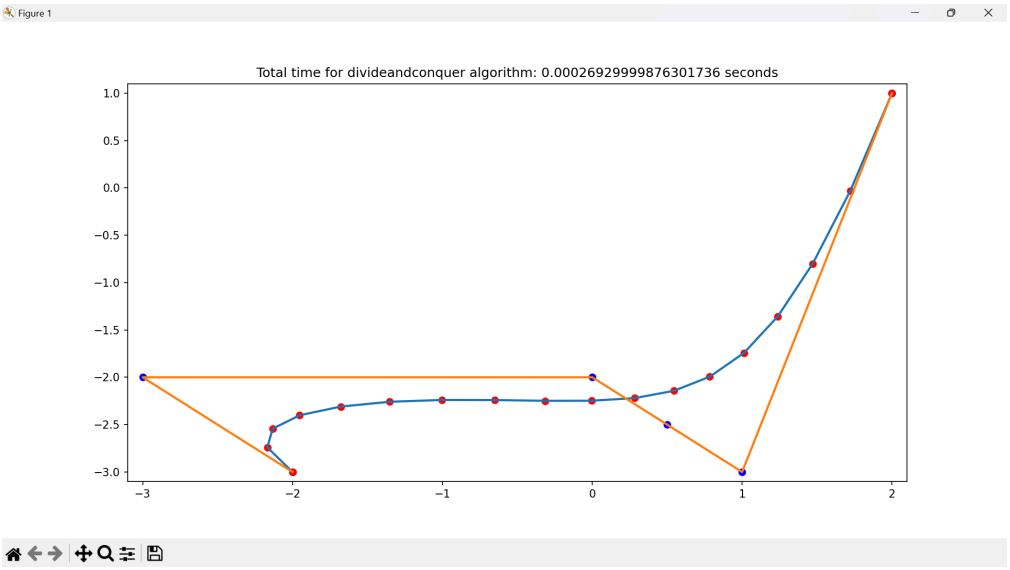


3.2 Pengujian Algoritma Divide and Conquer

1. Pengujian 1

   Titik kontrol : (-2, -3), (-3, -2), (0, -2), (0.5, -2.5), (1, -3), (2, 1).

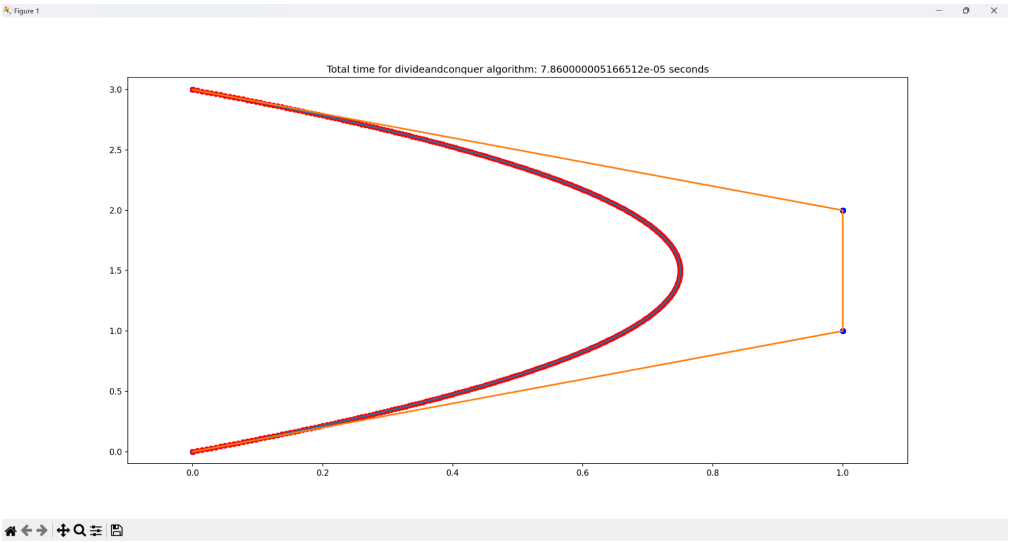
   Waktu : 0.000269 detik



2. Pengujian 2

   Titik kontrol : (0,0), (1,1), (1,2), (0,3)

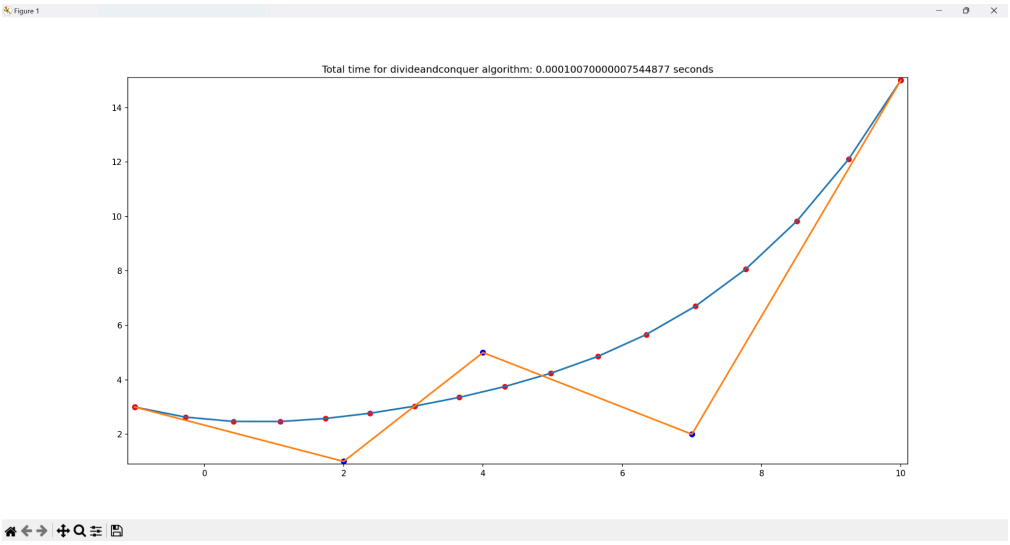
   Waktu : $7.86\ x\ 10^{-5}$ detik



3. Pengujian 3

   Titik kontrol : (-1,3), (2,1), (4,5), (7,2), (10,15)

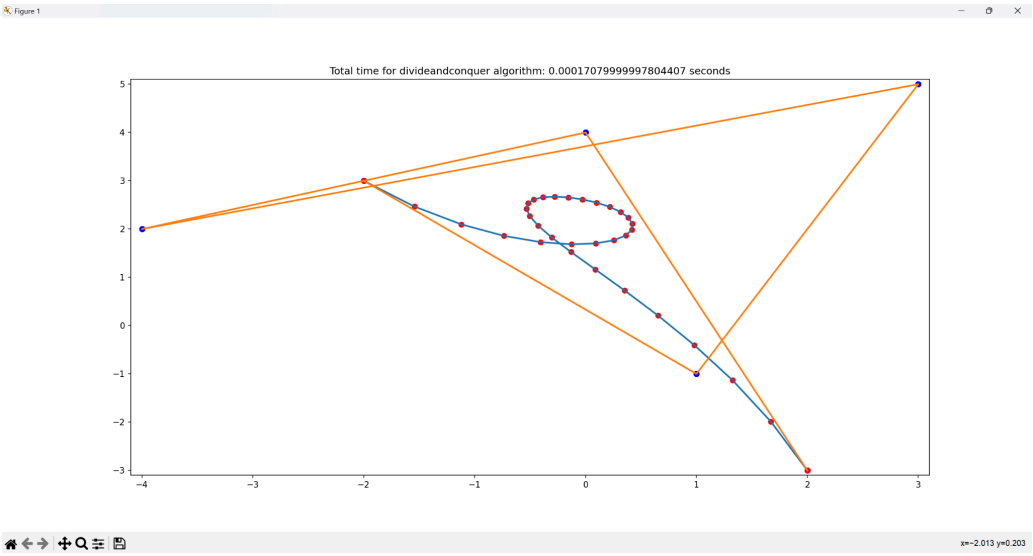
   Waktu : 0.0001 detik



4. Pengujian 4

   Titik Kontrol : (-2,3), (1,-1), (3,5), (-4,2), (0,4), (2,-3)

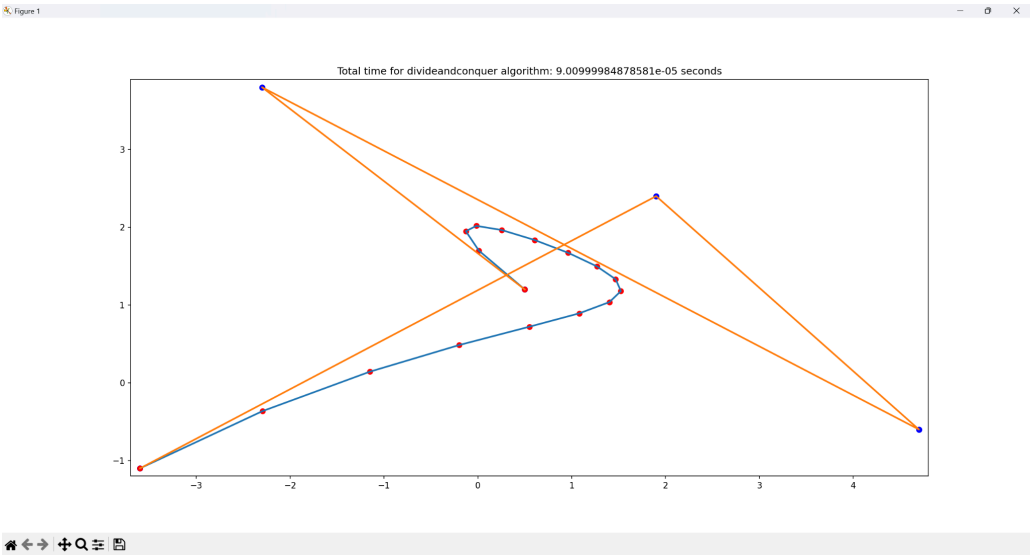
   Waktu : 0.000171 detik



5. Pengujian 5

   Titik Kontrol : (0.5, 1.2), (-2.3, 3.8), (4.7, -0.6), (1.9, 2.4), (-3.6, -1.1)

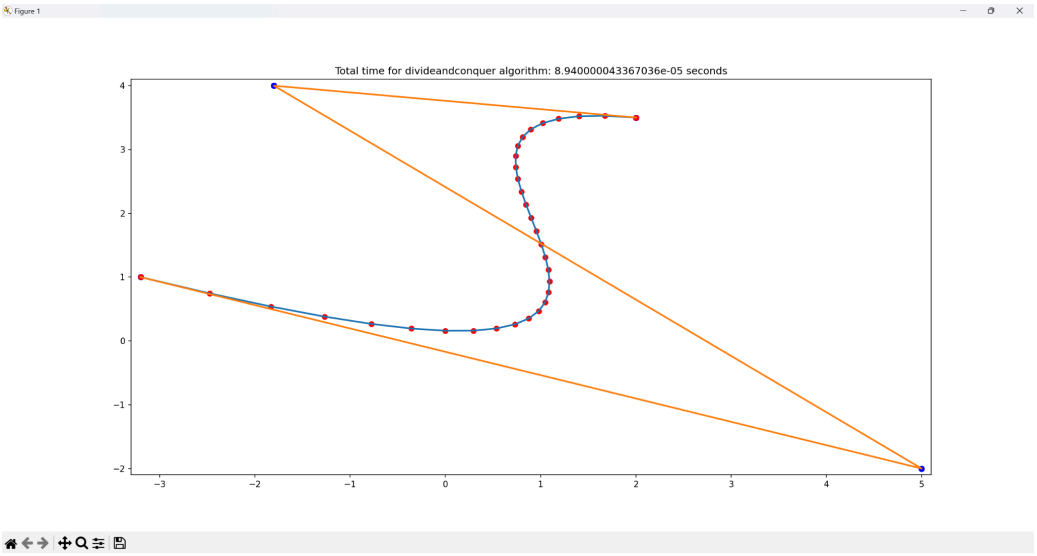
   Waktu : $9.009999\ x\ 10^{-5}$ detik



6. Pengujian 6

   Titik Kontrol : (2, 3.5), (-1.8, 4), (5, -2), (-3.2, 1)

   Waktu : $8.94\ x\ 10^{-5}$ detik



3.3 Analisis Perbandingan Hasil Pengujian

Dari kedua pengujian yang sudah dilakukan, dapat dilihat bahwa kecepatan algoritma *divide and conquer* lebih tinggi …% dibanding algoritma *brute force*. Dan pada kasus pembentukan kurva *Bezier* ini, keakuratan yang sangat detail tidaklah diperlukan. Sehingga dapat disimpulkan bahwa, untuk kebanyakan kasus, persoalan pembentukan kurva *Bezier* akan menghasilkan solusi terbaik jika digunakan strategi *divide and conquer.*

BAB IV
PENUTUP

4.1 Kesimpulan

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. Proses pembuatan kurva adalah perhitungan yang sama yang dilakukan berulang kali dengan parameter yang berbeda-beda. Hal ini membuat algoritma pembuatan kurva Bézier menjadi sangat mahal.

Penerapan strategi *divide and conquer* dapat menjadi jawaban terhadap permasalahan ini dengan mengurangi langkah atau iterasi yang perlu dijalani program. Namun, pengurangan iterasi ini dapat berdampak pada hasil akhir dari algoritma pembangun kurva Bézier. Hasil akhir dengan menggunakan strategi *divide and conquer* dapat dipastikan tidak seakurat dengan menggunakan strategi *brute force* karena iterasi yang dilakukan tidak sebanyak itu. Akan tetapi, waktu yang dibutuhkan untuk memproses input dapat menjadi jauh lebih cepat dibandingkan menggunakan strategi *brute force*. Sehingga ada pertukaran antara hasil akhir dan waktu eksekusi.

Setiap strategi memiliki kekurangan dan kelebihan masing-masing. Oleh karena itu tidak ada strategi yang *one-fits-for-all* karena kebutuhan setiap program tidaklah sama, sehingga pasti ada pertimbangan tertentu untuk menggunakan suatu pilihan strategi.

DAFTAR PUSTAKA

GeeksForGeeks. (2024). *Divide and Conquer Algorithms*. Retrieved from GeeksForGeeks:

https://www.geeksforgeeks.org/divide-and-conquer/

Munir, R. (2024). *Algoritma Divide and Conquer*. Retrieved from Homepage Rinaldi

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

LAMPIRAN

6.1 Ketercapaian

Poin	Ya	Tidak
1. Program berhasil dijalankan.	✓	
2. Program dapat melakukan visualisasi kurva Bézier.	✓	
3. Solusi yang diberikan program optimal.	✓	
4. [Bonus] Program dapat membuat kurva untuk n titik kontrol.	✓	
5. [Bonus] Program dapat melakukan visualisasi proses pembuatan kurva.	✓	

6.2 GitHub Repository

https://github.com/owenthe10x/Tucil2_13522131_13522155