

Tugas Besar
IF2211 Strategi Algoritma
Bezier Curve* dengan algoritma *Divide and Conquer



Bezier

Oleh :

Justin Aditya Putra Prabakti 13522130

Muhammad Fauzan Azhim 13522153

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2024

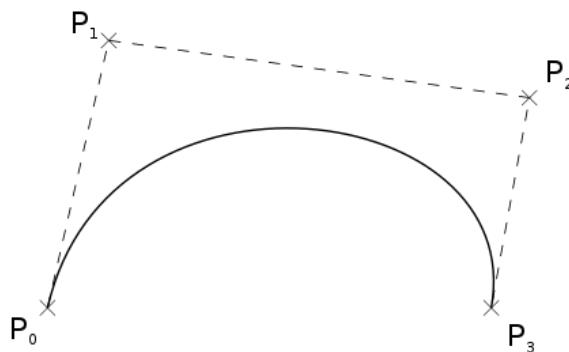
Daftar Isi

Daftar Isi.....	1
Bab I	
Deskripsi Tugas.....	3
Bab II	
Landasan Teori.....	5
2.1 Dasar Teori.....	5
2.2 Membuat Bezier Curve melalui Algoritma garis tengah.....	6
Bab III	
Aplikasi Strategi Divide and Conquer.....	8
3.1 Implementasi algoritma Divide and Conquer.....	8
3.2 Solusi algoritma brute force sebagai algoritma pembanding.....	8
3.3 BONUS : “Bezier” curve N titik.....	8
Bab IV	
Hasil dan Analisis.....	10
BONUS : Bezier curve N titik, dengan titik sbb : (0,0) (1,3) (2,1) (1,0).....	11
4.1 Analisis hasil.....	12
Bab V	
Kesimpulan dan Saran.....	13
5.1 Kesimpulan.....	13
5.2 Saran.....	13
Lampiran.....	14
Daftar Pustaka.....	14

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 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.



Gambar 1. Kurva Bézier Kubik

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

Sebuah kurva Bézier didefinisikan oleh satu set titik kontrol P_2 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.

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.

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

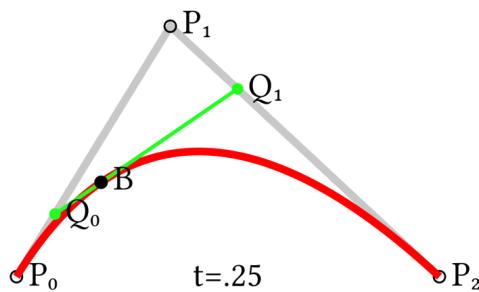
$$Q_1 = B(t) = (1 - t)P_1 + tP_2, \quad t \in [0, 1]$$

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

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

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

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.

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

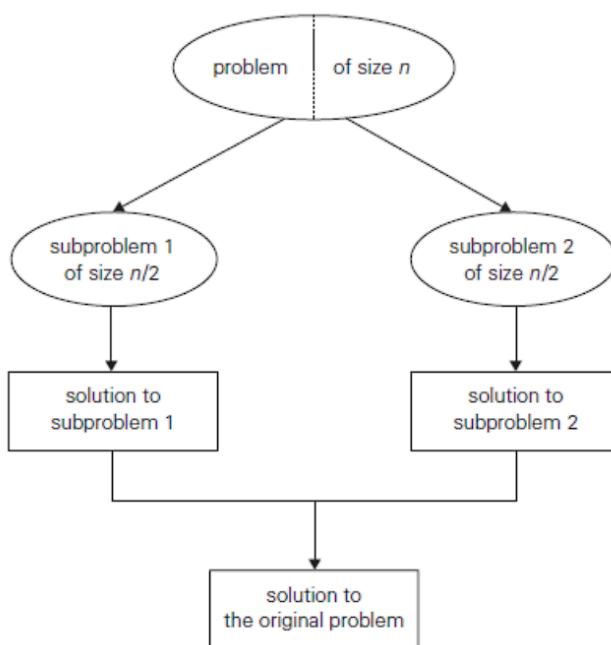
$$T_0 = B(t) = (1-t)^4 P_0 + 4(1-t)^3 t P_1 + 6(1-t)^2 t^2 P_2 + 4(1-t) t^3 P_3 + t^4 P_4, \quad t \in [0, 1]$$

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 II

Landasan Teori

2.1 Dasar Teori



Gambar 3. Ilustrasi algoritma Divide and Conquer.

(Sumber: Algoritma Divide and Conquer (Bagian 1) (Versi baru 2024))

[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)

Algoritma *Divide and Conquer* merupakan algoritma yang terdiri dari tiga bagian : **Divide** atau membagi, **Conquer** atau taklukan/selesaikan, dan **Combine** atau gabungkan.

DIVIDE : Permasalahan yang ada dibagi menjadi beberapa upa-permasalahan (sub-permasalahan) yang identik dengan masalah awal, namun memiliki ukuran yang lebih kecil, sehingga lebih mudah untuk diselesaikan.

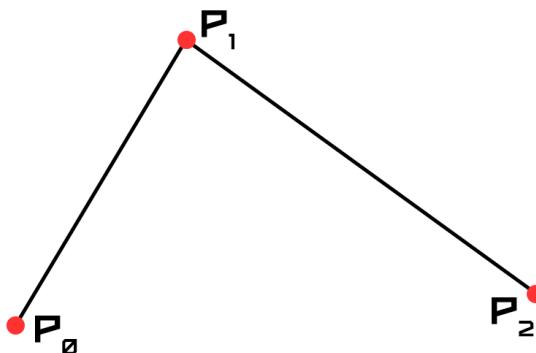
CONQUER : Masing-masing upa-permasalahan diselesaikan (atau dibagi lagi jika masih terlalu besar, hingga bisa diselesaikan)

COMBINE : Solusi semua upa-permasalahan digabungkan menjadi satu solusi yang membentuk solusi permasalahan semula.

2.2 Membuat Bezier Curve melalui Algoritma garis tengah

Selain menggunakan rumus yang telah dijelaskan di bab 1, terdapat cara lain untuk membuat garis bezier yang lebih approachable bagi manusia dan tidak memerlukan perhitungan matematika kompleks seperti yang dijelaskan pada bab 1.

Misalnya terdapat tiga titik: P_0 , P_1 , dan P_2 dengan P_1 sebagai titik kontrol atau titik ditengah,

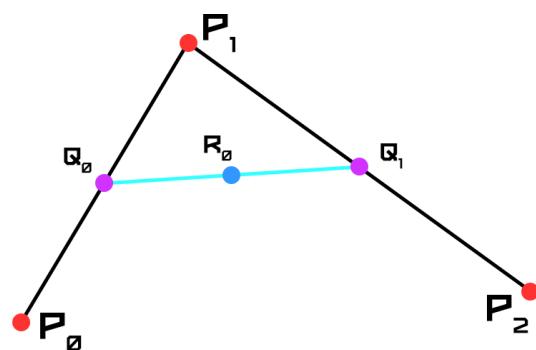


Gambar 4. Ilustrasi awal iterasi

maka dalam satu iterasi akan dilakukan seperti berikut:

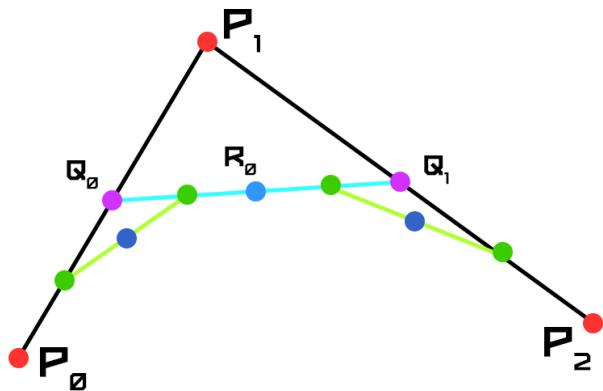
1. Buatlah dua titik baru Q_0 serta Q_1 dimana Q_0 merupakan titik diantara P_0 dengan P_1 dan Q_1 .
2. Buat titik baru R_0 yang berada diantara Q_0 dan Q_1
3. Buat garis yang menghubungkan Q_0 , R_0 , dan Q_1 .

Setelah satu iterasi, akan terbentuk suatu garis “melengkung”, namun masih belum mulus:



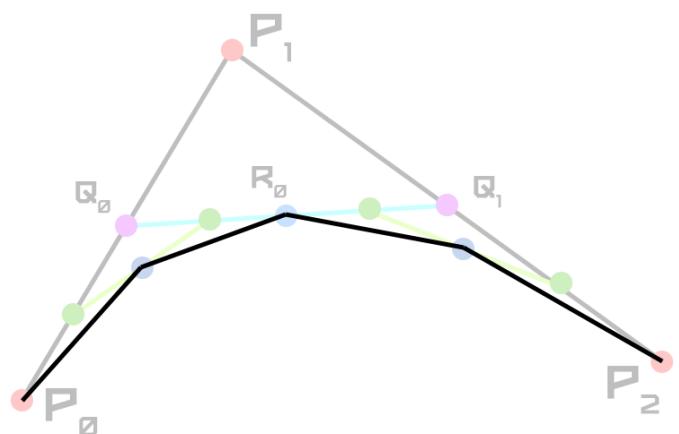
Gambar 5. Ilustrasi menambah titik tengah pertama

Untuk memperoleh garis bezier yang lebih halus, perlu dilakukan lagi iterasi, dimana iterasi kedua dilakukan antara P_0 - Q_0 - R_0 dan R_0 - Q_1 - P_2



Gambar 6. Lanjutan iterasi

Terakhir, ketika jumlah iterasi sudah dirasa cukup, maka akan digabungkan titik-titik mulai dari P_0 , melalui semua titik tengah yang terbuat hingga tersambung ke P_2



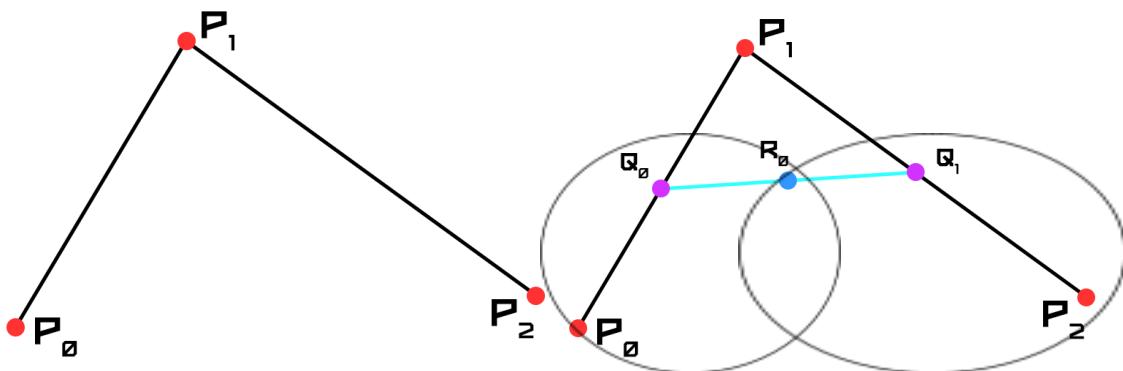
Gambar 7. Hasil final

Bab III

Aplikasi Strategi *Divide and Conquer*

3.1 Implementasi algoritma *Divide and Conquer*

Dalam proses pembuatan *bezier curve* kasus tiga titik, kita menerima tiga titik yang membentuk suatu “sudut” yang nantinya bisa kita proses hingga menjadi suatu bezier curve. Namun jika kita perhatikan gambar 5, ketika menambahkan titik tengah, kita sekarang memiliki 2 sudut baru, serta satu titik tengah yang merupakan bagian dari solusi akhir. Sudut-sudut ini lah yang menjadi upa-permasalahan yang akan diselesaikan secara rekursif dengan memasukkan tiga titik tersebut ke fungsi yang sama, dengan menambahkan informasi iterasi supaya fungsi berhenti pada iterasi tertentu



Gambar 8. Pembentukan 2 sudut baru yang menjadi upa-permasalahan

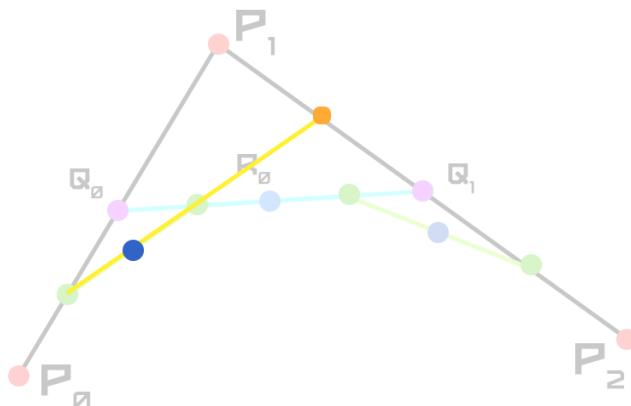
Sehingga solusi merupakan : [Solusi sudut kiri] + titik tengah hasil iterasi + [Solusi sudut kiri], dan terakhir solusi ditambahkan P_0 di paling kiri dan P_1 di paling kanan karena dalam proses iterasi, hanya titik tengah yang diambil.

3.2 Solusi algoritma *brute force* sebagai algoritma pembanding

Untuk membandingkan hasil algoritma Divide and Conquer, dibuat juga implementasi bezier curve dengan metode brute force, yaitu melalui iterasi yang dilakukan dari t terkecil menggunakan rumus yang sudah dijelaskan di bab 1.

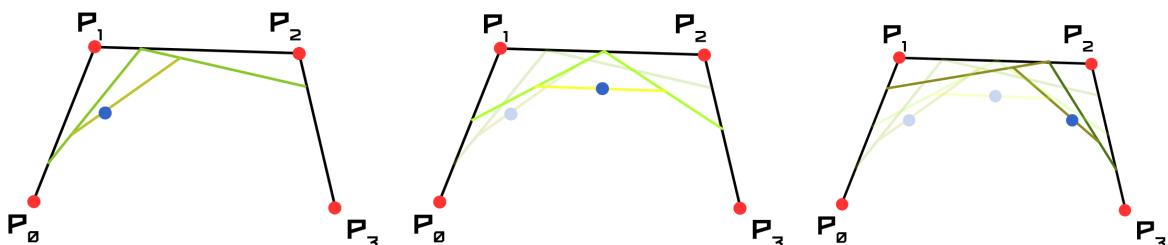
3.3 BONUS : “Bezier” curve N titik

Jika kita lihat pada gambar 5, proses iterasi dilakukan dengan menghubungkan titik tengah garis-garis yang bertetangga/bersebelahan, lalu garis yang dihasilkan tidak memiliki pasangan lain atau sudah tunggal, sehingga titik tengah nya diambil. hal yang sama bisa dilakukan dengan menarik titik dari $\frac{1}{4}$ garis P_0 ke P_1 dan dihubungkan dengan P_1 ke P_2 , lalu dari garis yang terbentuk, diambil titik dari $\frac{1}{4}$ garis, hasilnya akan sama dengan mengambil titik kiri dari iterasi dua seperti gambar 6



Gambar 9. Perbandingan metode bonus

Sehingga ketika berurusan dengan bezier curve dengan lebih dari tiga titik, maka rekursi pertama akan menghasilkan dua garis (satu sudut dari dua) yang ketika diaplikasikan lagi metode diatas akan menghasilkan satu garis. Metode ini diaplikasikan sesuai dengan jumlah iterasi yang diminta. Ini adalah bagian “Divide” dari Divide and Conquer, dimana Conquer adalah mengalikan koefisien iterasi dengan vektor garis, dan Combine adalah menghubungkan semua titik, sesuai dengan urutan dengan memasukkan titik awal dan akhir.



Gambar 10. Visualisasi proses satu iterasi

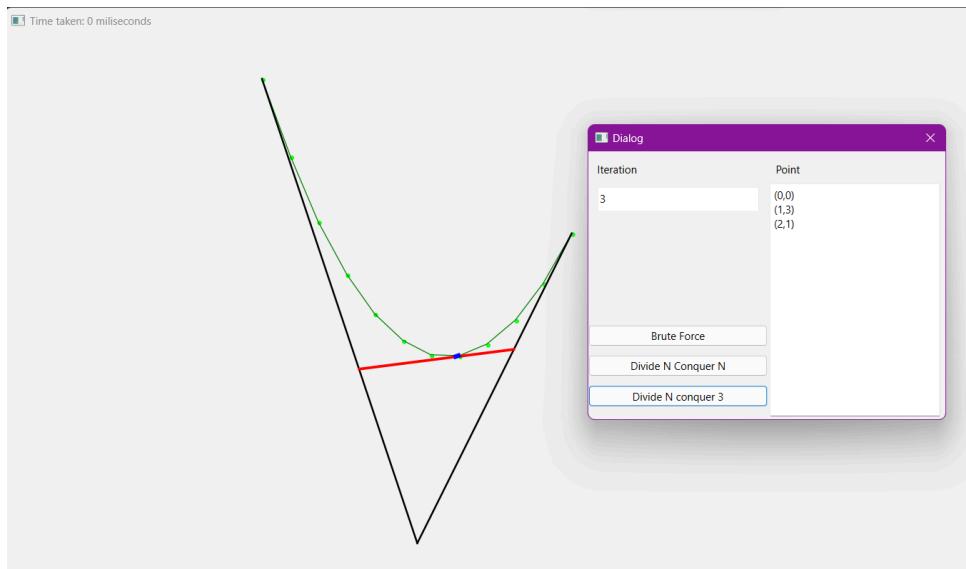
Bab IV

Hasil dan Analisis

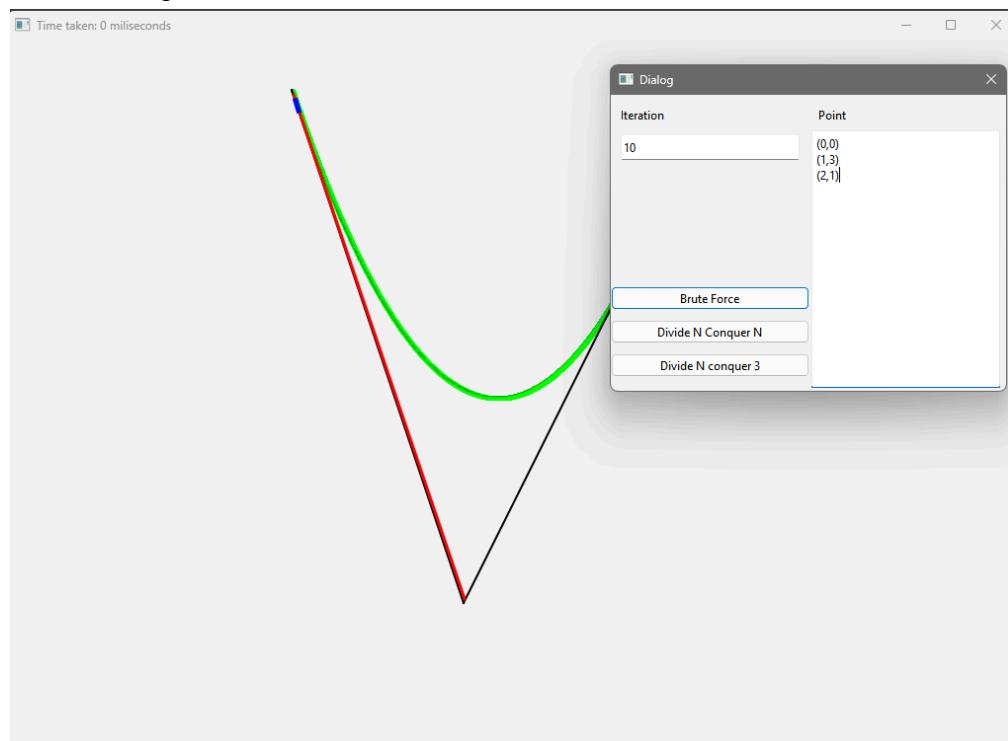
Untuk menguji hasil, kedua algoritma akan dijalankan dengan tiga titik yaitu $[(0,0),(1,3),(2,1)]$. Penginputan titik menggunakan format (x,y) dipisahkan dengan baris baru. Hasil dari waktu eksekusi akan ditampilkan pada title window.

note : Divide and conquer dihitung dengan function 3 points, bukan dengan n points

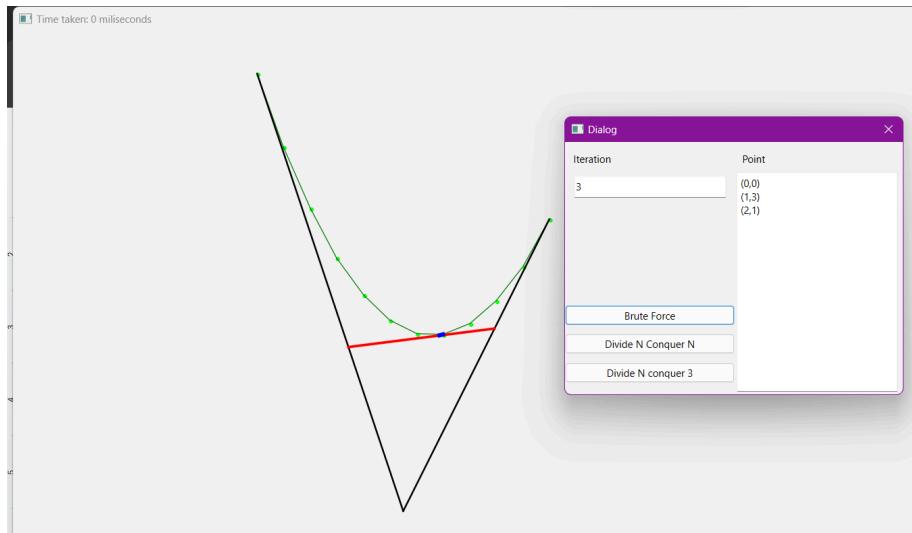
- Divide and Conquer, 3 iterations



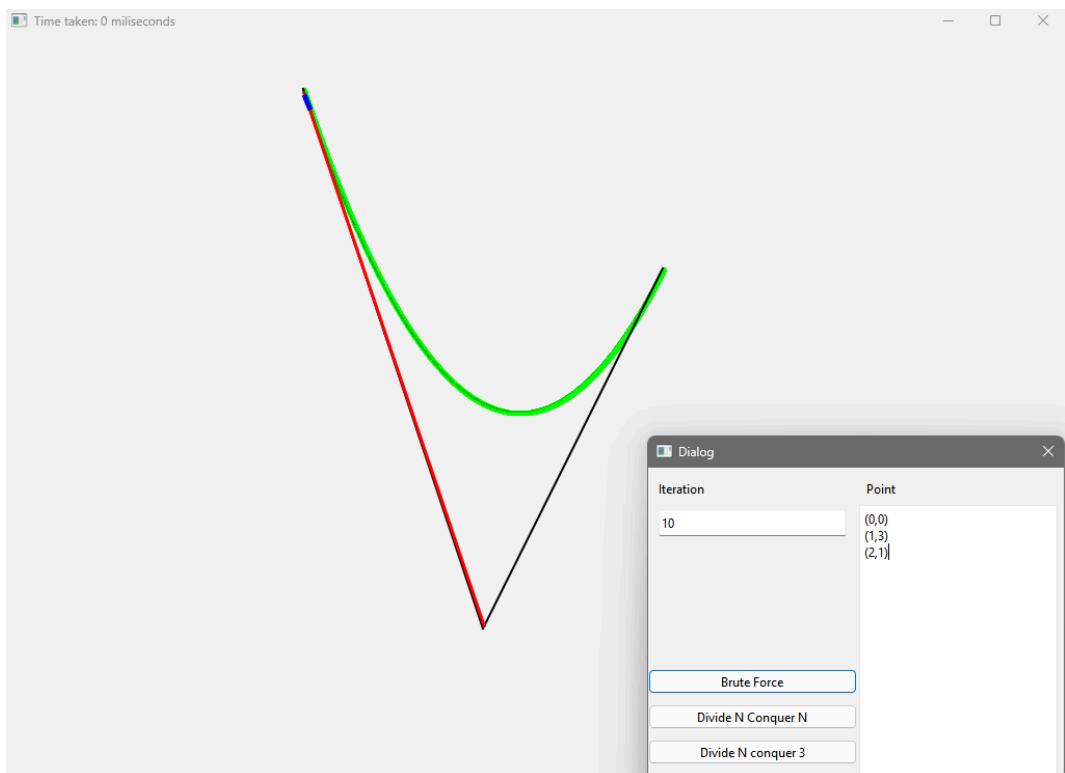
- Divide and conquer, 10 iterations



- Brute force, 3 iterations

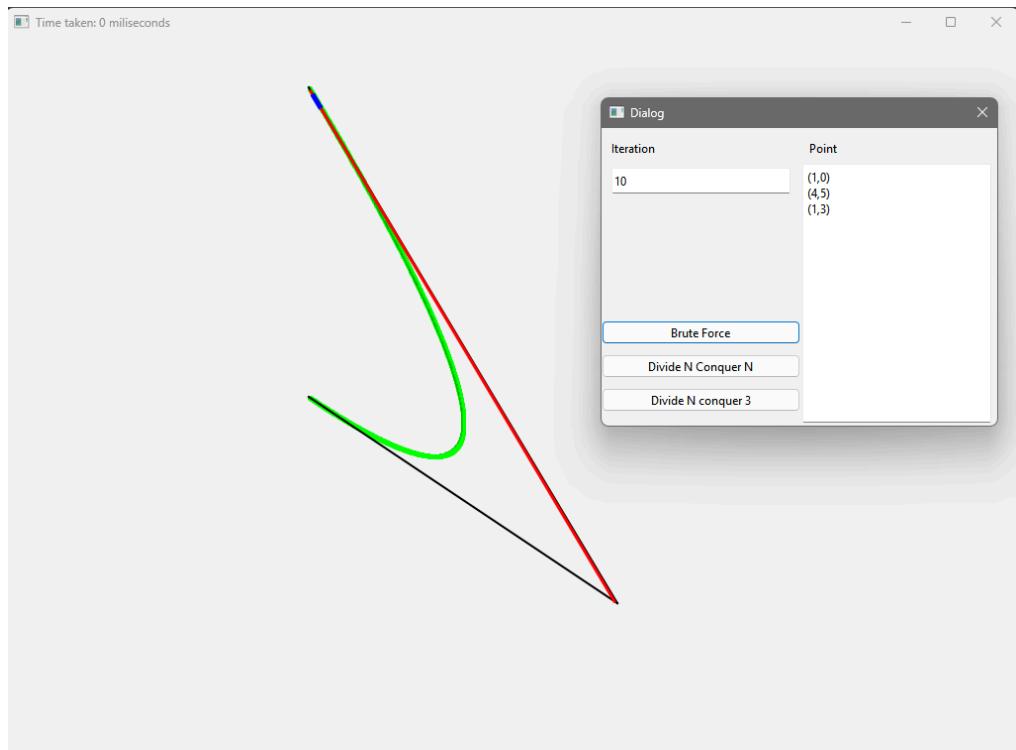


- Brute force, 10 iterations

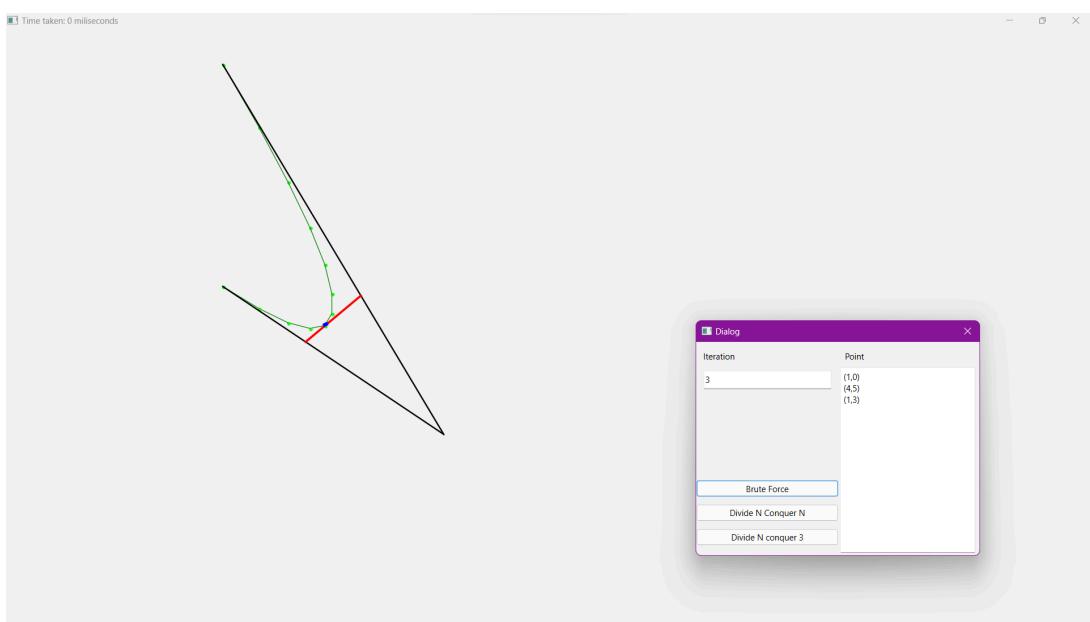


Pengujian selanjutnya, kedua algoritma akan dijalankan dengan tiga titik yaitu $[(1,0),(4,5),(1,3)]$

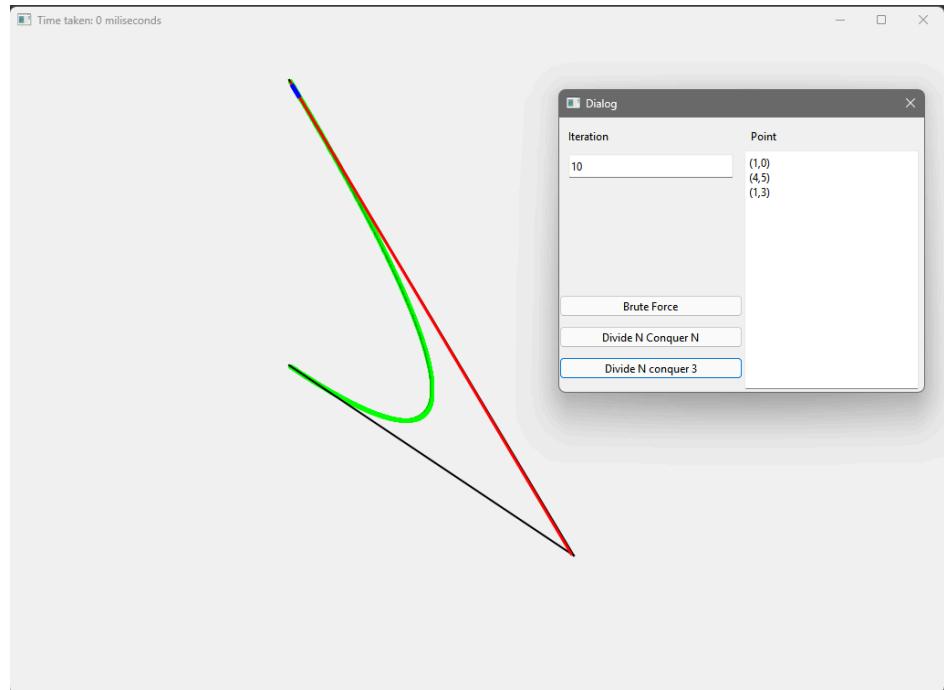
1. BruteForce Iterasi 10



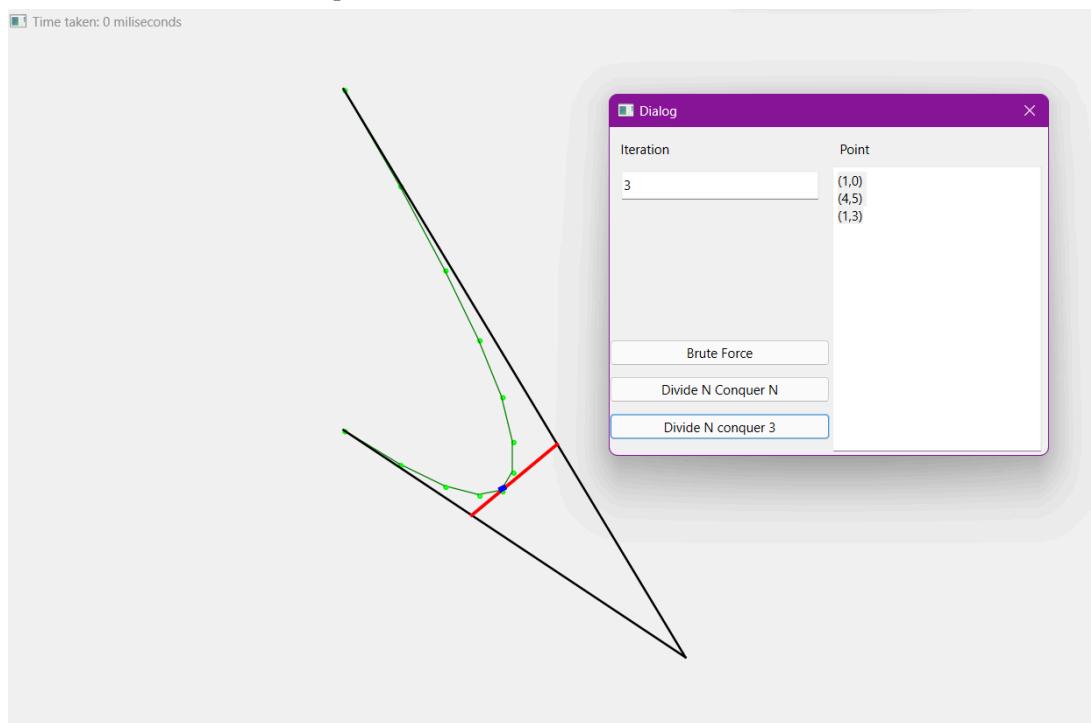
2. BruteForce Iterasi 3



3. Divide and conquer Iterasi 10

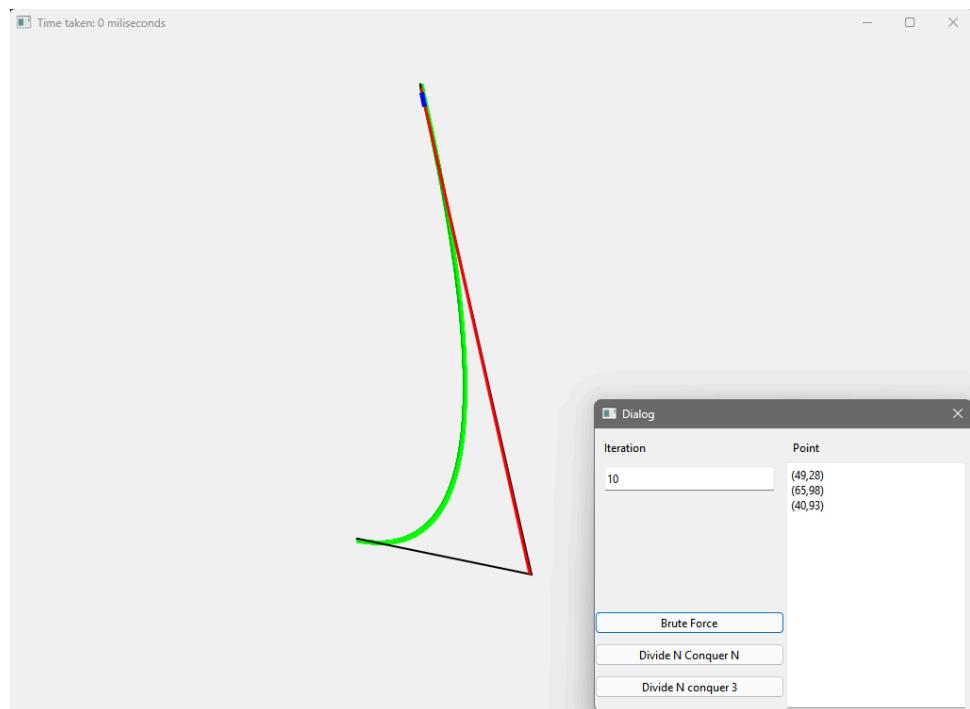


4. Divide and Conquer iterasi 3

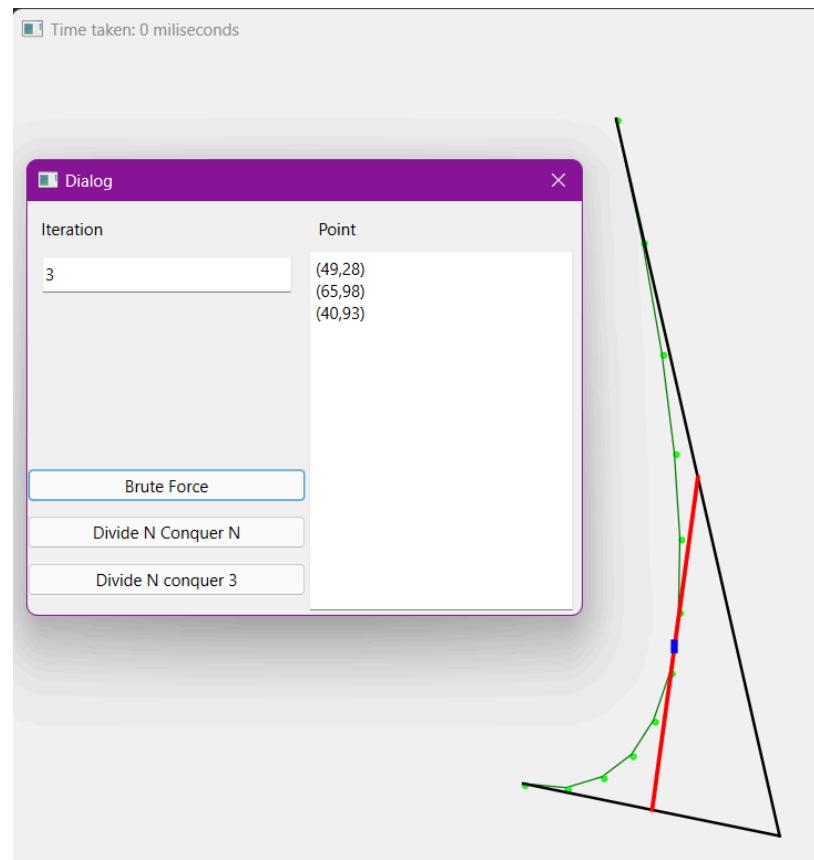


Pengujian selanjutnya, kedua algoritma akan dijalankan dengan tiga titik yaitu $[(2,5),(4,1),(3,3)]$

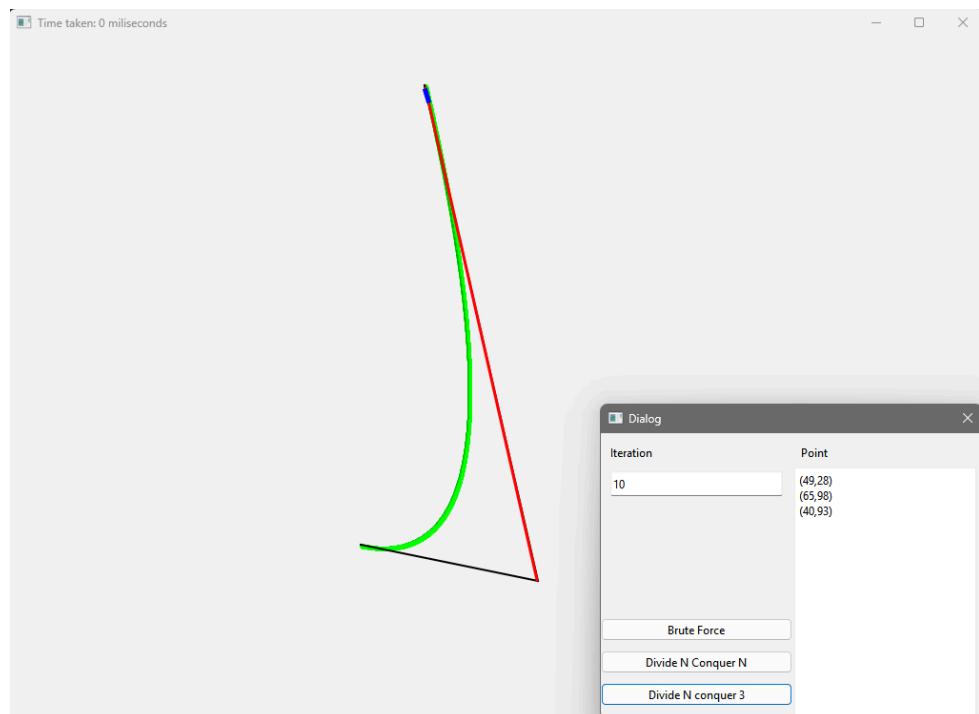
1. BruteForce Iterasi 10



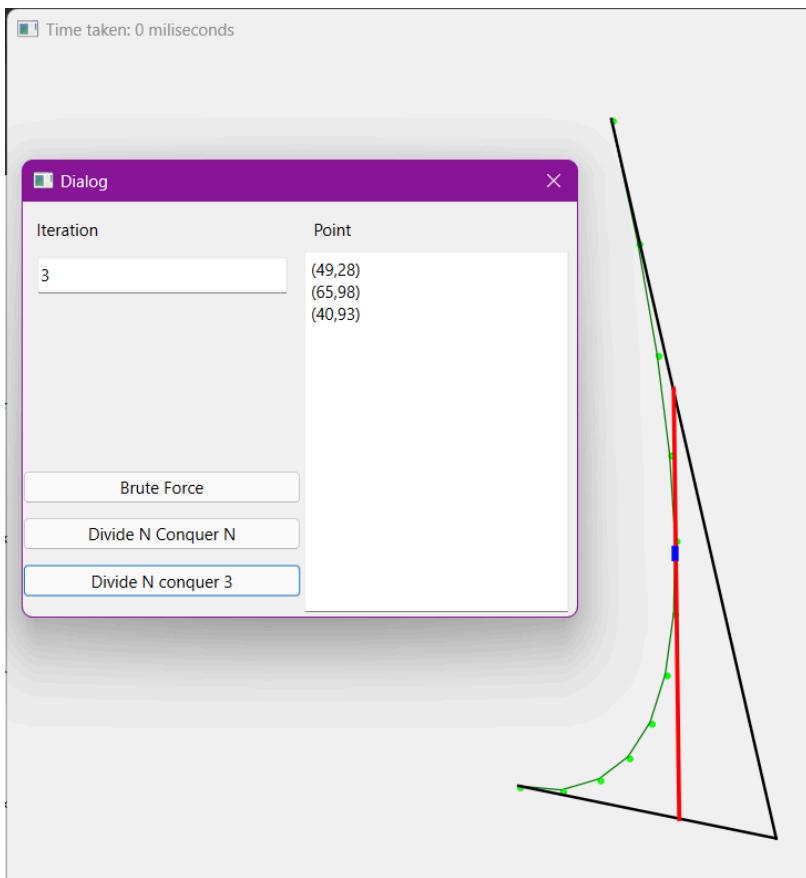
2. BruteForce Iterasi 3



3. Divide And Conquer Iterasi 10

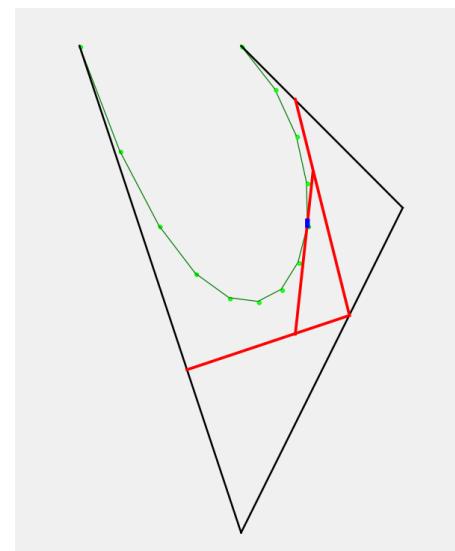


4. Divide And Conquer Iterasi 3

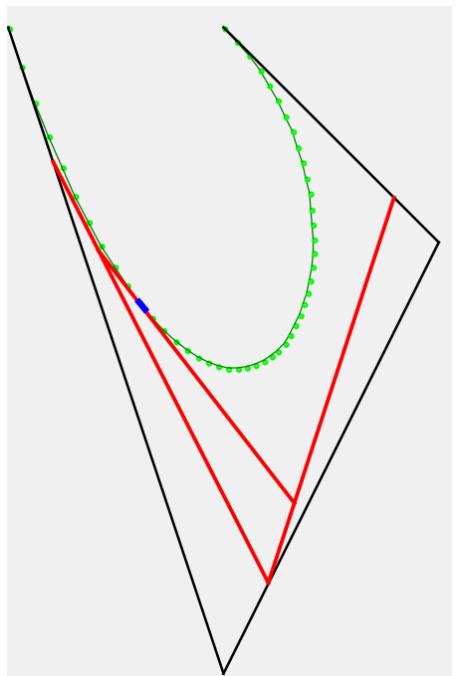


BONUS : Bezier curve N titik, dengan titik sbb : (0,0) (1,3) (2,1) (1,0)

- Divide and conquer, 3 iterasi

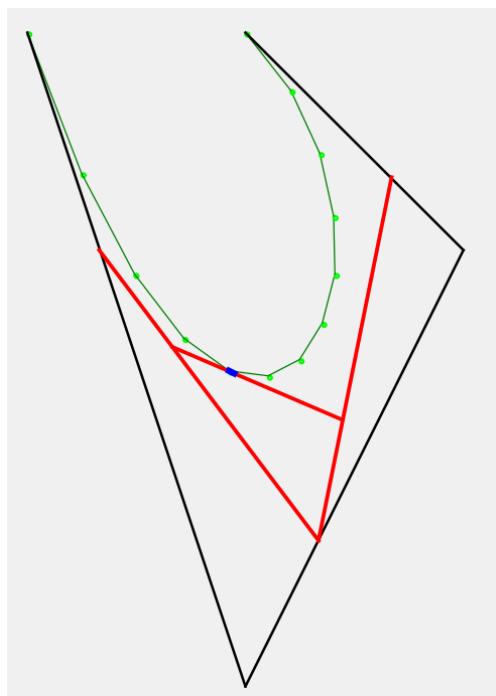


- Divide and conquer, 15 iterasi



⌚ Time taken: 24 miliseconds

- Brute force, 3 iterasi



⌚ Time taken: 0 miliseconds

- Brute force, 15 iterasi

⌚ Time taken: 7 miliseconds

4.1 Analisis hasil

Hasil menunjukkan bahwa metode *brute force* memerlukan waktu yang lebih sedikit (sekitar seperempat!) untuk menghitung solusi N titik. Penyebab utama dari kelambatan metode *divide and conquer* dari algoritma kita adalah jumlah rekursi yang sangat tinggi. Untuk menghitung satu titik saja, diperlukan $N-1$ rekursi dimana N adalah jumlah titik yang ada, dalam satu rekursi pun masih besar karena harus menghitung setiap titik.

Ketika dibandingkan dengan metode brute-force yang tidak menggunakan banyak rekursi atau divide and conquer dengan 3 titik, maka terlihat jelas bahwa metode brute-force lebih efisien.

Bab V

Kesimpulan dan Saran

5.1 Kesimpulan

Masalah yang berbeda dapat memerlukan strategi algoritma yang berbeda untuk memecahkannya. Selain itu, skala dari masalah yang sama juga dapat mempengaruhi efektivitas dari masing-masing algoritma.

5.2 Saran

Kami menyarankan pada saat menggunakan algoritma *divide and conquer* untuk memperhatikan kompleksitas yang disebabkan ketika sedang meng-*divide* suatu masalah. Memisahkan suatu masalah menjadi bagian-bagian yang terlalu kecil malah dapat menyulitkan proses kalkulasi.

Lampiran

Repository program dapat dilihat pada tautan berikut:
https://github.com/fauzanazz/Tucil2_13522130_13522153

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.	✓	

Daftar Pustaka

1. Rinaldi Munir. (2024). Algoritma Divide and Conquer (Bagian 1) (Versi baru 2024)
[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)
2. Rinaldi Munir. (2024). Algoritma Divide and Conquer (Bagian 2) (Versi baru 2024)
[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)
3. Rinaldi Munir. (2024). Algoritma Divide and Conquer (Bagian 3) (Versi baru 2024)
[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)
4. Rinaldi Munir. (2024). Algoritma Divide and Conquer (Bagian 4) (Versi baru 2024)
[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)