

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

DAFTAR ISI	1
KATA PENGANTAR	2
BAB I DESKRIPSI TUGAS	3
1.1. Teori Dasar	3
1.2. Ilustrasi Kasus	6
BAB II ANALISIS DAN IMPLEMENTASI ALGORITMA	8
2.1 Analisis dan Implementasi Algoritma Brute Force	8
2.2 Analisis dan Implementasi Algoritma Divide and Conquer	8
2.3 Source Code Algoritma Brute Force	9
2.4 Source Code Algoritma Divide and Conquer	10
2.5 Implementasi Bonus	10
BAB III PENGUJIAN DAN HASIL	12
3.1 Pengujian Algoritma Brute Force	12
3.2 Pengujian Algoritma Divide and Conquer	15
3.3 Analisis Perbandingan Hasil Pengujian	18
BAB IV PENUTUP	20
4.1 Kesimpulan	20
DAFTAR PUSTAKA	21
LAMPIRAN	22
6.1 Ketercapaian	22
6.2 GitHub Repository	22

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.

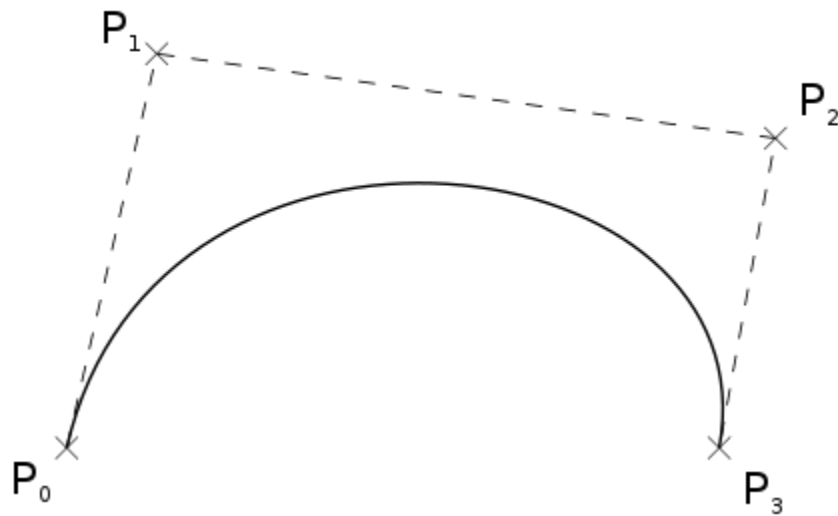
Kami menyadari bahwa tugas kecil ini masih jauh dari sempurna. Oleh karena itu, kami sangat terbuka untuk menerima kritik dan saran yang konstruktif demi perbaikan di masa yang akan datang. Akhir kata, semoga tugas kecil ini dapat bermanfaat bagi semua pihak yang berkepentingan.

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 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 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.

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

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.

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

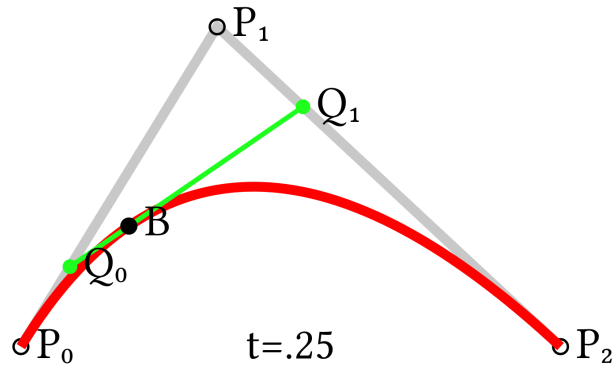
$$Q_1 = B(t) = (1 - t)P_1 + tP_2, \quad t \in [0, 1]$$

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

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

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

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.

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

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

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.

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, P_0 , P_1 , dan P_2 , dengan titik P_1 menjadi titik kontrol antara, maka:

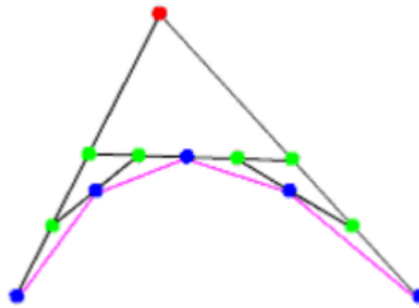
- a. Buatlah sebuah titik baru Q_0 yang berada di tengah garis yang menghubungkan P_0 dan P_1 , serta titik Q_1 yang berada di tengah garis yang menghubungkan P_1 dan P_2 .
- b. Hubungkan Q_0 dan Q_1 sehingga terbentuk sebuah garis baru.
- c. Buatlah sebuah titik baru R_0 yang berada di tengah Q_0 dan Q_1 .
- d. 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.

- a. Buatlah beberapa titik baru, yaitu S_0 yang berada di tengah P_0 dan Q_0 , S_1 yang berada di tengah Q_0 dan R_0 , S_2 yang berada di tengah R_0 dan Q_1 , dan S_3 yang berada di tengah Q_1 dan P_2 .
- b. Hubungkan S_0 dengan S_1 dan S_2 dengan S_3 sehingga terbentuk garis baru.

- c. Buatlah dua buah titik baru, yaitu T_0 yang berada di tengah S_0 dan S_1 , serta T_1 yang berada di tengah S_2 dan S_3 .
- d. Buatlah sebuah garis yang menghubungkan $P_0 - T_0 - R_0 - T_1 - P_2$.

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 semakin banyak.

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

1. Diberikan n titik kontrol dengan dua titik paling ujung menjadi titik kontrol dan titik lainnya menjadi titik kontrol antara. Lalu diberikan juga jumlah iterasi yang akan dijalani.
2. Titik diproses menjadi titik-titik baru berdasarkan rumus berikut.

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

3. Titik-titik tersebut dihubungkan untuk membangun kurva *Bezier*.

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. Pembagian masalah menjadi beberapa sub-masalah inilah yang menjadi inti dari algoritma *divide and conquer*.

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

1. Diberikan n titik kontrol dengan dua titik paling ujung menjadi titik kontrol dan titik lainnya menjadi titik kontrol antara. Lalu diberikan juga jumlah iterasi yang akan dijalani.
2. Kumpulan titik-titik tersebut akan dibagi ke dalam 2 proses terpisah berdasarkan rumus berikut.

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

3. Kalkulasi dilakukan secara rekursif hingga mencapai *base case*.
4. Setelah mencapai *base case*, didapatkan hasil kalkulasi berupa titik-titik baru.
5. Titik-titik baru tersebut dihubungkan untuk membangun kurva *Bezier*.

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

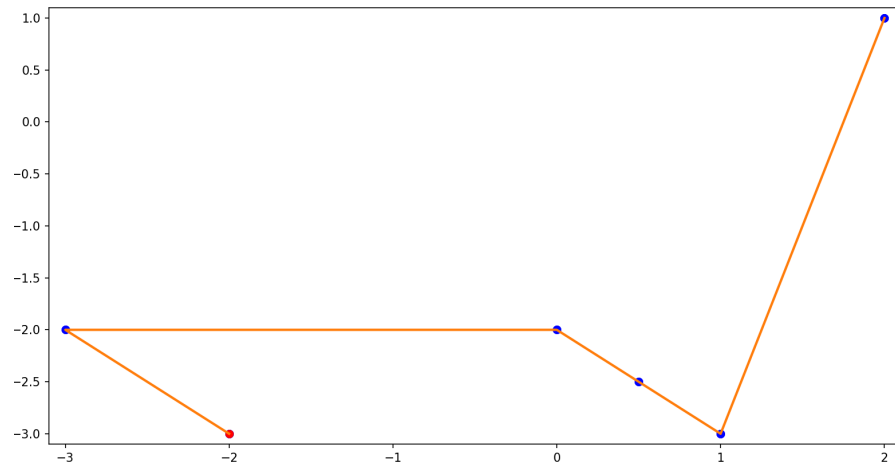
$$\mathbf{B}(t) = \mathbf{B}_{\mathbf{P}_0 \mathbf{P}_1 \dots \mathbf{P}_n}(t) = (1-t) \mathbf{B}_{\mathbf{P}_0 \mathbf{P}_1 \dots \mathbf{P}_{n-1}}(t) + t \mathbf{B}_{\mathbf{P}_1 \mathbf{P}_2 \dots \mathbf{P}_n}(t)$$

untuk algoritma *divide and conquer*.

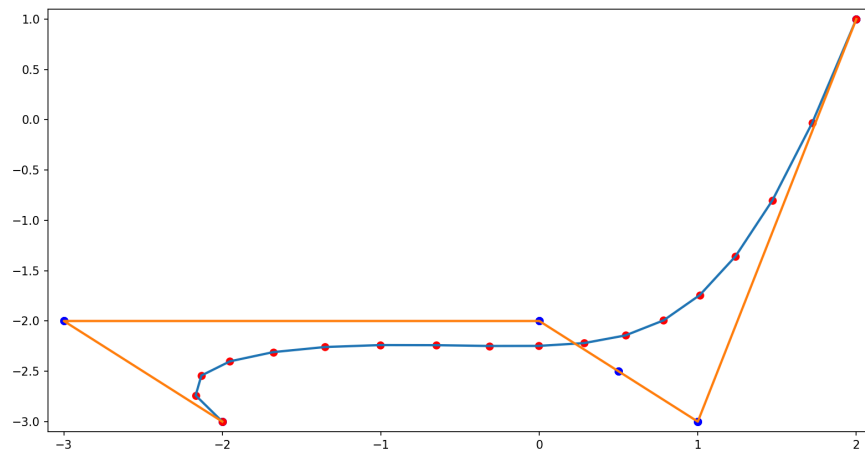
B. Bonus 2

Kami mengimplementasikan bonus kedua yaitu visualisasi proses pembentukan kurva menggunakan *library* matplotlib. 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 Bezier.



BAB III

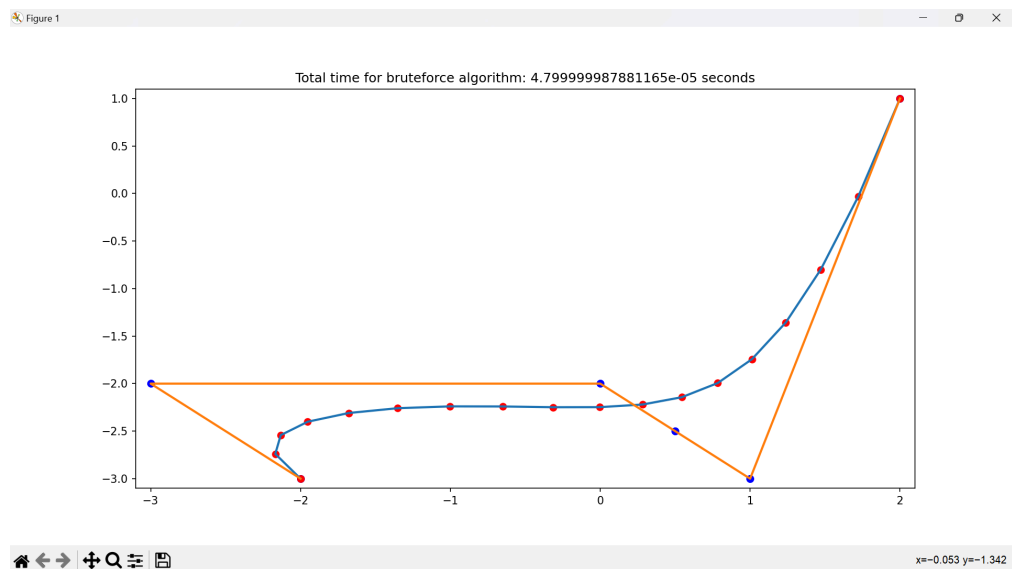
PENGUJIAN DAN HASIL

3.1 Pengujian Algoritma Brute Force

1. Pengujian 1

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

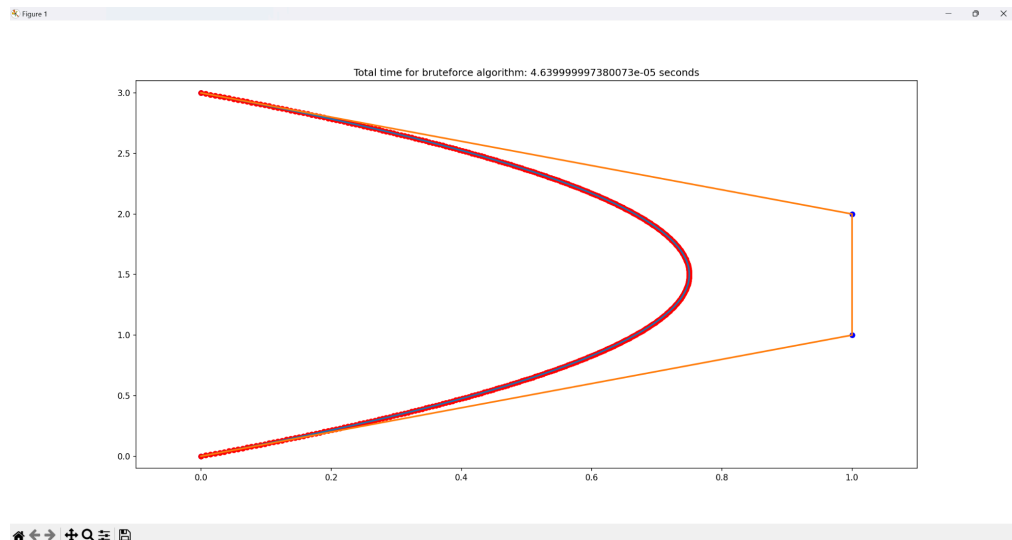
Waktu : 4.79999×10^{-5} detik



2. Pengujian 2

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

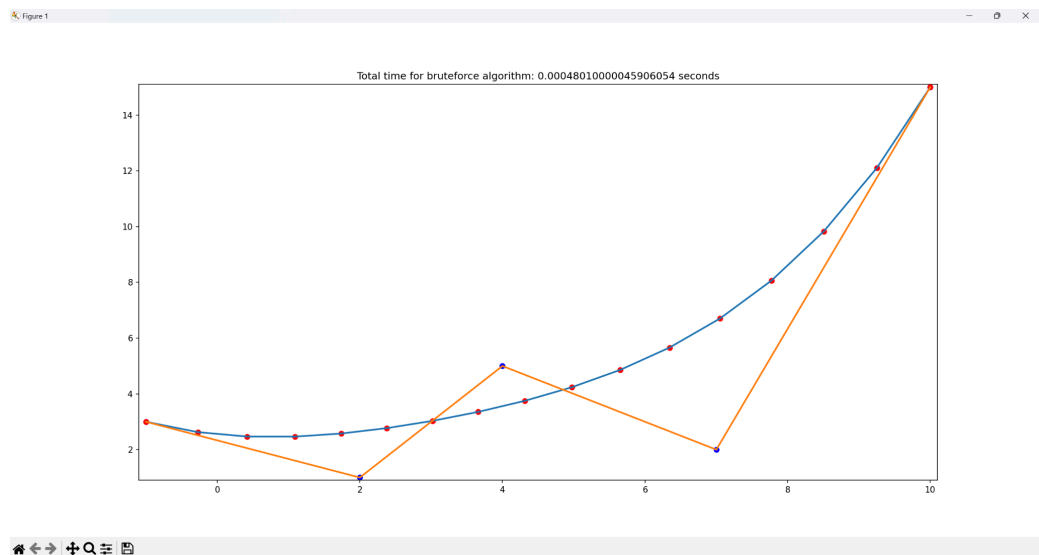
Waktu : 4.639999×10^{-5} detik



3. Pengujian 3

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

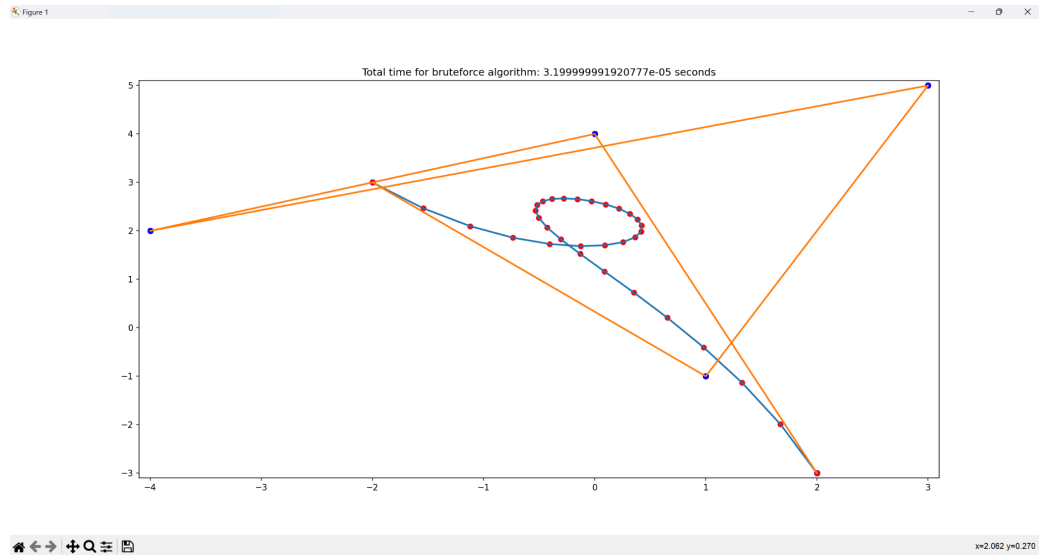
Waktu : 0.00048 detik



4. Pengujian 4

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

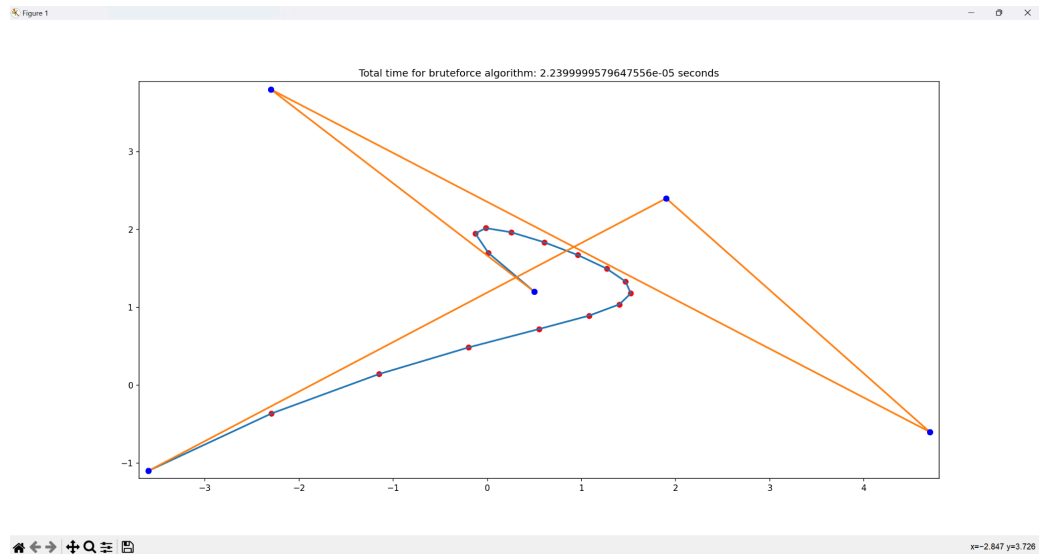
Waktu : 3.199999×10^{-5} detik



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)

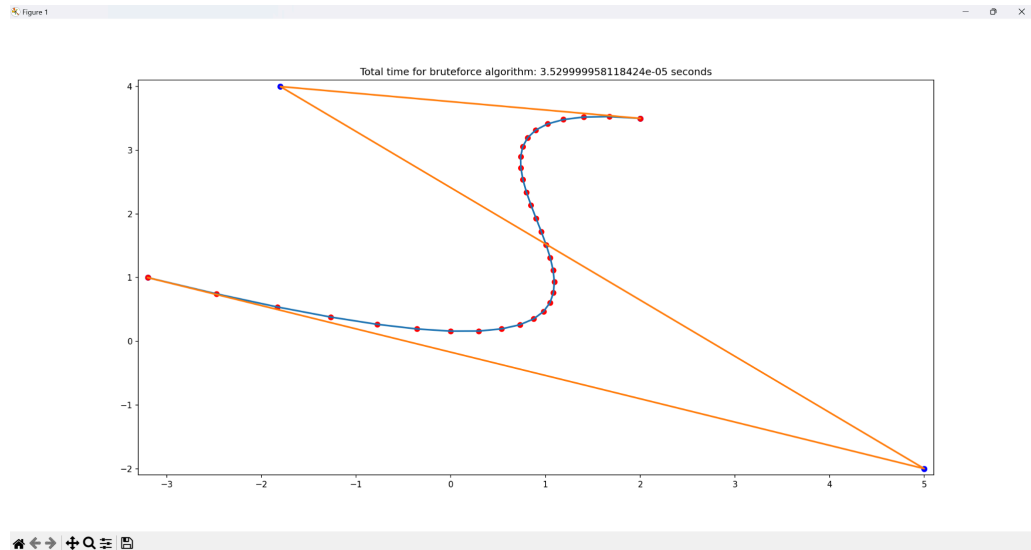
Waktu : 2.239999×10^{-5} detik



6. Pengujian 6

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

Waktu : 3.529999×10^{-5} detik

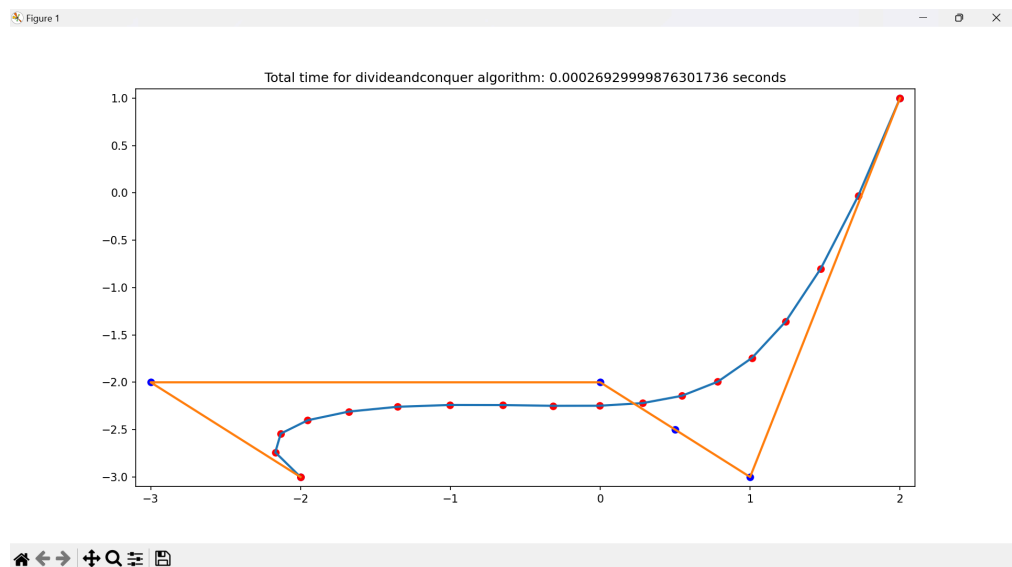


3.2 Pengujian Algoritma Divide and Conquer

1. Pengujian 1

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

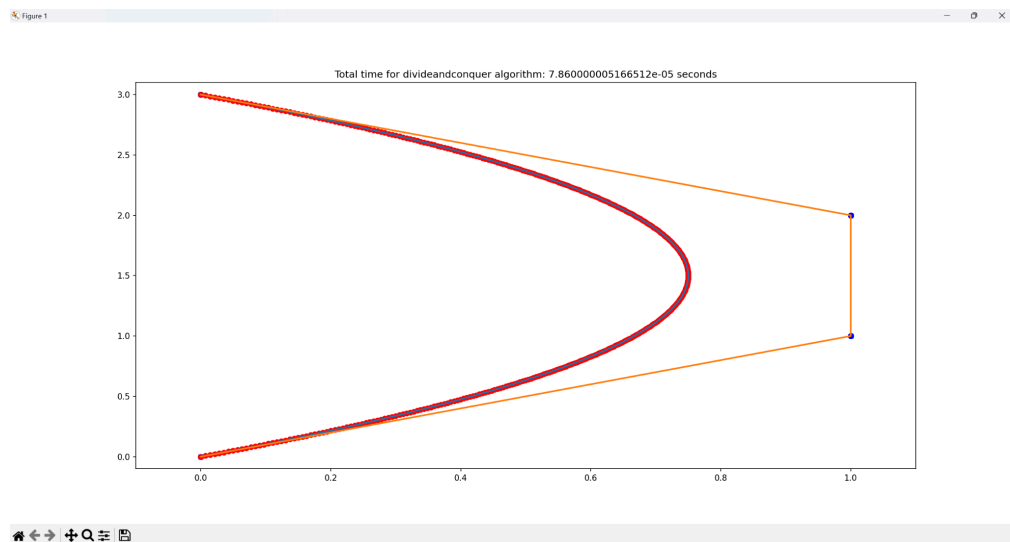
Waktu : 0.000269 detik



2. Pengujian 2

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

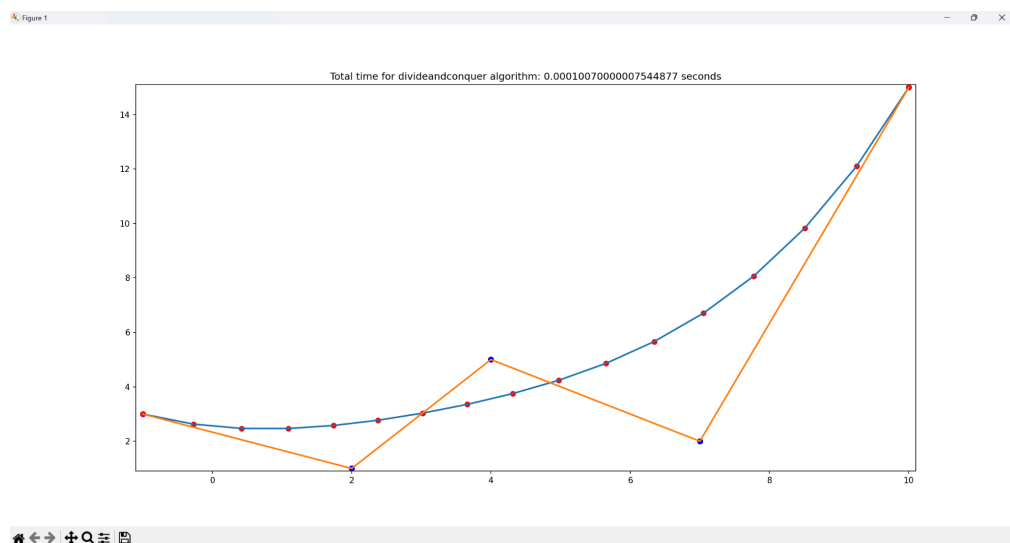
Waktu : 7.86×10^{-5} detik



3. Pengujian 3

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

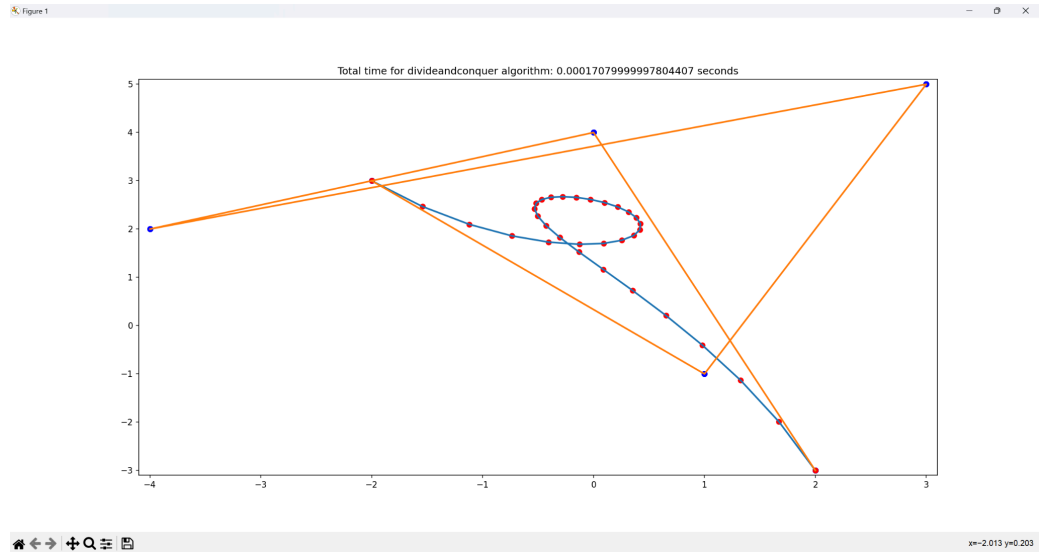
Waktu : 0.0001 detik



4. Pengujian 4

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

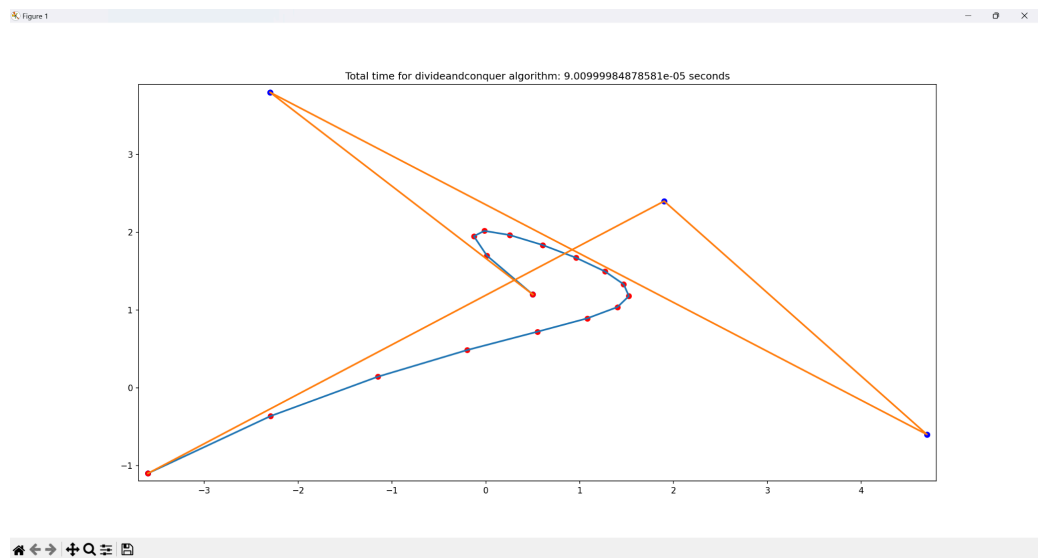
Waktu : 0.000171 detik



5. Pengujian 5

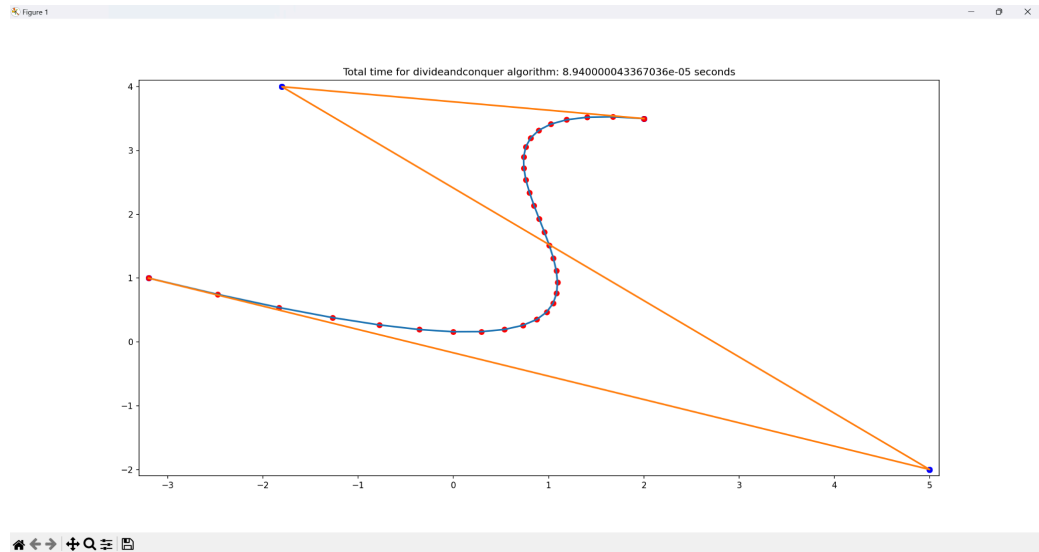
Titik Kontrol : (0.5, 1.2), (-2.3, 3.8), (4.7, -0.6), (1.9, 2.4), (-3.6, -1.1)

Waktu : 9.009999×10^{-5} detik



6. Pengujian 6

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

Waktu : 8.94×10^{-5} detik

3.3 Analisis Perbandingan Hasil Pengujian

Dari kedua pengujian yang sudah dilakukan, dapat dilihat bahwa kecepatan kedua algoritma tidak cukup konsisten untuk dapat dibandingkan. Kompleksitas waktu algoritma dengan notasi Big O juga tidak dapat digunakan sebagai alat ukur karena operasi yang dilakukan kedua algoritma tidaklah sama.

Pada algoritma *brute force*, operasinya:

```
coeff = math.comb(n, i) * ((1 - t) ** (n - i)) * (t**i)
x += coeff * positions[i][0]
y += coeff * positions[i][1]
```

Sementara algoritma *divide and conquer*, algoritmanya:

```
return (1 - t) * bezier_dnc(left_points, t) + t * bezier_dnc(right_points, t)
```

Secara garis besar, memang rumus yang digunakan sama, tetapi pada perhitungan koefisien, ada perbedaan yang signifikan. Pada algoritma *brute force*, perkalian dan perpangkatan dieksekusi secara langsung, tetapi pada algoritma *divide and conquer*, perkalian dan perpangkatan diperantarai oleh pemanggilan fungsi. Pemanggilan fungsi tersebut tentu saja mengonsumsi cukup waktu untuk membuat perbedaan pada selisih durasi eksekusi kedua algoritma.

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, untuk jumlah titik yang banyak dapat membuat algoritma alternatif tersebut menjadi lebih lambat karena pemanggilan fungsi yang banyak.

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](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