

TUGAS KECIL 2
IF2211 STRATEGI ALGORITMA
Membangun Kurva Bézier dengan Algoritma Titik Tengah
berbasis Divide and Conquer



DISUSUN OLEH:
Jonathan Emmanuel Saragih **13522121**
Aland Mulia Pratama **13522124**

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

DAFTAR ISI

DAFTAR ISI	1
BAB I	3
DESKRIPSI MASALAH DAN ALGORITMA	3
1.1 Algoritma Brute Force	3
1.2 Algoritma Divide and Conquer	3
1.3 Pembentukan Kurva Bezier (Kurva Mulus)	4
1.4 Algoritma Penentuan Titik Bezier dengan Pendekatan Brute Force	7
1.5 Algoritma Penentuan Titik Bezier dengan Pendekatan Divide and Conquer	7
BAB II	9
IMPLEMENTASI DAN ANALISIS ALGORITMA DALAM BAHASA JAVASCRIPT DENGAN FRAMEWORK NEXT.JS	9
2.1 ./app/page.jsx	9
2.2 ./app/brute-force/page.jsx	9
2.3 ./utils/bruteforce.js	12
2.4 ./app/div-n-conquer/page.jsx	14
2.5 ./utils/divnconquer.jsx	17
BAB III	20
SOURCE CODE PROGRAM	20
3.1 Repository Program	20
3.2 Source Code Program	20
3.2.1. ./app/page.jsx	20
3.2.2. ./app/brute-force/page.jsx	21
3.2.3. ./utils/bruteforce.js	25
3.2.4. ./app/div-n-conquer/page.jsx	26
3.2.5. ./utils/divnconquer.jsx	30
BAB IV	31
UJI COBA PROGRAM	31
4.1 Set Percobaan Pertama	31
4.2 Set Percobaan Kedua	33
4.3 Set Percobaan Ketiga	35
4.4 Set Percobaan Keempat	37
4.5 Set Percobaan Kelima	39
4.6 Set Percobaan Keenam	41
4.7 Percobaan Kurva Bezier Kubik	43
4.8 Percobaan Kurva Bezier Kuartik	45
BAB V	48
ANALISIS, SARAN, DAN KESIMPULAN	48
5.1 Analisis Efektivitas Algoritma	48
5.2 Saran	48
5.3 Kesimpulan	49
5.4 Refleksi	49
LAMPIRAN	50

DAFTAR TABEL

Tabel 1 Method Table bruteforce page	9, 10
Tabel 2 Variable Table bruteforce page	11, 12
Tabel 3. Method Table utils bruteforce	13
Tabel 4 Variable Table utils bruteforce	13
Tabel 5 Method Table div-n-conquer page	13, 14
Tabel 6 Variable div-n-conquer page Deskripsi Kebutuhan Non Fungsional Perangkat Lunak	14, 15
Tabel 7. Method utils divnconquer	18
Tabel 8. Method utils divnconquer	18

BAB I

DESKRIPSI MASALAH DAN ALGORITMA

1.1 Algoritma Brute Force

Algoritma Brute Force merupakan pendekatan yang langsung dan sederhana untuk memecahkan suatu persoalan dengan cara yang jelas dan straightforward. Dalam algoritma ini, setiap kemungkinan solusi dieksplorasi secara sistematis tanpa memanfaatkan pengetahuan khusus tentang struktur masalah. Sebagai contoh, dalam mencari elemen terbesar dalam sebuah senarai, setiap elemen dibandingkan satu per satu untuk menemukan yang terbesar. Namun, kelemahannya terletak pada keterbatasan efisiensi karena memerlukan volume komputasi besar dan waktu yang lama, terutama ketika ukuran masukan (n) menjadi besar. Meskipun demikian, kekuatannya terletak pada kemampuannya untuk memecahkan sebagian besar masalah dengan pendekatan yang sederhana dan mudah dimengerti.

Karakteristik algoritma Brute Force juga mencerminkan keunggulan dan kelemahannya. Algoritma ini umumnya tidak memberikan solusi yang optimal atau efisien, namun dapat diterapkan pada berbagai macam masalah. Teknik brute force cocok digunakan ketika ukuran masukan kecil dan implementasinya mudah. Meskipun demikian, untuk masukan yang besar, performanya tidak dapat diterima. Namun, algoritma Brute Force tetap penting karena seringkali menjadi dasar perbandingan dengan algoritma yang lebih canggih dan efisien.

Sementara algoritma Brute Force menawarkan kemudahan dalam implementasi dan pemahaman, kelemahannya dalam efisiensi seringkali memicu eksplorasi alternatif. Salah satu teknik yang digunakan untuk meningkatkan kinerja algoritma Brute Force adalah teknik heuristik. Heuristik memungkinkan peningkatan kinerja dengan menghilangkan beberapa kemungkinan solusi tanpa harus mengeksplorasi semua kemungkinan secara menyeluruh. Meskipun berbasis pada pendekatan intuitif dan penilaian non-formal, teknik heuristik dapat membantu meningkatkan efisiensi algoritma Brute Force dalam menyelesaikan masalah.

Dalam banyak kasus, algoritma Brute Force digunakan sebagai titik awal untuk memahami dan memecahkan masalah yang kompleks. Meskipun mungkin tidak memberikan solusi optimal, algoritma ini tetap menjadi alat yang sangat berguna dalam penyelesaian berbagai macam persoalan. Karena sifatnya yang straightforward dan jelas, algoritma Brute Force membantu dalam pemahaman dasar tentang struktur masalah, sehingga dapat menjadi pondasi untuk pengembangan solusi yang lebih canggih dan efisien.

1.2 Algoritma Divide and Conquer

Algoritma *divide and conquer* merupakan sebuah metode dalam menyelesaikan masalah pemrograman untuk mencari suatu solusi. Pada algoritma *divide and conquer*, terdapat dua elemen utama, yaitu *Divide and Conquer*. Dalam pendekatan *divide and conquer* untuk menangani suatu masalah, langkah awal adalah memecah masalah tersebut menjadi sub-masalah yang lebih kecil dan mirip dengan masalah asli. Idealnya, sub-masalah ini seharusnya berukuran seragam. Setelah masalah terpecah, tiap sub-masalah dipecahkan (*conquer*) baik secara langsung atau melalui metode rekursif jika sub-masalah masih terlalu besar. Pada akhirnya, solusi dari semua sub-masalah dikombinasikan untuk mendapatkan solusi masalah utama. Karena karakteristik yang seragam di setiap sub-masalah, pendekatan divide and conquer sering kali lebih efektif dengan menggunakan rekursi, baik dalam konsep maupun implementasi.

Metode Divide and Conquer adalah sebuah paradigma algoritma yang terdiri dari tiga langkah utama: Divide (membagi), Conquer (menaklukkan), dan Combine (menggabungkan). Pertama, persoalan utama dibagi menjadi beberapa upa-persoalan yang memiliki kemiripan

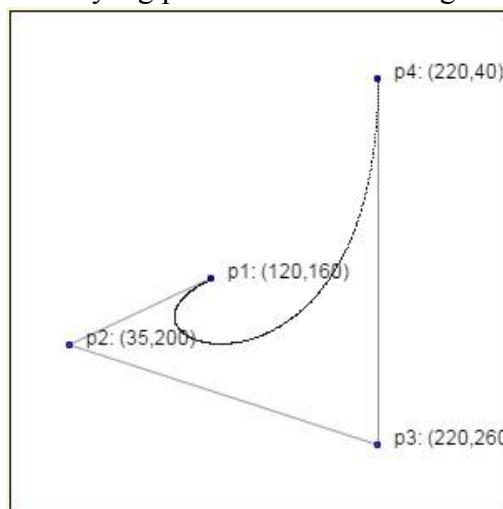
dengan persoalan semula namun berukuran lebih kecil. Setiap upa-persoalan idealnya memiliki ukuran hampir sama. Kedua, setiap upa-persoalan diselesaikan secara langsung jika sudah berukuran kecil, atau secara rekursif jika masih berukuran besar. Langkah terakhir adalah menggabungkan solusi masing-masing upa-persoalan untuk membentuk solusi persoalan semula.

Objek persoalan yang dipecahkan menggunakan metode Divide and Conquer biasanya adalah masukan (input) atau instances persoalan yang memiliki ukuran n , seperti tabel (larik), matriks, eksponen, polinom, dan lainnya, bergantung pada sifat persoalan tersebut. Setiap upa-persoalan yang dihasilkan memiliki karakteristik yang sama dengan persoalan semula, namun berukuran lebih kecil. Oleh karena itu, penggunaan metode Divide and Conquer lebih alami diungkapkan dalam skema rekursif, karena struktur permasalahan tersebut memungkinkan pemecahan bertahap.

Ada beberapa persoalan klasik yang diselesaikan dengan menggunakan metode Divide and Conquer. Ini termasuk Persoalan MinMaks (mencari nilai minimum dan maksimum), menghitung perpangkatan, pengurutan (sorting) seperti *Mergesort* dan *Quicksort*, mencari sepasang titik terdekat, Convex Hull, perkalian matriks, perkalian bilangan bulat besar, perkalian dua polinom, dan lain sebagainya. Pendekatan ini sering memberikan solusi yang efisien untuk persoalan-persoalan tersebut, dengan membagi persoalan besar menjadi sub-persoalan yang lebih kecil dan lebih mudah dipecahkan. Dengan demikian, metode Divide and Conquer menjadi alat yang berguna dalam menyelesaikan berbagai masalah dengan efisien.

1.3 Pembentukan Kurva Bezier (Kurva Mulus)

Kurva Bézier adalah sebuah jenis kurva yang halus dan sering dipakai dalam berbagai bidang seperti desain grafis, animasi, dan manufaktur. Kurva ini dibentuk dengan menghubungkan beberapa titik kontrol yang menentukan bentuk dan arah kurva. Pembuatan kurva Bézier relatif mudah, yakni dengan menetapkan titik kontrol dan menghubungkannya membentuk kurva. Keberagaman penggunaan kurva Bézier dalam kehidupan sehari-hari mencakup alat pena digital, animasi yang tampak mulus dan realistik, desain produk yang rumit dan akurat, serta pembuatan font yang menarik dan unik. Salah satu kelebihan utama kurva Bézier adalah kemampuannya untuk diubah dan dimanipulasi dengan mudah, sehingga memungkinkan pembuatan desain yang presisi dan sesuai dengan kebutuhan.



Gambar 1. Ilustrasi Kurva Bezier dengan empat titik kontrol
(Sumber: <https://majalah1000guru.net/2015/02/kurva-bezier/>)

Sebuah kurva Bézier ditentukan oleh serangkaian titik kontrol dari P₀ hingga P_n, di

mana n menunjukkan orde kurva ($n = 1$ untuk garis lurus, $n = 2$ untuk kuadrat, dan seterusnya). Titik kontrol awal dan akhir selalu menjadi titik ujung dari kurva, tetapi titik kontrol di antara keduanya (jika ada) biasanya tidak secara langsung berada pada jalur kurva tersebut. Berdasarkan gambar 1, P_0 merupakan titik kontrol pertama, sedangkan P_3 adalah titik kontrol terakhir. Titik kontrol P_1 dan P_2 , yang disebut sebagai titik kontrol antara, tidak secara langsung terletak pada kurva yang terbentuk.

Implementasi kurva Bézier dengan menggunakan algoritma brute-force sebenarnya cukup sederhana dari sisi programmer. Kurva Bézier ini dibentuk dengan menggunakan persamaan sebagai berikut:

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

Pada persamaan diatas, terdapat sebuah titik Q_0 yang berada pada garis yang dibentuk oleh P_0 dan P_1 . Dalam fungsi kurva Bézier linier, pergerakan $B(t)$ dari P_0 ke P_1 sejauh seberapa besar t dalam rentang nilai 0 hingga 1. Misalkan, ketika $t = 0.25$, maka $B(t)$ mencapai seperempat jarak dari titik P_0 ke P_1 . Oleh karena itu, ketika t berubah dari 0 hingga 1, seluruh rentang nilai t akan menghasilkan persamaan $B(t)$ membentuk garis lurus yang menghubungkan P_0 dan P_1 .

Dalam kasus lain, kita dapat menambahkan titik baru, P_2 , antara P_0 dan P_1 , dengan P_0 dan P_2 sebagai titik kontrol awal dan akhir, dan P_1 sebagai titik kontrol antara. Jika kita letakkan titik Q_1 di antara garis yang menghubungkan P_1 dan P_2 , maka kita dapat membentuk sebuah kurva Bézier linier yang berbeda dengan P_0 sebagai titik awal dan Q_0 sebagai titik akhir. Selanjutnya, kita dapat menemukan sebuah titik baru, R_0 , yang berada di antara garis yang menghubungkan Q_0 dan Q_1 , yang kemudian membentuk kurva Bézier kuadratik baru dengan P_0 dan P_2 sebagai titik awal dan akhir. Berikut adalah persamaannya:

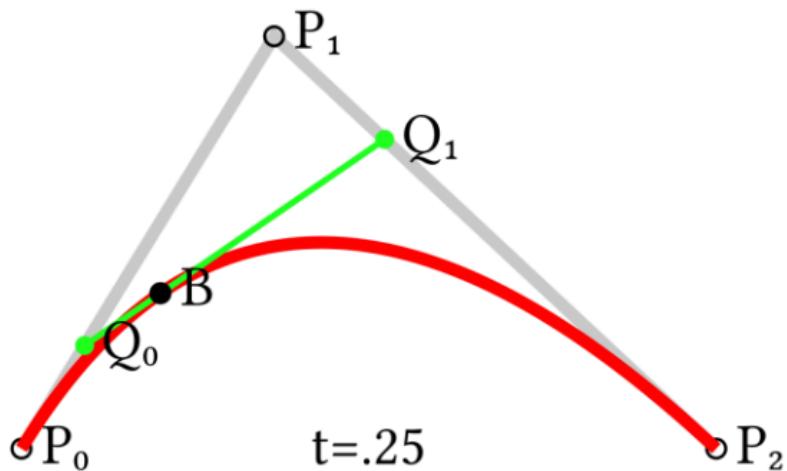
$$Q_0 = B(t) = (1 - t)P_0 + tP_1, \quad t \in [0, 1]$$

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

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

Ketiga persamaan tersebut dapat digabungkan menjadi satu persamaan dengan melakukan substitusi persamaan Q_0 dan Q_1 kedalam persamaan R_0 menjadi seperti berikut:

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



Gambar 2. Kurva Bézier kuadratik (3 titik kontrol)

Proses ini dapat diaplikasikan untuk jumlah titik yang lebih dari 3 titik kontrol. Misalkan, empat titik kontrol akan menghasilkan *kurva Bézier kubik*, lima titik kontrol akan menghasilkan *kurva Bézier kuartik*, dan seterusnya. Berikut adalah persamaan kurva Bézier

kubik dan kuartik dengan menggunakan prosedur yang sama dengan yang sebelumnya.

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

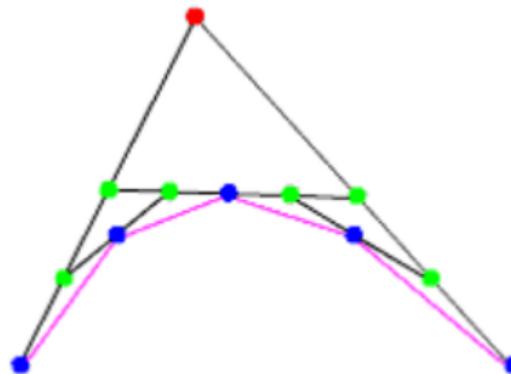
$$T_0 = B(t) = (1-t)^4 P_0 + 4(1-t)^3 t P_1 + 6(1-t)^2 t^2 P_2 + 4(1-t)t^3 P_3 + t^4 P_4, \quad t \in [0, 1]$$

Persamaan diatas tentunya akan semakin rumit seiring bertambahnya jumlah titik kontrol. Program ini memerlukan komputasi yang rumit dalam prosesnya meskipun algoritma ini mudah untuk diimplementasikan ke dalam program. Oleh sebab itu, dalam rangka melakukan efisiensi sehingga semakin mangkus dan sangkil dalam program pembuatan kurva Bézier yang sangat berguna ini, kami perlu mengimplementasikan pembuatan kurva Bézier dengan algoritma titik tengah berdasarkan prinsip dari algoritma divide and conquer.

Ide untuk algoritma divide and conquer bisa dibilang sederhana, prosesnya dimulai dengan tiga titik, yaitu P_0 , P_1 , dan P_2 , dengan P_1 sebagai titik kontrol antara. Langkah pertama adalah menciptakan dua titik baru, Q_0 dan Q_1 , yang masing-masing berada di tengah garis yang menghubungkan P_0-P_1 dan P_1-P_2 . Kemudian, kita menghubungkan Q_0 dan Q_1 untuk membentuk garis baru. Titik baru, R_0 , dibuat di tengah garis Q_0-Q_1 . Dengan garis yang dibuat dari P_0 ke R_0 dan dari R_0 ke P_2 , kita telah melakukan satu iterasi dan mendapatkan sebuah kurva yang masih belum cukup mulus dengan tiga titik.

Untuk meningkatkan kualitas kurva, dilakukan iterasi lanjutan. Titik-titik baru, S_0 , S_1 , S_2 , dan S_3 , dibuat di tengah-tengah segmen antara P_0-Q_0 , Q_0-R_0 , R_0-Q_1 , dan Q_1-P_2 . Kemudian, garis-garis baru dibuat menghubungkan S_0-S_1 dan S_2-S_3 . Dua titik baru, T_0 dan T_1 , dibuat di tengah segmen antara S_0-S_1 dan S_2-S_3 . Terakhir, garis yang menghubungkan $P_0-T_0-R_0-T_1-P_2$ dibuat.

Melalui iterasi kedua, kurva yang dihasilkan semakin mendekati kurva Bézier dengan aproksimasi lima titik. Proses ini dapat diulangi dengan iterasi tambahan untuk mendekati kurva yang semakin halus, dengan jumlah titik yang semakin banyak. Ini mengilustrasikan bahwa semakin banyak iterasi yang dilakukan, kurva yang dihasilkan akan semakin mendekati kurva Bézier.



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

Tantangan terbesar dalam pembuatan algoritma divide and conquer adalah algoritma ini harus lebih mangkus dan sangkil dibandingkan algoritma brute force. Berdasarkan kompleksitas algoritma, perbedaan efektifitas kedua algoritma ini akan mulai terlihat saat jumlah titik taksiran yang dihasilkan berjumlah sangat banyak. Pada **tugas kecil 2 IF2211 Strategi Algoritma** kami ingin membuktikan bahwa algoritma divide and conquer lebih efisien dibandingkan algoritma brute force.

1.4 Algoritma Penentuan Titik Bezier dengan Pendekatan Brute Force

Algoritma brute force dapat menentukan titik titik dalam membangun Kurva Bezier. Algoritma ini merupakan algoritma yang melakukan pendekatan secara langsung dan sederhana untuk menentukan titik titik pada kurva dengan mencoba berbagai kemungkinan titik kontrol yang memenuhi syarat-syarat tertentu. Dalam kasus ini, titik harus dimasukkan secara berurutan untuk mendapatkan hasil yang benar. Berikut merupakan langkah untuk mendapatkan kurva Bezier yang dicari menggunakan strategi algoritma brute force :

1. Input Pengguna:
 - a. Pengguna memasukkan jumlah titik kontrol, direpresentasikan dalam koordinat x dan y.
 - b. Jumlah iterasi ditentukan oleh pengguna, mempengaruhi kehalusan hasil gambar.
2. Proses Menghitung Titik Kurva Bezier:
 - a. Setelah mendapatkan input, program menghitung titik-titik pada kurva Bezier.
 - b. Penghitungan dilakukan dengan iterasi melalui semua titik kontrol.
 - c. Menggunakan rumus polinomial Bezier untuk menentukan kontribusi tiap titik kontrol pada titik kurva.
3. Penyimpanan Hasil dan Visualisasi:
 - a. Titik-titik hasil perhitungan disimpan dalam array untuk visualisasi.
 - b. Proses visualisasi menggunakan Recharts pada website.
 - c. Recharts memerlukan array objek dengan key x dan y yang terdefinisi.
4. Pengurutan dan Penampilan Data di Recharts:
 - a. Array objek harus diurutkan berdasarkan sumbu x.
 - b. Urutan ini penting karena Recharts menampilkan titik sesuai dengan urutan dalam array.
 - c. Pastikan titik-titik diurutkan dengan benar sebelum diserahkan ke Recharts.
5. Visualisasi Akhir:
 - a. Kurva Bezier ditampilkan di website dengan urutan yang benar berdasarkan sumbu x.
 - b. Visualisasi mencerminkan keseluruhan titik yang telah dihitung dengan algoritma brute force.

1.5 Algoritma Penentuan Titik Bezier dengan Pendekatan Divide and Conquer

Algoritma Divide and Conquer dapat digunakan untuk menentukan titik titik dalam membangun kurva bezier. Algoritma ini akan membagi proses pembuatan kurva menjadi sub masalah yang lebih kecil dan kemudian menyelesaikan sub masalahnya secara terpisah, lalu setelah menyelesaikan sub masalahnya, solusi dari sub masalah akan digabung untuk membentuk kurva. Dalam kasus ini, titik harus dimasukkan secara berurutan untuk mendapatkan hasil yang benar. Berikut merupakan langkah untuk mendapatkan kurva Bezier yang dicari menggunakan algoritma Divide and Conquer :

1. Proses Awal:
 - a. Mirip dengan algoritma brute force, menerima titik-titik kontrol dan jumlah iterasi sebagai input.
 - b. Kurva dibagi menjadi bagian yang lebih kecil menggunakan metode Divide and Conquer.
 - c. Pembagian terus dilakukan hingga mencapai tingkat detail yang diinginkan.
2. Pembagian Kurva dan Titik Kontrol:
 - a. Hitung titik tengah antara titik kontrol yang berdekatan.
 - b. Buat titik kontrol baru berdasarkan titik tengah ini.
 - c. Proses rekursif ini berulang hingga mencakup semua titik kontrol.
3. Penyelesaian dan Penggabungan Sub-Masalah:
 - a. Setelah semua titik kontrol dan titik pada kurva dihitung, kurva Bezier digambarkan secara menyeluruh.
4. Implementasi dengan Recharts dan Next.js:
 - a. Kurva Bezier ditampilkan melalui website dengan menggunakan pustaka Recharts di framework Next.js.
 - b. Recharts menerima input berupa map; kunci (key) dari map harus terdefinisi untuk membedakan titik sebagai koordinat x dan y.
 - c. Map yang diterima harus terurut berdasarkan sumbu-x karena Recharts menampilkan titik sesuai urutan.
5. Visualisasi Kurva Bezier:
 - a. Visualisasi akhir kurva Bezier adalah hasil penggabungan semua titik yang telah dihitung dan diurutkan.
 - b. Dengan pendekatan ini, pembuatan kurva Bezier menjadi lebih efisien dan terstruktur, memanfaatkan kekuatan algoritma Divide and Conquer dalam memecahkan masalah kompleks menjadi lebih sederhana dan terkelola.

BAB II

IMPLEMENTASI DAN ANALISIS ALGORITMA DALAM BAHASA JAVASCRIPT DENGAN FRAMEWORK NEXT.JS

Dalam pembuatan program pembentukan kurva bezier (kurva mulus), kami menggunakan bahasa pemrograman JavaScript dengan framework situs web Next.Js yang dibangun diatas React.js. React.js merupakan sebuah pustaka yang disediakan bahasa pemrograman JavaScript yang sering digunakan untuk membangun antarmuka pengguna (UI) dalam pengembangan aplikasi web. Dalam rangka membuat program kami modular, kami membagi program ini ke dalam beberapa file yaitu `./app/page.jsx`, `./app/brute-force/page.jsx`, `./utils/bruteforce.js`, `./app/div-n-conquer/page.jsx` dan juga `./utils/divnconquer.jsx` dengan asumsi root directory berada pada folder src.

Dalam pengembangan program ini, proses perhitungan titik tidak menggunakan *backend* yang terpisah dari framework Next.JS. Karena perhitungannya sederhana, kami mengimplementasikan program ini secara static. Algoritma dari kurva bezier baik menggunakan algoritma brute force maupun algoritma divide and conquer diimplementasikan dalam suatu folder bernama utils. Algoritma yang telah dibuat di dalam folder utils ini nantinya akan diimpor fungsi utamanya ke dalam halaman antarmuka pengguna (UI) dan memproses perhitungan berdasarkan state yang diperoleh dari *user*.

Proses implementasi algoritma secara static dengan antarmuka pengguna (UI) atau front end website memiliki beberapa kelemahan. Kelemahan paling utama dalam proses implementasi ini adalah keterbatasan kapasitas pemrosesan program. Program ini memiliki maksimal titik yang di generate seperti berada pada kisaran 100 hingga 150 titik.

2.1 ./app/page.jsx

File dengan format JavaScript Extension ini memiliki fungsi sebagai tampilan utama dari halaman situs program. Tampilan antarmuka dari program ini dapat diakses pada localhost:3000/ dimana tampilan home page dari program ini dapat menavigasi kita ke halaman ilustrasi kurva bezier dengan algoritma brute force maupun divide and conquer.

2.2 ./app/brute-force/page.jsx

Program ini merupakan aplikasi web yang dibangun dengan Next.js yang memiliki fungsi untuk membuat visualisasi kurva Bezier menggunakan algoritma brute force. Aplikasi ini mengintegrasikan komponen React seperti BruteforceClient untuk menampilkan grafik kurva, bruteforcealgorithms untuk perhitungan titik kurva, dan PointsInput untuk memungkinkan pengguna memasukkan dan mengelola titik-titik kurva. Program ini memanfaatkan library recharts untuk visualisasi grafik dan menggunakan React Hooks untuk manajemen state. Fitur lain termasuk kemampuan mengatur iterasi dan menampilkan waktu eksekusi algoritma. Berikut merupakan tabel yang berisi variabel dan method yang digunakan pada program.

Tabel 1 Method Table bruteforce page

Methods	Description
BruteforceClient	Ini adalah komponen React yang bertanggung jawab untuk menampilkan

	<p>grafik bezier curve menggunakan pustaka Recharts.</p> <p>Menerima prop points, yang merupakan array dari titik-titik bezier curve.</p>
bruteforcealgorithms:	<p>Ini adalah fungsi JavaScript yang menghitung titik-titik bezier curve menggunakan algoritma brute force.</p> <p>Menerima objek yang berisi points (titik-titik bezier), iteration (jumlah iterasi), dan setBezierPoint (fungsi untuk mengatur titik-titik bezier hasil perhitungan).</p>
PointsInput	<p>Ini adalah komponen React yang menangani input untuk titik-titik bezier dan jumlah iterasi.</p> <p>Menerima beberapa props, seperti mode (mode input), numPoints (jumlah titik awal), setInputPoints (fungsi untuk mengatur titik-titik input), numIterations (jumlah iterasi), dan setIteration (fungsi untuk mengatur jumlah iterasi).</p> <p>Memiliki fungsi handlePointChange yang menangani perubahan pada input titik-titik bezier.</p>
Bruteforce	<p>Ini adalah komponen utama yang menampilkan UI untuk aplikasi.</p> <p>Menggunakan beberapa state, seperti mode (mode input), inputPoints (titik-titik bezier input), bezierPoints (titik-titik bezier hasil perhitungan), iteration (jumlah iterasi), dan executionTime (waktu eksekusi).</p> <p>Mendefinisikan fungsi handleSubmit untuk menangani submit form input dan memanggil fungsi brute force untuk menghitung titik-titik bezier.</p> <p>Menggunakan beberapa efek samping (useEffect) untuk memantau perubahan pada</p>

	state tertentu, seperti inputPoints dan executionTime.
--	--

Tabel 2 Variable Table bruteforce page

Variables	Description
points	Variabel yang digunakan untuk menyimpan titik-titik bezier yang diinput oleh pengguna.
map	Variabel yang menyimpan titik-titik bezier dalam format yang sesuai dengan yang diterima oleh komponen <LineChart> dari pustaka Recharts.
sortedPoints	Variabel yang berisi salinan titik-titik bezier yang diurutkan berdasarkan nilai x.
numIterations	Variabel yang menentukan jumlah iterasi yang akan dilakukan dalam perhitungan algoritma brute force.
curvePoints	Variabel yang digunakan untuk menyimpan titik-titik bezier hasil perhitungan algoritma brute force.
jumlahTitik	Variabel state yang digunakan untuk menyimpan jumlah titik bezier yang akan dimasukkan oleh pengguna.
inputPoints	Variabel state yang digunakan untuk menyimpan titik-titik bezier yang dimasukkan oleh pengguna.
bezierPoints	Variabel state yang digunakan untuk menyimpan titik-titik bezier hasil perhitungan algoritma brute force.
iteration	Variabel state yang digunakan untuk menyimpan titik-titik bezier hasil perhitungan algoritma brute force.

executionTime	Variabel state yang digunakan untuk menyimpan waktu eksekusi perhitungan algoritma brute force.
timeStart	Variabel yang menyimpan waktu awal saat perhitungan dimulai.
timeEnd	Variabel yang menyimpan waktu saat perhitungan selesai.
updatedPoints	Variabel yang digunakan untuk menyimpan titik-titik bezier yang telah diperbarui setelah perubahan jumlah titik bezier.
index	Variabel yang digunakan untuk menyimpan indeks dalam loop atau pemetaan.
coord	Variabel yang digunakan untuk menyimpan koordinat (x atau y) dari titik bezier.
value	Variabel yang digunakan untuk menyimpan nilai baru dari input titik bezier.
mode	input (3-Points atau N-Points).
timeElapsed	Variabel yang digunakan untuk menyimpan selisih waktu antara timeEnd dan timeStart.

2.3 ./utils/bruteforce.js

Program ini mendefinisikan fungsi bruteForceBezierCurve, yang memiliki fungsi untuk menghitung posisi titik pada kurva Bezier berdasarkan t dan sekumpulan titik kontrol. Fungsi ini menggunakan formula matematis dari kurva Bezier, yang menggabungkan koefisien binomial dan kombinasi berbobot dari titik kontrol. Koefisien binomial dihitung menggunakan fungsi bantuan binomialCoefficient. Hasil akhir adalah koordinat titik pada kurva Bezier untuk nilai t tertentu.

Dalam mengimplementasikan strategi algoritma ini, hasil yang didapatkan cukup akurat dan memiliki kecepatan yang relatif tidak terlalu cepat jika dibandingkan algoritma divide and conquer. Algoritma ini memiliki fleksibilitas dimana algoritma ini dapat menghitung dan menentukan kurva bezier dengan jumlah titik yang beragam. Penggunaan koefisien binomial juga cukup membuat algoritma lebih efisien dikarenakan hal tersebut menghindari perhitungan yang berulang dengan memanfaatkan kesimetrisan koefisien binomial. Dalam hal ini, semakin besar titik iterasinya maka waktu memproses perhitungan juga semakin lama.

Algoritma brute force memiliki rumus yang berbeda dengan algoritma Divide and Conquer sehingga algoritma ini memiliki jumlah titik pada kurva bezier yang lebih sedikit karena rumus yang tidak eksponensial. Berikut merupakan tabel yang berisi method dan variabel dari file tersebut.

Tabel 3 Method Table utils bruteforce

Methods	Description
bruteForceBezierCurve(t, controlPoints)	Metode ini digunakan untuk menghitung titik pada kurva bezier menggunakan pendekatan brute force. Ini adalah implementasi langsung dari rumus bezier. Dengan parameter t: Parameter yang menentukan posisi pada kurva bezier. Biasanya berada dalam rentang 0 hingga 1.
controlPoints	Array dari titik kontrol bezier.

Tabel 4 Variable Table utils bruteforce

Methods	Description
t	Variabel ini merupakan parameter yang menentukan posisi pada kurva Bezier. Biasanya nilainya berada dalam rentang 0 hingga 1.
controlPoints	Variabel ini merupakan array dari titik kontrol Bezier. Setiap elemen dalam array merupakan objek yang memiliki properti x dan y, yang mewakili koordinat titik kontrol.
n	Variabel ini digunakan untuk menyimpan jumlah titik kontrol Bezier dikurangi satu (panjang array controlPoints dikurangi satu). Hal ini diperlukan untuk menghitung koefisien binomial dan melakukan iterasi melalui titik kontrol.
point	Variabel ini digunakan untuk menyimpan koordinat dari titik pada kurva Bezier yang dihasilkan.
binomialCoefficient:	Fungsi ini digunakan untuk menghitung koefisien binomial (n choose k) yang

	dibutuhkan dalam rumus Bezier. Fungsi ini menerima dua parameter, yaitu n dan k, dan mengembalikan nilai koefisien binomial.
x	Variabel ini digunakan untuk menyimpan nilai koordinat x dari titik kontrol saat iterasi.
y	Variabel ini digunakan untuk menyimpan nilai koordinat y dari titik kontrol saat iterasi.
binomialCoefficientValue:	Variabel ini digunakan untuk menyimpan nilai koefisien binomial saat iterasi. Ini diperlukan untuk mengalikan dengan komponen lain dari rumus Bezier.

2.4 ./app/div-n-conquer/page.jsx

Program ini adalah program yang digunakan untuk menjalankan web yang dibangun menggunakan Next.js dan React untuk memvisualisasikan kurva Bezier menggunakan algoritma "Divide and Conquer". Komponen utama termasuk DivideConquerClient untuk menampilkan grafik kurva, divideconqueralgorithms untuk perhitungan titik kurva, dan PointsInput yang memungkinkan pengguna untuk memasukkan dan mengelola titik-titik kurva. Aplikasi ini memanfaatkan library recharts untuk visualisasi grafik dan React Hooks untuk manajemen state. Berikut merupakan method dan variabel yang ada pada file ini :

Tabel 5 Method Table div-n-conquer page

Variable	Description
DivideConquerClient	Komponen ini digunakan untuk menampilkan kurva bezier menggunakan Recharts. Method DivideConquerClient mengembalikan elemen JSX yang mewakili LineChart dari Recharts. Komponen ini menerima props points dan inputPoints.
divideconqueralgorithms	Ini adalah fungsi yang menghitung titik-titik kurva bezier menggunakan algoritma divide and conquer. Method ini menerima objek yang berisi points (titik kontrol), iteration (jumlah iterasi), dan setBezierPoint (fungsi untuk mengatur titik bezier).

PointsInput	Komponen ini digunakan untuk mengatur input titik kontrol dan jumlah iterasi. Method PointsInput mengembalikan elemen JSX yang berisi input untuk mengatur titik kontrol dan jumlah iterasi. Komponen ini menerima props mode, numPoints, setInputPoints, setIteration, dan numIterations.
handleSubmit	Fungsi ini dipanggil saat tombol submit ditekan. Ini menghitung waktu eksekusi algoritma divide and conquer dan memanggil fungsi divideconqueralgorithms dengan input yang sesuai.
useEffect	Terdapat beberapa penggunaan hook useEffect untuk menangani perubahan state. Ini digunakan untuk memperbarui titik kontrol, mengatur titik bezier, dan mencatat perubahan input points.
useState	Terdapat beberapa penggunaan hook useState untuk menyimpan state seperti mode masukkan (3-Points atau N-Points), input points, titik bezier, jumlah iterasi, dan waktu eksekusi. Variabel ini digunakan untuk memperbarui dan menampilkan data pada aplikasi.

Tabel 6 Variable div-n-conquer page

Variable	Description
mode	Menyimpan mode input saat ini ('3-Points' atau 'N-Points').
inputPoints	Menyimpan titik-titik yang dimasukkan oleh pengguna.
bezierPoints	Menyimpan titik-titik hasil kurva bezier yang dihasilkan.

executionTime	Menyimpan waktu eksekusi untuk menghitung titik-titik bezier.
jumlahTitik	Menyimpan jumlah titik yang dimasukkan oleh pengguna.
points	Menyimpan titik-titik yang dimasukkan oleh pengguna.
updatedPoints	Menyimpan titik-titik yang diperbarui setelah perubahan jumlah titik.
sortedPoints	Menyimpan titik-titik yang diurutkan berdasarkan nilai x.
curvePoints	Menyimpan titik-titik hasil dari algoritma divide and conquer.
timeStart	Menyimpan waktu awal eksekusi perhitungan titik bezier.
timeEnd	Menyimpan waktu akhir eksekusi perhitungan titik bezier.
converted	Menyimpan titik-titik yang diubah formatnya untuk digunakan dalam grafik.
updatedPoints	Menyimpan titik-titik yang diperbarui setelah perubahan jumlah titik.
newArr	Menyimpan titik-titik baru yang dihasilkan setelah setiap iterasi dalam algoritma divide and conquer.

2.5 ./utils/divnconquer.jsx

Program yang ada pada file ini merupakan program yang menerapkan fungsi divncon yang merupakan algoritma divide and conquer untuk menghitung titik-titik pada kurva Bezier. Fungsi ini memecah kumpulan titik kontrol menjadi titik-titik tengah yang lebih kecil (menggunakan fungsi midpoint) dan secara rekursif membangun titik-titik kurva. Algoritma ini digunakan untuk secara efisien menghitung bentuk kurva Bezier dengan jumlah iterasi tertentu.

Program ini cukup efektif dalam memecahkan masalah menjadi sub-masalah yang lebih kecil. Algoritma ini lebih efisien bila dibandingkan dengan algoritma bruteforce. Algoritma ini menggunakan metode rekursif yang memungkinkan untuk penanganan kasus yang lebih kompleks dan titik kontrol yang lebih banyak. Terdapat juga fungsi midpoint yang bisa dibilang cukup efektif untuk menghitung titik tengah antar dua titik. Algoritma devide and conquer memiliki kompleksitas waktu yang lebih baik. Membagi masalah menjadi submasalah menjadikan masalah tersebut lebih mudah diselesaikan.

Algoritma ini memiliki hasil titik pada kurva bezier yang lebih banyak bila dibandingkan dengan Algoritma brute force. Hal tersebut bisa terjadi dikarenakan terdapat rumus pencarian divide and conquer yang selalu membagi dua sehingga jumlah titiknya juga merupakan perpangkatan dari 2. Dengan adanya rumusan tersebut, algoritma ini memiliki Berikut merupakan method dan variabel yang terdapat pada file program ini :

Tabel 7 Method utils divnconquer

Method	Description
midpoint	Ini adalah fungsi yang menghitung titik tengah antara dua titik yang diberikan. Variabel midpoint adalah sebuah fungsi dengan parameter p1 dan p2, yang mengembalikan sebuah objek berisi titik tengah dengan koordinat x dan y.
divncon	Ini adalah fungsi yang mengimplementasikan algoritma divide and conquer untuk menghitung titik bezier. Variabel divncon adalah sebuah fungsi dengan parameter arr, left, right, iter, iterations, dan setBezierPoint. Fungsi ini menggunakan rekursi untuk membagi titik-titik masukan menjadi subtitik dan menghitung titik bezier. Pada akhirnya, fungsi ini mengembalikan titik bezier dan memanggil fungsi setBezierPoint untuk mengatur titik bezier di dalam komponen React.

Tabel 8 Variable utils divnconquer

Variable	Description
arr	Array titik-titik yang akan diproses.
left	Array titik-titik pada sisi kiri.
right	Array titik-titik pada sisi kanan.
iter	Indeks iterasi saat ini.
iterations	Jumlah iterasi yang diinginkan.
setBezierPoint	Fungsi untuk mengatur titik bezier hasil.

BAB III

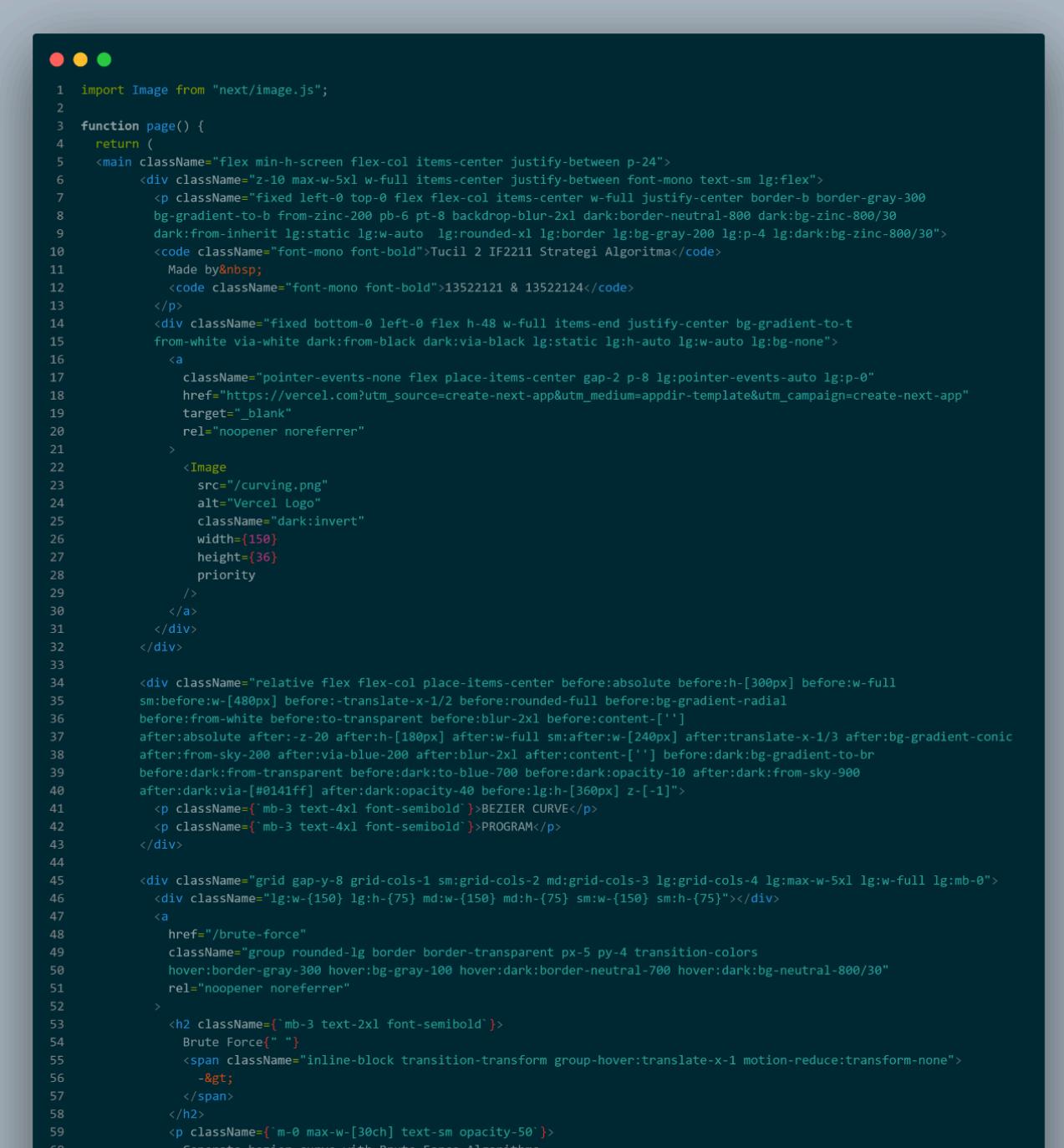
SOURCE CODE PROGRAM

3.1 Repository Program

Repository program dapat diakses melalui pranala *GitHub* berikut:
https://github.com/alandmpptma/Tucil2_13522121_13522124

3.2 Source Code Program

3.2.1. ./app/page.jsx



```
1 import Image from "next/image.js";
2
3 function page() {
4   return (
5     <main className="flex min-h-screen flex-col items-center justify-between p-24">
6       <div className="z-10 max-w-5xl w-full items-center justify-between font-mono text-sm lg:flex">
7         <p className="fixed left-0 top-0 flex flex-col items-center w-full justify-center border-b border-gray-300 bg-gradient-to-b from-zinc-200 pb-8 backdrop-blur-2xl dark:border-neutral-800 dark:bg-zinc-800/30 dark:from-inherit lg:static lg:w-auto lg:rounded-xl lg:border lg:gray-200 lg:p-4 lg:dark:bg-zinc-800/30">
8           <code className="font-mono font-bold">Tucil 2 IF2211 Strategi Algoritma</code>
9           Made by&nbsp;
10          <code className="font-mono font-bold">13522121 & 13522124</code>
11        </p>
12        <div className="fixed bottom-0 left-0 flex h-48 w-full items-end justify-center bg-gradient-to-t from-white via-white dark:from-black dark:via-black lg:static lg:h-auto lg:w-auto lg:bg-none">
13          <a
14            className="pointer-events-none flex place-items-center gap-2 p-8 lg:pointer-events-auto lg:p-0"
15            href="https://vercel.com?utm_source=create-next-app&utm_medium=appdir-template&utm_campaign=create-next-app"
16            target="_blank"
17            rel="noopener noreferrer"
18          >
19            <Image
20              src="/curving.png"
21              alt="Vercel Logo"
22              className="dark:invert"
23              width={150}
24              height={36}
25              priority
26            />
27          </a>
28        </div>
29      </div>
30
31      <div className="relative flex flex-col place-items-center before:absolute before:h-[300px] before:w-full sm:before:w-[480px] before:-translate-x-1/2 before:rounded-full before:bg-gradient-radial before:from-white before:to-transparent before:blur-2xl before:content-[''] after:absolute after:-z-20 after:h-[180px] after:w-full sm:after:w-[240px] after:translate-x-1/3 after:bg-gradient-conic after:from-sky-200 after:via-blue-200 after:blur-2xl after:content-[''] before:dark:gradient-to-br before:dark:from-transparent before:dark:to-blue-700 before:dark:opacity-10 after:dark:from-sky-900 after:dark:via-[#0141ff] after:dark:opacity-40 before:lg:h-[360px] z-[-1]">
32        <p className={`mb-3 text-4xl font-semibold`}>BEZIER CURVE</p>
33        <p className={`mb-3 text-4xl font-semibold`}>PROGRAM</p>
34      </div>
35
36      <div className="grid gap-y-8 grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 lg:max-w-5xl lg:w-full lg:mb-0">
37        <div className="lg:w-[150] lg:h-[75] md:w-[75] sm:w-[150] sm:h-[75]"></div>
38        <a
39          href="/brute-force"
40          className="group rounded-lg border border-transparent px-5 py-4 transition-colors hover:border-gray-300 hover:bg-gray-100 hover:dark:border-neutral-700 hover:dark:bg-neutral-800/30"
41          rel="noopener noreferrer"
42        >
43          <h2 className={`mb-3 text-2xl font-semibold`}>
44            Brute Force<" ">
45            <span className="inline-block transition-transform group-hover:translate-x-1 motion-reduce:transform-none">
46              -&gt;
47            </span>
48          </h2>
49          <p className={`m-0 max-w-[30ch] text-sm opacity-50`}>
50            Generate bezier curve with Brute Force Algorithms.
51          </p>
52        </a>
53      </div>
54    </div>
55  </main>
56</div>
```

```

61         </p>
62     </a>
63
64     <a href="/div-n-conquer"
65         className="group rounded-lg border border-transparent px-5 py-4 transition-colors
66         hover:border-gray-300 hover:bg-gray-100 hover:dark:border-neutral-700 hover:dark:bg-neutral-800/30"
67         rel="noopener noreferrer"
68         style={{ width: '275px' }}>
69     >
70         <h2 className="mb-3 text-2xl font-semibold">
71             Divide & Conquer{" "}
72             <span className="inline-block transition-transform group-hover:translate-x-1 motion-reduce:transform-none">
73                 -&gt;;
74             </span>
75         </h2>
76         <p className="m-0 max-w-[30ch] text-sm opacity-50 text-balanced">
77             Generate bezier curve with Divide & Conquer Algorithms.
78         </p>
79     </a>
80     <div className="lg:w-[150] lg:h-[75] md:w-[150] md:h-[75] sm:w-[150] sm:h-[75]"></div>
81 </div>
82 </main>
83 );
84 }
85 }
86
87 export default page;
88

```

3.2.2. ./app/brute-force/page.jsx

```

1 "use client"
2 import React, { useState, useEffect } from 'react'; // Tambahkan impor useEffect
3 import { LineChart, Line, CartesianGrid, XAxis, YAxis, Tooltip, Legend } from 'recharts';
4 import Image from "next/image";
5 import { bruteForceBezierCurve } from '../../../../../utils/bruteforce';
6
7 function BruteforceClient({points, inputPoints}) {
8     // passing data titik digunakan sebagai state
9     // Mengurutkan titik berdasarkan nilai karena pustaka ini tidak melakukan pengurutan dalam penampilan kurvanya
10    const map = points.map(([x, y]) => ({ x, y }));
11    const sortedInput = inputPoints.slice().sort((a, b) => a.x - b.x);
12    console.log("Masukan titik kontrol :");
13    console.log(sortedInput);
14    console.log("Hasil bezier Points dengan Algoritma Brute Force :");
15    console.log(map);
16    // Chart component
17
18    return (
19        <LineChart width={600} height={600} >
20            <CartesianGrid strokeDasharray="3 3" />
21                <XAxis type="number" dataKey="x" domain={[ 'auto', 'auto' ]} allowDataOverflow />
22                <YAxis type="number" domain={[ 'auto', 'auto' ]} allowDataOverflow />
23                <Tooltip formatter={({value, name, props}) => [value, props.payload.x]} />
24                <Legend />
25                <Line type="linear" dataKey="y" name="Bezier Curve" stroke="#8884d8" data={map} dot={{ fill: 'blue' }} />
26                <Line type="linear" dataKey="y" name="Control Points" stroke="#82ca9d" data={sortedInput} dot={{ fill: 'green' }} />
27        </LineChart>
28    );
29 }
30
31 function bruteforcealgorithms({ points, iteration, setBezierPoint }) {
32     // Menghitung titik dengan brute force
33     const calculateBezierPoints = () => {
34         const sortedPoints = points.slice().sort((a, b) => a.x - b.x);
35         const numIterations = parseInt(iteration) * (points.length-1); // Pastikan numIterations berupa angka
36         const curvePoints = [];
37         for (let i = 0; i < numIterations+1; i++) {
38             const t = i / (numIterations);
39             curvePoints.push(bruteForceBezierCurve(t, sortedPoints));
40         }
41     }
42 }

```

```

1      setBezierPoint(curvePoints);
2  };
3
4 // Panggil calculateBezierPoints Langsung
5 calculateBezierPoints();
6 }
7
8 function PointsInput({mode, numPoints, setInputPoints, setIteration, numIterations}) {
9   // Inisialisasi titik dengan objek kosong
10  const [jumlahTitik, setJumlahTitik] = useState(numPoints)
11  const [points, setPoints] = useState(
12    Array.from({ length: jumlahTitik }, () => ({ x: '', y: '' }))
13  );
14
15 useEffect(() => {
16   // Membuat array baru dengan panjang baru
17   const updatedPoints = Array.from({ length: jumlahTitik }, (_, index) => {
18     // Kalau indeks masih memiliki panjang yang sama dengan array, gunakan titik yang sudah ada
19     if (index < points.length) {
20       return points[index];
21     } else {
22       // Jika tidak, buat titik baru dengan nilai kosong
23       return { x: '', y: '' };
24     }
25   });
26   // Buat array point baru
27   setPoints(updatedPoints);
28 }, [jumlahTitik]);
29
30 useEffect(() => {
31   setInputPoints(points);
32   console.log(points);
33   console.log("Points Updated");
34 }, [points]);
35
36 // Menangani perubahan point pada input
37 const handlePointChange = (index, coord, value) => {
38   const updatedPoints = points.map((point, i) =>
39     i === index ? { ...point, [coord]: value } : point
40   );
41   setPoints(updatedPoints);
42   // onPointsChange(updatedPoints); // Kirim balik titik yang sudah di perbarui ke component utama
43 };
44
45 // Form input untuk titik baik untuk N points maupun 3 titik
46 return (
47   <div>
48     {mode === 'N-Points' && (
49       <div className='ml-4 space-x-4 space-y-2'>
50         <label>Number of Desired Points:</label>
51         <input
52           type="number"
53           className="bg-white text-black p-1 rounded border border-gray-300 w-[75px]"
54           value={jumlahTitik}
55           onChange={(e) => setJumlahTitik(Number(e.target.value))}>
56         />
57       </div>
58     )}
59     <div className='flex flex-row justify-center space-x-10 translate-x-[15px] font-semibold'>
60       <p>sb-X</p>
61       <p>sb-Y</p>
62     </div>
63     {points.map((point, index) => (
64
65       <div key={index} className='ml-4 space-x-4 space-y-2'>
66         <label>` Point ${index + 1}` </label>
67         <input
68           type="number"
69           className="bg-white text-black p-1 rounded border border-gray-300 w-[75px]"
70           value={point.x}
71           onChange={(e) => handlePointChange(index, 'x', Number(e.target.value))}>
72         />
73         <input
74           type="number"
75           className="bg-white text-black p-1 rounded border border-gray-300 w-[75px]">
76       </div>
77     ))}
78   </div>
79 }
80
81 <script>
82   window.addEventListener('load', () => {
83     const canvas = document.getElementById('curveCanvas');
84     const context = canvas.getContext('2d');
85     const curvePoints = [
86       { x: 100, y: 100 },
87       { x: 200, y: 150 },
88       { x: 300, y: 200 },
89       { x: 400, y: 250 },
90       { x: 500, y: 300 },
91       { x: 600, y: 350 },
92       { x: 700, y: 400 },
93       { x: 800, y: 450 },
94       { x: 900, y: 500 }
95     ];
96     const curve = new BezierCurve(context, curvePoints);
97     curve.draw();
98   });
99 </script>
100
```

```

1      value={point.y}
2      onChange={(e) => handlePointChange(index, 'y', Number(e.target.value))};
3    />
4
5    </div>
6  )));
7  /* Input untuk jumlah iterasi */
8  <div className='m1-4 space-x-4 space-y-2'>
9    <label>Number of Iterations:</label>
10   <input type="number" className="bg-white text-black p-1 rounded border border-gray-300 w-[75px]"
11     value={numIterations} onChange={(e) => setIteration(Number(e.target.value))}/>
12  </div>
13  /* Input jumlah poin yang diinginkan (hanya untuk mode 'N-Points') */
14  </div>
15 );
16 }
17
18 export default function BruteForce() {
19  const [mode, setMode] = useState('3-Points'); // Kontrol mode masukkan
20  const [inputPoints, setInputPoints] = useState([]); // menyimpanan inputan titik dari form
21  const [bezierPoints, setBezierPoints] = useState([]);
22  const [iteration, setIteration] = useState();
23  const [executionTime, setExecutionTime] = useState();
24  const [input, setInput] = useState([]);
25
26  // Fungsi yang mengatasi inputan dari formulir ketika tombol submit diclick
27  const handleSubmit = (event) => {
28    event.preventDefault(); // Mengupdate input points
29    setInput(inputPoints);
30    const timeStart = performance.now();
31    bruteForcealgorithms({ points: inputPoints, iteration: iteration, setBezierPoint: setBezierPoints });
32    const timeEnd = performance.now();
33    const timeElapsed = timeEnd - timeStart;
34    setExecutionTime(timeElapsed);
35    console.log(timeElapsed);
36  };
37 };
38
39 useEffect(() => {
40  // Melakukan sesuatu saat ada perubahan inputan
41  console.log("Input points changed:", inputPoints);
42  console.log(inputPoints);
43 }, [inputPoints]);
44
45 return (
46  <div className="flex min-h-screen flex-col items-center justify-between p-24">
47    <div className="z-10 max-w-5xl w-full items-center justify-between font-mono text-sm lg:flex">
48      <p className="fixed left-0 top-0 flex w-full justify-center border-b border-gray-300 bg-gradient-to-b
49        from-zinc-200 pb-6 pt-8 backdrop-blur-2xl dark:border-neutral-800 dark:bg-zinc-800/30
50        dark:from-inherit lg:static lg:w-auto lg:rounded-xl lg:border lg:gray-200 lg:p-4 lg:dark:bg-zinc-800/30">
51        <code className="font-mono font-bold">Brute Force Algorithms</code>
52      </p>
53      <div className="fixed bottom-0 left-0 flex h-48 w-full items-end justify-center bg-gradient-to-t
54        from-white via-white dark:from-black dark:via-black lg:static lg:h-auto lg:w-auto lg:bg-none">
55        <a className="pointer-events-none flex place-items-center gap-2 p-8 lg:pointer-events-auto lg:p-0">
56          <Image
57            src="/curving.png"
58            alt="Vercel Logo"
59            className="dark:invert"
60            width={150}
61            height={36}
62            priority
63          />
64        </a>
65      </div>
66    </div>
67    <div className="relative flex lg:flex-row md:flex-col sm:flex-col place-items-center before:absolute before:h-[300px]
68      before:w-full sm:before:w-[100px] before:-translate-x-1/2 before:rounded-full before:bg-gradient-radial
69      before:from-white before:to-transparent before:blur-2xl before:content-[''] after:absolute after:-z-20
70      after:h-[180px] after:w-full sm:after:w-[80px] after:translate-x-1/3 after:bg-gradient-conic after:from-sky-200
71      after:via-blue-200 after:blur-2xl after:content-[''] before:dark:gradient-to-br before:dark:from-transparent
72      before:dark:to-blue-700 before:dark:opacity-10 after:dark:from-sky-900 after:dark:via-[#0141ff]
73      after:dark:opacity-40 before:lg:h-[360px] z-[1]">
74      <div className="m-[25px] h-[700px] border-b border-gray-300 bg-gradient-to-b from-zinc-200 pb-6 pt-8 backdrop-blur-2xl
75      dark:border-neutral-800 dark:bg-zinc-800/30 dark:from-inherit lg:static lg:w-[700px] lg:rounded-xl lg:border
76      lg:gray-200 lg:p-4 lg:dark:bg-zinc-800/30">

```

```

1      <p className="mb-3 text-xl font-semibold">Bezier Curve Illustration</p>
2      <BruteforceClient points={bezierPoints} inputPoints={input}/>
3    </div>
4    <div className="m-[25px] z-50 border-b border-gray-300 bg-gradient-to-b from-zinc-200 pb-6 pt-8 backdrop-blur-2xl
5 dark:border-neutral-800 dark:bg-zinc-800/30 dark:from-inherit lg:w-[400px] lg:rounded-xl lg:border
6 lg:bg-gray-200 lg:p-4 lg:dark:bg-zinc-800/30 h-fit">
7      <div className="m-4 p-4">
8        <h2 className="text-xl font-semibold">Input Points</h2>
9        <form onSubmit={handleSubmit}>
10          <div className="m-4 flex items-center space-x-4 mt-[20px]">
11            <button type="button" onClick={() => setMode('3-Points')} className={mode === '3-Points' ? 'font-bold border-b border-gray-300' : ''}>3-Points</button>
12            <button type="button" onClick={() => setMode('N-Points')} className={mode === 'N-Points' ? 'font-bold border-b border-gray-300' : ''}>N-Points</button>
13          </div>
14        </div>
15        <div>
16          {mode == '3-Points' && <PointsInput mode={mode} numPoints={3} setInputPoints={setInputPoints} numIterations={iteration} setIteration={setIteration} />}
17          {mode == 'N-Points' && <PointsInput mode={mode} numPoints={5} setInputPoints={setInputPoints} numIterations={iteration} setIteration={setIteration} />}
18          <p className="mt-[16px]">Time Elapsed: {executionTime} ms</p>
19        </div>
20        <button type="submit" className="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded mt-[35px]">
21          Submit
22        </button>
23      </form>
24    </div>
25  </div>
26</div>
27
28<div className="grid gap-y-8 grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 lg:max-w-5xl lg:w-full lg:mb-0">
29  <div className="lg:w-[150] lg:h-[75] md:w-[150] md:h-[75] sm:w-[150] sm:h-[75]"></div>
30  <a href="/">
31    <span className="group rounded-lg border border-transparent px-5 py-4 transition-colors hover:border-gray-300
32      hover:bg-gray-100 hover:dark:border-neutral-700 hover:dark:bg-neutral-800/30">
33      <h2 className="mb-3 text-2xl font-semibold">
34        Home(" ")
35        <span className="inline-block transition-transform group-hover:translate-x-1 motion-reduce:transform-none">
36          -&gt;
37        </span>
38      </h2>
39      <p className=" m-0 max-w-[30ch] text-sm opacity-50 ">
40        The front page or main page of this project.
41      </p>
42    </a>
43
44  <a href="/div-n-conquer">
45    <span className="group rounded-lg border border-transparent px-5 py-4 transition-colors hover:border-gray-300
46      hover:bg-gray-100 hover:dark:border-neutral-700 hover:dark:bg-neutral-800/30"
47      rel="noopener noreferrer"
48      style={{ width: '275px' }}>
49      <h2 className="mb-3 text-2xl font-semibold">
50        Divide & Conquer(" ")
51        <span className="inline-block transition-transform group-hover:translate-x-1 motion-reduce:transform-none">
52          -&gt;
53        </span>
54      </h2>
55      <p className=" m-0 max-w-[30ch] text-sm opacity-50 text-balanced">
56        Generate bezier curve with Divide & Conquer Algorithms.
57      </p>
58    </a>
59    <div className="lg:w-[150] lg:h-[75] md:w-[150] md:h-[75] sm:w-[150] sm:h-[75]"></div>
60  </div>
61</div>
62</div>
63</div>
64</div>
65</div>
66</div>
67</div>
68</div>

```

3.2.3. ./utils/bruteforce.js

```
1 // utils/bruteforce.js
2
3 export const bruteForceBezierCurve = (t, controlPoints) => {
4     // Menghitung titik yang ada pada kurva bezier menggunakan pendekatan brute force
5     const n = controlPoints.length - 1;
6     console.log("controlPoints");
7     console.log(controlPoints[0]);
8     let point = [0, 0];
9     const binomialCoefficient = (n, k) => {
10         if (k < 0 || k > n) return 0;
11         if (k === 0 || k === n) return 1;
12         k = Math.min(k, n - k);
13         let c = 1;
14         for (let i = 0; i < k; i++) {
15             c = c * (n - i) / (i + 1);
16         }
17         return c;
18     };
19     for (let i = 0; i <= n; i++) {
20         const { x, y } = controlPoints[i];
21         console.log("Nilai x", x);
22         console.log("Nilai y", y);
23         const binomialCoefficientValue = binomialCoefficient(n, i);
24         point[0] += binomialCoefficientValue * (t ** i) * ((1 - t) ** (n - i)) * x;
25         point[1] += binomialCoefficientValue * (t ** i) * ((1 - t) ** (n - i)) * y;
26     }
27     console.log("Bezier");
28     console.log(point);
29     return point;
30 };
31
```

3.2.4. ./app/div-n-conquer/page.jsx

```
 1 "use client"
 2 import React, { useState, useEffect } from 'react'; // Tambahkan import useEffect
 3 import { LineChart, Line, CartesianGrid, XAxis, YAxis, Tooltip, Legend } from 'recharts';
 4 import Image from "next/image";
 5 import { divncon } from '../../../../../utils/divnconquer';
 6
 7 function DivideConquerClient({points, inputPoints}) {
 8   // passing data titik digunakan sebagai state
 9   // Mengurutkan titik berdasarkan nilai karena pustaka ini tidak melakukan pengurutan dalam penampilan kurvanya
10   // Chart component
11   const converted = points && points.length > 0
12     ? points[0].map(item => {{ x: item.x, y: item.y }})
13     : [];
14   const sortedInput = inputPoints && inputPoints.length
15     ? inputPoints.slice().sort((a, b) => a.x - b.x)
16     : [];
17
18   if (sortedInput.length > 0) {
19     converted.push(sortedInput[0]); //titik awal
20     converted.push(sortedInput[sortedInput.length - 1]); //titik akhir
21   }
22   const sortedPoints = converted.length
23     ? converted.slice().sort((a, b) => a.x - b.x)
24     : [];
25   console.log("Masukan titik kontrol : ");
26   console.log(sortedInput);
27   console.log("Hasil bezier Points dengan Algoritma Divide & Conquer : ");
28   console.log(converted);
29
30   return (
31     <LineChart width={600} height={600} >
32       <CartesianGrid strokeDasharray="3 3" />
33         <XAxis type="number" dataKey="x" domain={['auto', 'auto']} allowDataOverflow />
34         <YAxis type="number" domain={['auto', 'auto']} allowDataOverflow />
35         <Legend />
36         <Line type="linear" dataKey="y" name="Bezier Curve" stroke="#8884d8" data={sortedPoints} dot={{ fill: 'blue' }} />
37         <Line type="linear" dataKey="y" name="Control Points" stroke="#82ca9d" data={sortedInput} dot={{ fill: 'green' }} />
38         <Tooltip formatter={(value, name, props) => [value, props.payload.x]} />
39
40     </LineChart>
41   );
42 }
43
44 function divideconqueralgorithms({ points, iteration, setBezierPoint }) {
45   // Menghitung titik dengan brute force
46   const calculateBezierPoints = () => {
47     const numIterations = parseInt(iteration); // Pastikan numIterations berupa angka
48     const curvePoints = [];
49     const sorted = points.slice().sort((a, b) => a.x - b.x);
50     curvePoints.push(divncon(sorted, [], [], 0, numIterations, setBezierPoint));
51
52     setBezierPoint(curvePoints);
53   };
54
55   // Panggil calculateBezierPoints langsung
56   calculateBezierPoints();
57 }
58
59 function PointsInput({mode, numPoints, setInputPoints, setIteration, numIterations}) {
60   // Inisialisasi titik dengan objek kosong
61   const [jumlahTitik, setJumlahTitik] = useState(numPoints)
62   const [points, setPoints] = useState(
63     Array.from({ length: jumlahTitik }, () => ({ x: '', y: '' }));
64   );
65
66   useEffect(() => {
67     // Membuat array baru dengan panjang baru
68     const updatedPoints = Array.from({ length: jumlahTitik }, (_, index) => {
69       // Kalau indeks masih memiliki panjang yang sama dengan array, gunakan titik yang sudah ada
70       if (index < points.length) {
71         return points[index];
72       } else {
73         // Jika tidak, buat titik baru dengan nilai kosong
74         return { x: '', y: '' };
75       }
76     });
77     // Buat array point baru
78     setPoints(updatedPoints);
79   }, [jumlahTitik]);
```

```

1  useEffect(() => {
2    setInputPoints(points);
3    console.log(points);
4    console.log("Points Updated");
5  }, [points]);
6
7 // Menangani perubahan point pada input
8 const handlePointChange = (index, coord, value) => {
9  const updatedPoints = points.map((point, i) =>
10   i === index ? { ...point, [coord]: value } : point
11 );
12  setPoints(updatedPoints);
13  // onPointsChange(updatedPoints); // Kirim balik titik yang sudah di perbarui ke component utama
14};
15
16 // Form input untuk titik baik untuk N points maupun 3 titik
17 return (
18  <div>
19   {mode === 'N-Points' && (
20    <div className='ml-4 space-x-4 space-y-2'>
21      <label>Number of Desired Points:</label>
22      <input
23        type="number"
24        className="bg-white text-black p-1 rounded border border-gray-300 w-[75px]"
25        value={jumlahTitik}
26        onChange={(e) => setJumlahTitik(Number(e.target.value))}>
27      />
28    </div>
29  )}
30  <div className='flex flex-row justify-center space-x-10 translate-x-[15px] font-semibold'>
31    <p>sb-X</p>
32    <p>sb-Y</p>
33  </div>
34  {points.map((point, index) => (
35
36   <div key={index} className='ml-4 space-x-4 space-y-2'>
37     <label> Point ${index + 1}</label>
38     <input
39       type="number"
40       className="bg-white text-black p-1 rounded border border-gray-300 w-[75px]"
41       value={point.x}
42       onChange={(e) => handlePointChange(index, 'x', Number(e.target.value))}>
43     />
44     <input
45       type="number"
46       className="bg-white text-black p-1 rounded border border-gray-300 w-[75px]"
47       value={point.y}
48       onChange={(e) => handlePointChange(index, 'y', Number(e.target.value))}>
49     />
50   </div>
51  ))}
52  /* Input untuk jumlah iterasi */
53  <div className='ml-4 space-x-4 space-y-2'>
54    <label>Number of Iterations:</label>
55    <input type="number" className="bg-white text-black p-1 rounded border border-gray-300 w-[75px]"
56    value={numIterations} onChange={(e) => setIteration(Number(e.target.value))}/>
57  </div>
58  /* Input jumlah poin yang diinginkan (hanya untuk mode 'N-Points') */
59  </div>
60 );
61 );
62 }
63
64 export default function DivideConquer() {
65  const [mode, setMode] = useState('3-Points'); // Kontrol mode masukkan
66  const [inputPoints, setInputPoints] = useState([]); //Menyimpan inputan
67  const [bezierPoints, setBezierPoints] = useState([]);
68  const [iteration, setIteration] = useState();
69  const [executionTime, setExecutionTime] = useState();
70
71 // Fungsi yang mengatasi inputan
72  const handleSubmit = (event) => {
73    event.preventDefault(); // Mengupdate input points
74    const timeStart = performance.now();
75    divideconqueralgorithms({ points: inputPoints, iteration: iteration, setBezierPoint: setBezierPoints });
76    const timeEnd = performance.now();
77    const timeElapsed = timeEnd - timeStart;
78    setExecutionTime(timeElapsed);
79  };
80}

```

```

1  useEffect(() => {
2    // Melakukan sesuatu saat ada perubahan inputan
3    console.log("Input points changed:", inputPoints);
4    console.log(inputPoints);
5  }, [inputPoints]);
6
7  return (
8    <div className="flex min-h-screen flex-col items-center justify-between p-24">
9      <div className="z-10 max-w-5xl w-full items-center justify-between font-mono text-sm lg:flex">
10        <p className="fixed left-0 top-0 flex w-full justify-center border-b border-gray-300 bg-gradient-to-b
from-zinc-200 pb-6 pt-8 backdrop-blur-2xl dark:border-neutral-800 dark:bg-zinc-800/30 dark:from-inherit
lg:static lg:w-auto lg:rounded-xl lg:border lg:bg-gray-200 lg:p-4 lg:dark:bg-zinc-800/30">
11          <code className="font-mono font-bold">Divide and Conquer Algorithms</code>
12        </p>
13        <div className="fixed bottom-0 left-0 flex h-48 w-full items-end justify-center bg-gradient-to-t from-white
via-white dark:from-black dark:via-black lg:static lg:h-auto lg:w-auto lg:bg-none">
14          <a className="pointer-events-none flex place-items-center gap-2 p-8 lg:pointer-events-auto lg:p-0">
15            <Image
16              src="/curving.png"
17              alt="Vercel Logo"
18              className="dark:invert"
19              width={150}
20              height={36}
21              priority
22            />
23          </a>
24        </div>
25      </div>
26      <div className="relative flex lg:flex-row md:flex-col sm:flex-col place-items-center before:absolute before:h-[300px]
before:w-full sm:before:w-[1000px] before:-translate-x-1/2 before:rounded-full before:bg-gradient-radial before:from-white
before:to-transparent before:blur-2xl before:content[''] after:absolute after:-z-20 after:h-[180px] after:w-full
sm:after:w-[800px] after:translate-x-1/3 after:bg-gradient-conic after:from-sky-200 after:via-blue-200 after:blur-2xl
after:content[''] before:dark:bg-gradient-to-bn before:dark:from-transparent before:dark:to-blue-700
before:dark:opacity-10 after:dark:from-sky-900 after:dark:via-[#0141ff] after:dark:opacity-40 before:lg:h-[360px] z-[1]">
27        <div className="m-[25px] h-[700px] border-b border-gray-300 bg-gradient-to-b from-zinc-200 pb-6 pt-8 backdrop-blur-2xl
dark:border-neutral-800 dark:bg-zinc-800/30 dark:from-inherit lg:static lg:w-[700px] lg:rounded-xl lg:border
lg:bg-gray-200 lg:p-4 lg:dark:bg-zinc-800/30">
28          <p className="mb-3 text-xl font-semibold">Bezier Curve Illustration</p>
29          {bezierPoints.length > 0 && <div>ConquerClient points={bezierPoints} inputPoints={inputPoints}/>}
30        </div>
31        <div className="m-[25px] z-50 border-b border-gray-300 bg-gradient-to-b from-zinc-200 pb-6 pt-8 backdrop-blur-2xl
dark:border-neutral-800 dark:bg-zinc-800/30 dark:from-inherit lg:static lg:w-[400px] lg:rounded-xl lg:border
lg:bg-gray-200 lg:p-4 lg:dark:bg-zinc-800/30 h-fit">
32          <div className="m-4 p-4">
33            <h2>Input Points</h2>
34            <form onSubmit={handleSubmit}>
35              <div>
36                <button type="button" onClick={() => setMode('3-Points')} className={mode === '3-Points' ? 'font-bold border-b border-gray-300' : ''}>3-Points</button>
37                <button type="button" onClick={() => setMode('N-Points')} className={mode === 'N-Points' ? 'font-bold border-b border-gray-300' : ''}>N-Points</button>
38              </div>
39            </form>
40            {mode === '3-Points' && <PointsInput mode={mode} numPoints={3} setInputPoints={setInputPoints} numIterations={iteration} setIteration={setIteration} />}
41            {mode === 'N-Points' && <PointsInput mode={mode} numPoints={5} setInputPoints={setInputPoints} numIterations={iteration} setIteration={setIteration} />}
42            <p>Time Elapsed: {executionTime} ms</p>
43          </div>
44          <button type="submit" className="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded mt-[35px]">
45            Submit
46          </button>
47        </div>
48      </div>
49    </div>
50  </div>
51
52
53  <div className="grid gap-y-8 grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 lg:max-w-5xl lg:w-full lg:mb-0">
54    <div className="lg:w-[150] lg:h-[75] md:w-[150] md:h-[75] sm:w-[150] sm:h-[75]"></div>
55    <a href="/brute-force">
56      <span>Brute Force<br/>
57      <span>Generate bezier curve with Brute Force Algorithms.<br/>
58    </span>
59  </a>
60</div>

```

```
1      <a
2        href="/"
3        className="group rounded-lg border border-transparent px-5 py-4 transition-colors hover:border-gray-300
4        hover:bg-gray-100 hover:dark:border-neutral-700 hover:dark:bg-neutral-800/30"
5        rel="noopener noreferrer"
6        style={{ width: '275px' }}
7      >
8        <h2 className="mb-3 text-2xl font-semibold">
9          Home{" "}
10         <span className="inline-block transition-transform group-hover:translate-x-1 motion-reduce:transform-none">
11           -&gt;
12         </span>
13       </h2>
14       <p className={`m-0 max-w-[30ch] text-sm opacity-50 text-balanced`}>
15         The front page or main page of this project.
16       </p>
17     </a>
18     <div className="lg:w-[150] lg:h-[75] md:w-[150] md:h-[75] sm:w-[150] sm:h-[75]"></div>
19   </div>
20 </div>
21 );
22 }
23 }
```

3.2.5. ./utils/divnconquer.jsx

```
1 // utils/divnconquer.js
2
3
4 // Fungsi untuk melakukan kalkulasi modpoint atau titik tengah
5 const midpoint = (p1, p2) => ({
6   x: (p1.x + p2.x) / 2,
7   y: (p1.y + p2.y) / 2
8 });
9
10 // Fungsi untuk melakukan reduksi titik menggunakan algoritma divide and conquer
11 export const divncon = (arr, left, right, iter, iterations, setBezierPoint) => {
12   let leftArr = [arr[0]];
13   let rightArr = [arr[arr.length - 1]];
14
15   while (arr.length > 1) {
16     let newArr = [];
17     for (let i = 0; i < arr.length - 1; i++) {
18       newArr.push(midpoint(arr[i], arr[i + 1]));
19     }
20
21     leftArr = leftArr.concat([newArr[0]]);
22     rightArr = [newArr[newArr.length - 1]].concat(rightArr);
23     arr = newArr;
24   }
25
26   if (iter === iterations - 1) {
27     setBezierPoint(arr);
28     return arr;
29   } else {
30     iter += 1;
31     return divncon(
32       leftArr.concat(arr),
33       left,
34       arr,
35       iter,
36       iterations,
37       setBezierPoint
38     ).concat(arr).concat(
39       divncon(arr.concat(rightArr), [], right, iter, iterations, setBezierPoint)
40     );
41   }
42};
```

BAB IV

UJI COBA PROGRAM

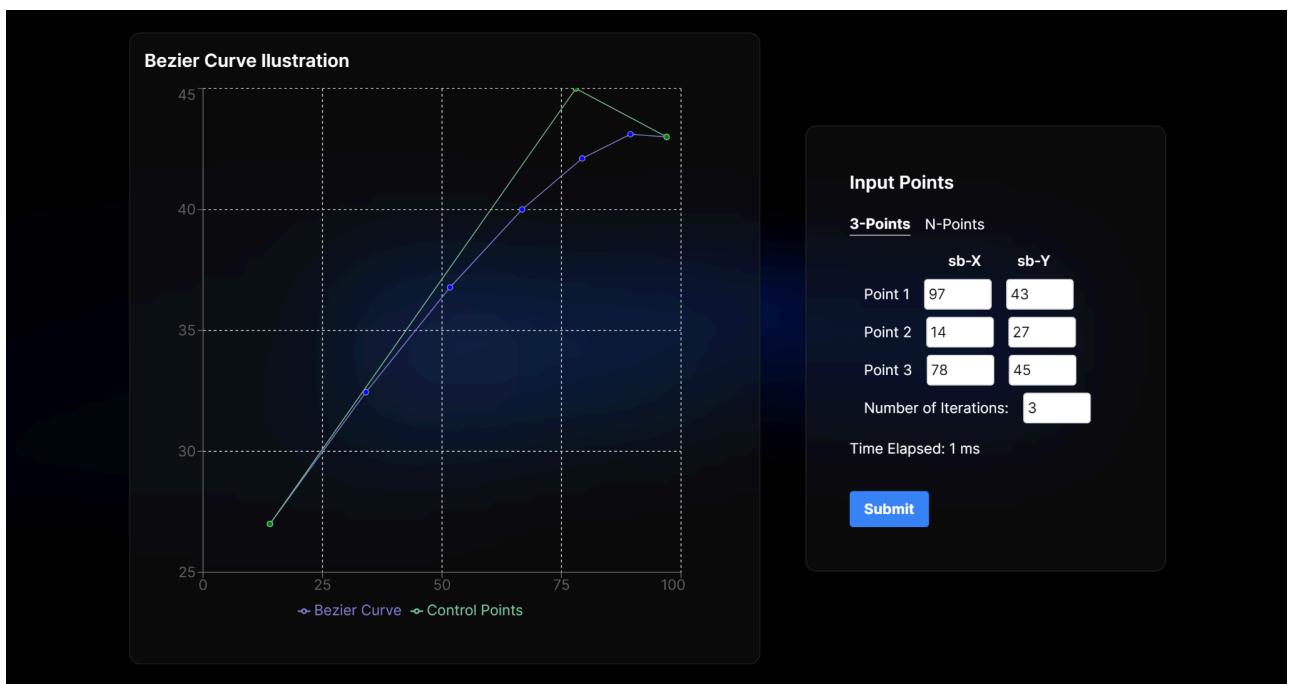
Pada uji coba program kurva bezier, kami akan melakukan uji coba pada spesifikasi wajib terlebih dahulu yaitu kurva bezier dengan masukan tiga (3) titik kontrol. Kami akan melakukan 6 set percobaan yang kami lakukan terhadap program kami yang akan terdiri dari pengujian kurva bezier dengan algoritma brute force dan algoritma divide and conquer. Kami akan melakukan pengujian untuk 3 dan 5 iterasi untuk masing-masing algoritma dalam setiap set percobaan. Berikut adalah 6 set titik yang akan kami gunakan dalam uji coba program kurva bezier:

- Set 1: (97, 43), (14, 27), (78, 45)
- Set 2: (67, 100), (76, 60), (14, 62)
- Set 3: (91, 63), (39, 19), (37, 165)
- Set 4: (-49, -44), (86, 40), (59, 59)
- Set 5: (4, 87), (-18, 25), (-26, 29)
- Set 6: (-52, 46), (84, 99), (1, -63)

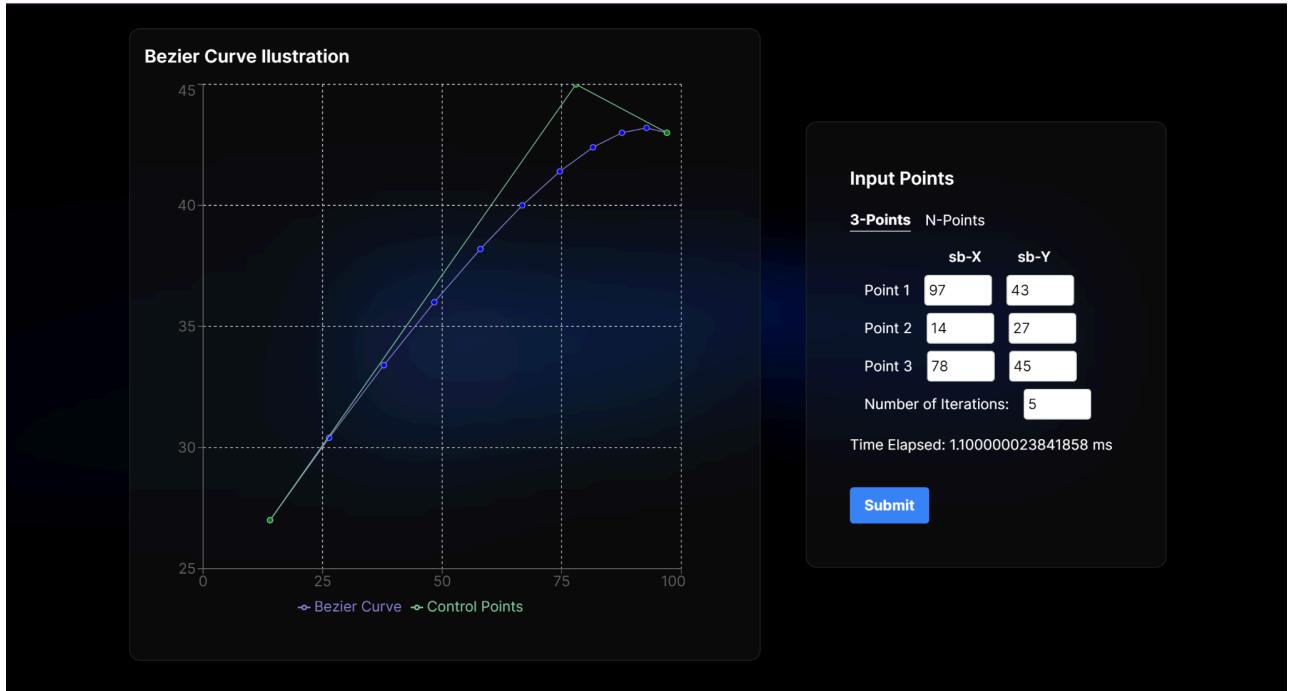
Setelah melakukan uji coba program kurva bezier untuk spesifikasi wajib akan dilanjutkan dengan uji coba spesifikasi bonus. Spesifikasi bonus ini merupakan program kurva bezier yang dapat menerima n titik kontrol. Pada uji coba kali ini kami akan membatasi untuk melakukan uji coba pada kurva bezier kubik (empat titik kontrol) dan kurva bezier kuartik (lima titik kontrol).

4.1 Set Percobaan Pertama

- a) Algoritma Brute Force
3 iterasi

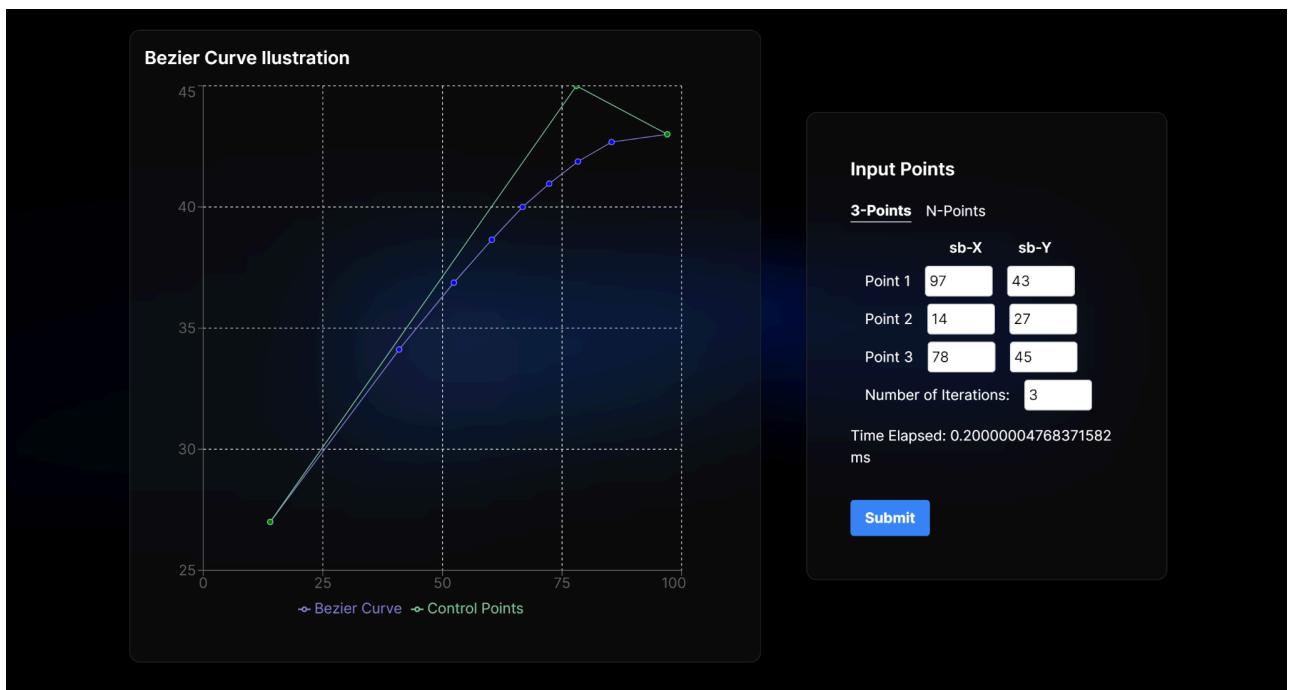


Gambar 8 Hasil kurva set percobaan pertama algoritma brute force dengan 3 iterasi
5 iterasi

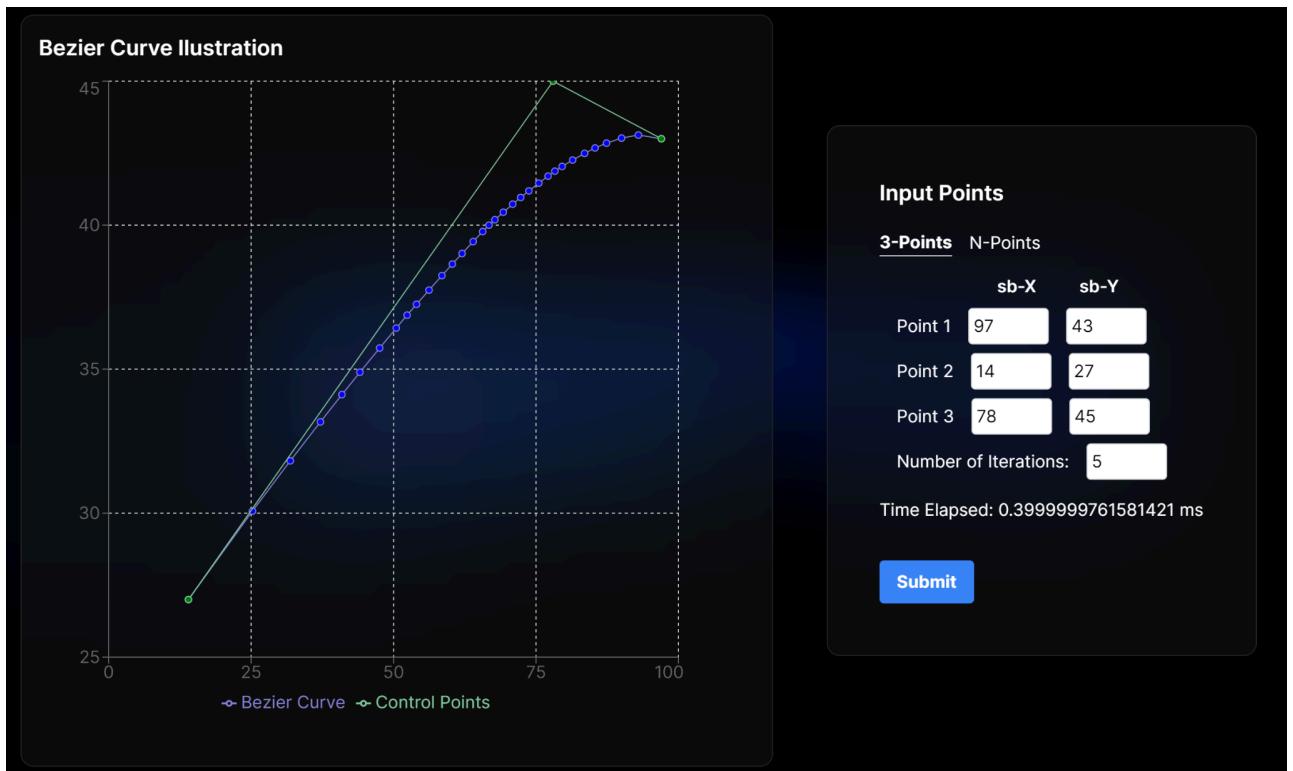


Gambar 9 Hasil kurva set percobaan pertama algoritma brute force dengan 5 iterasi

- b) Algoritma Divide and Conquer
3 iterasi



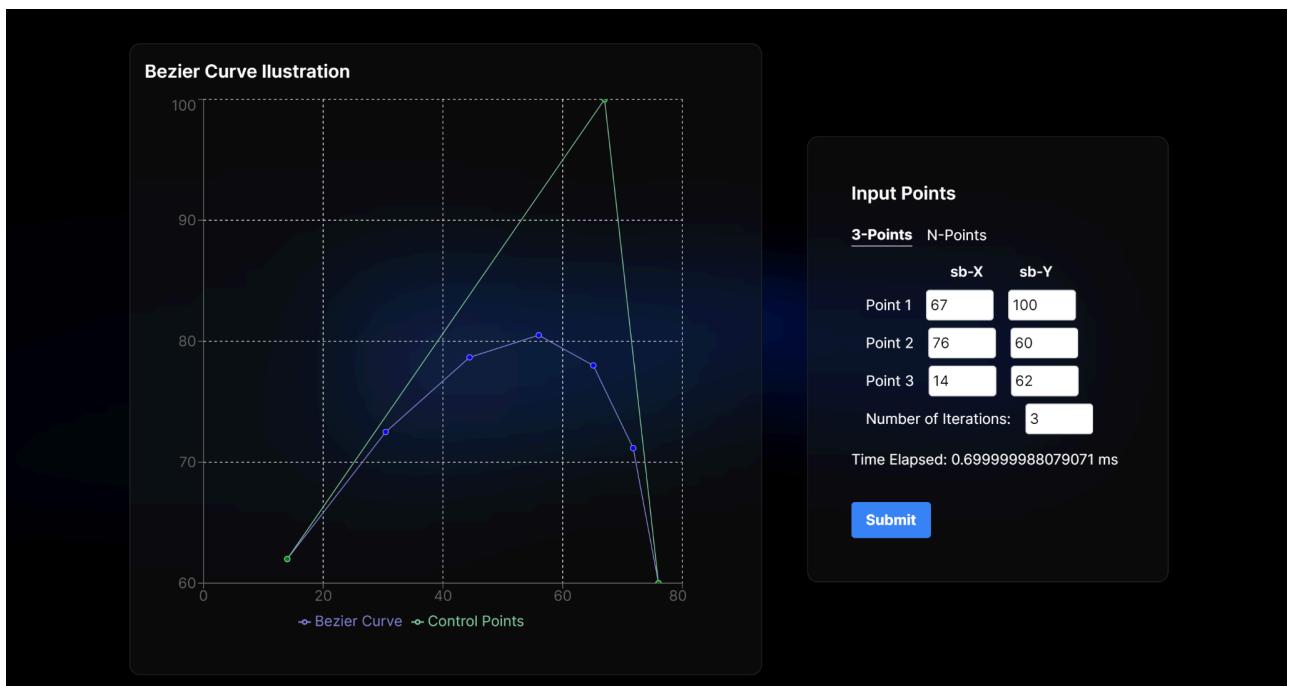
Gambar 10 Hasil kurva set percobaan pertama algoritma divide and conquer dengan 3 iterasi
5 iterasi



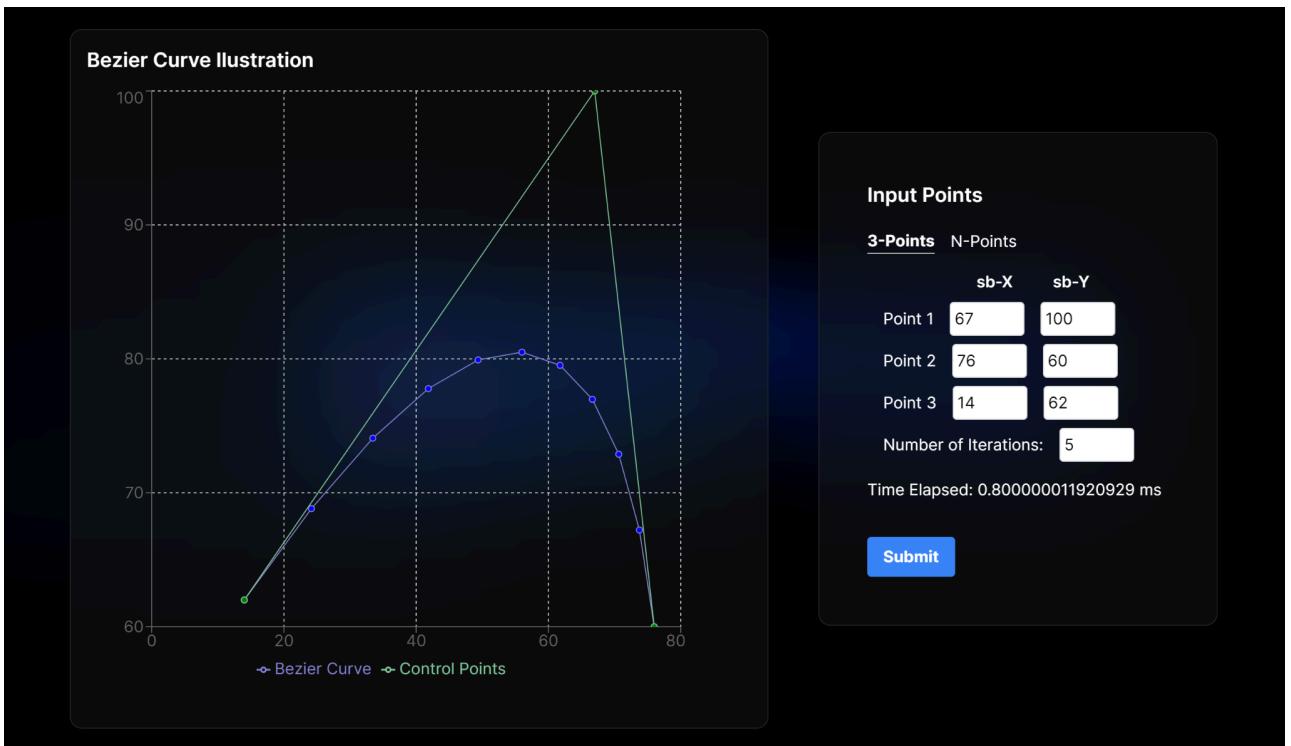
Gambar 11 Hasil kurva set percobaan pertama algoritma divide and conquer dengan 5 iterasi

4.2 Set Percobaan Kedua

- a) Algoritma Brute Force
- 3 iterasi

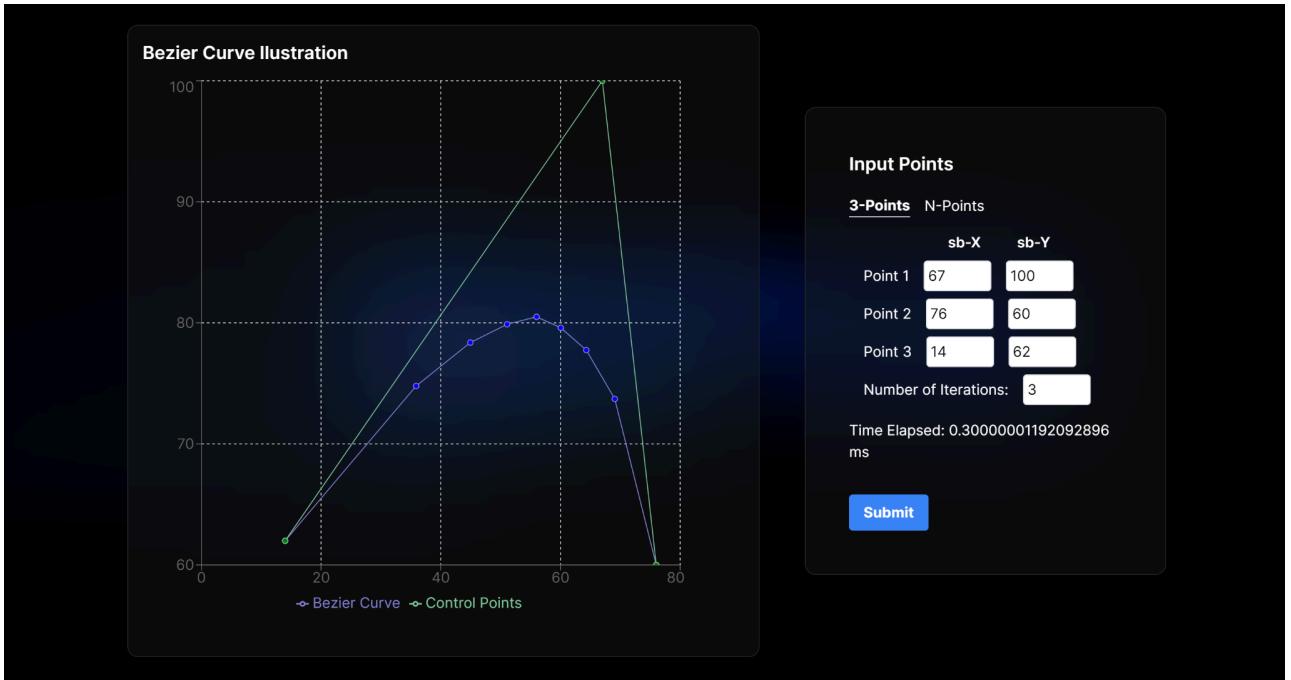


Gambar 12 Hasil kurva set percobaan kedua algoritma brute force dengan 3 iterasi
5 iterasi

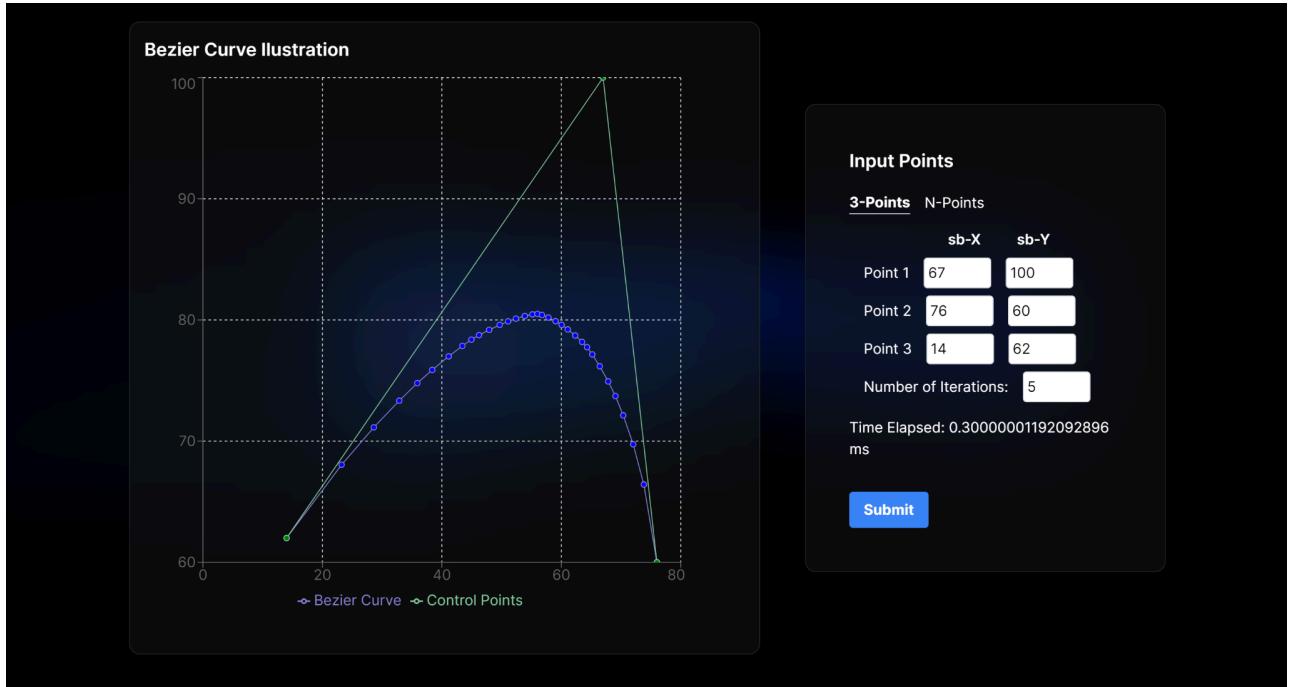


Gambar 13 Hasil kurva set percobaan kedua algoritma brute force dengan 5 iterasi

- b) Algoritma Divide and Conquer
3 iterasi



Gambar 14 Hasil kurva set percobaan kedua algoritma divide and conquer dengan 3 iterasi
5 iterasi

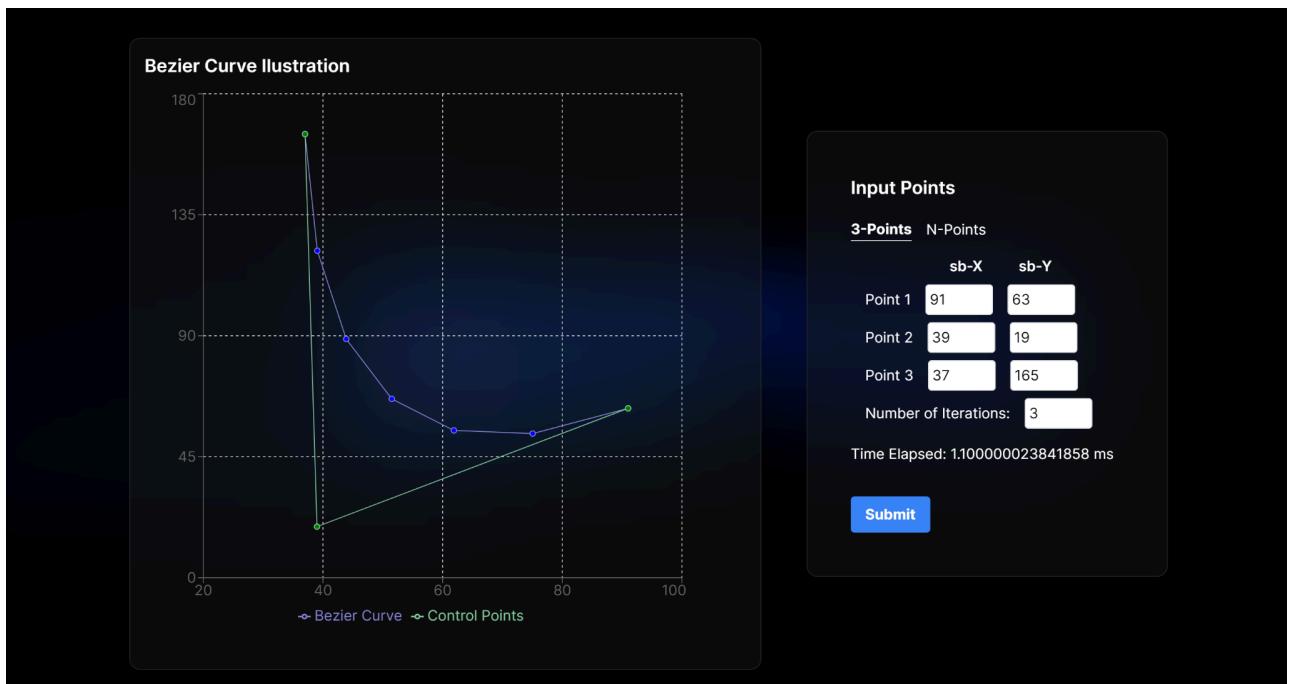


Gambar 15 Hasil kurva set percobaan kedua algoritma divide and conquer dengan 5 iterasi

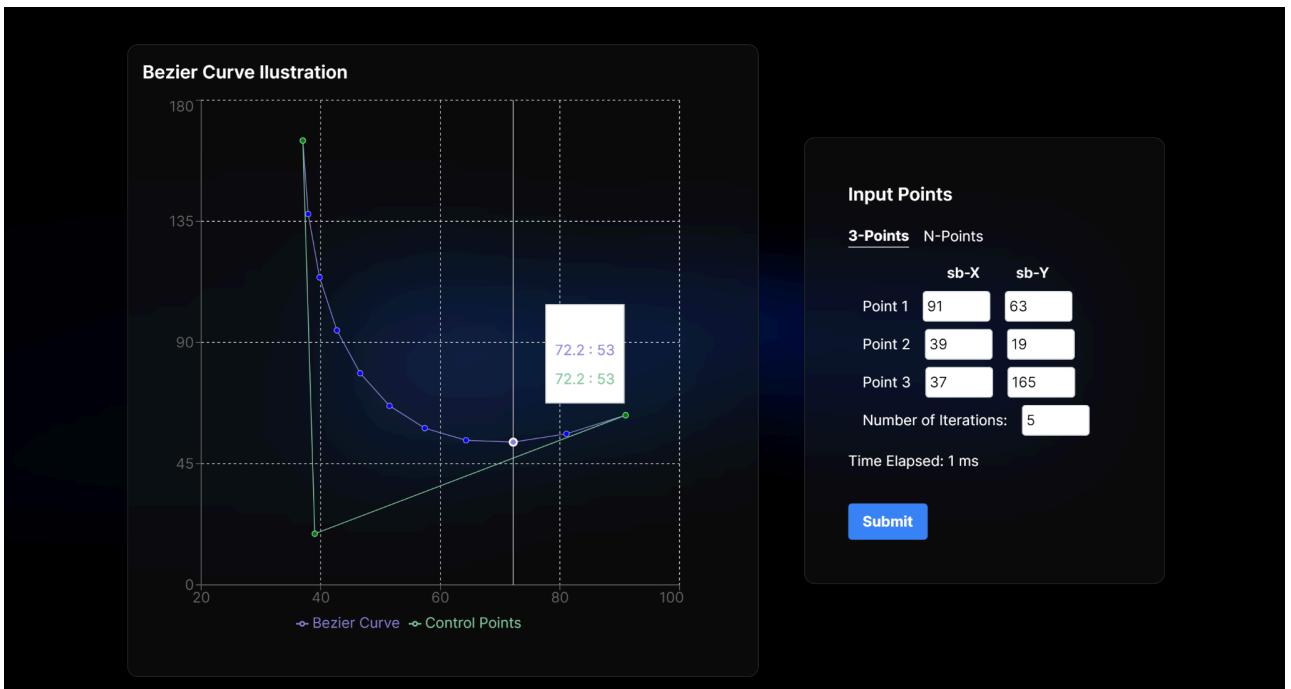
4.3 Set Percobaan Ketiga

a) Algoritma Brute Force

3 iterasi

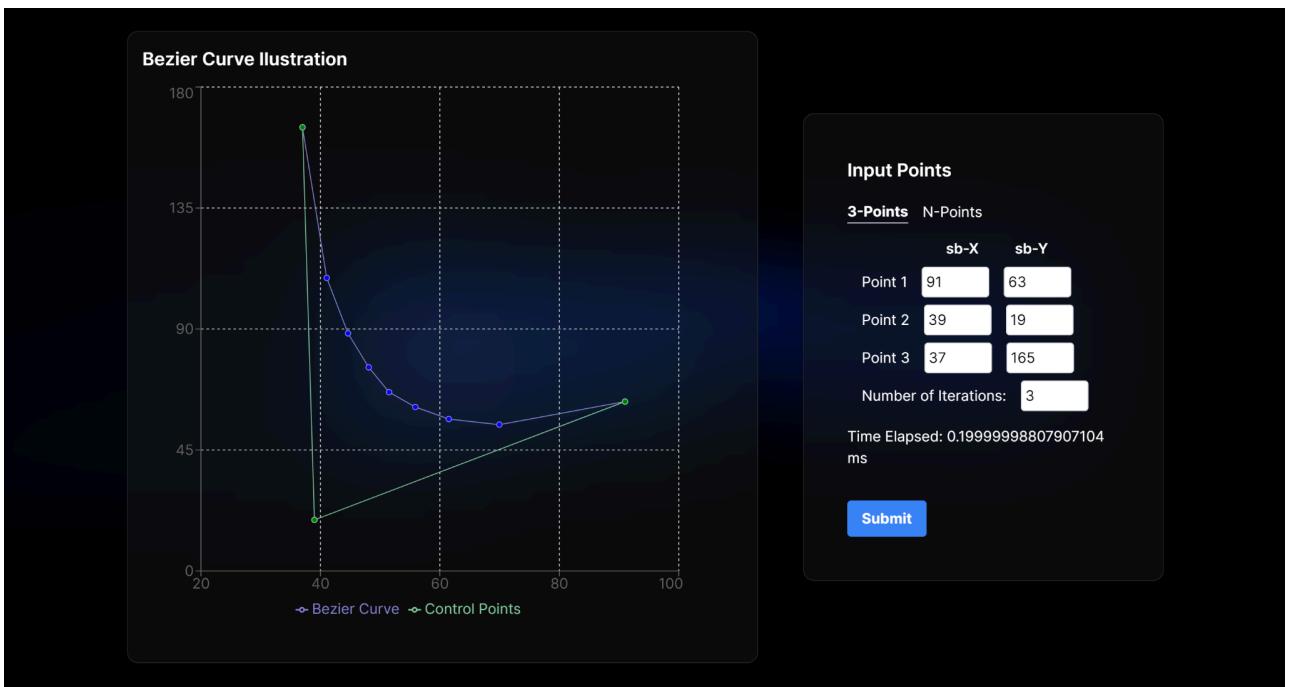


Gambar 16 Hasil kurva set percobaan ketiga algoritma brute force dengan 3 iterasi
5 iterasi

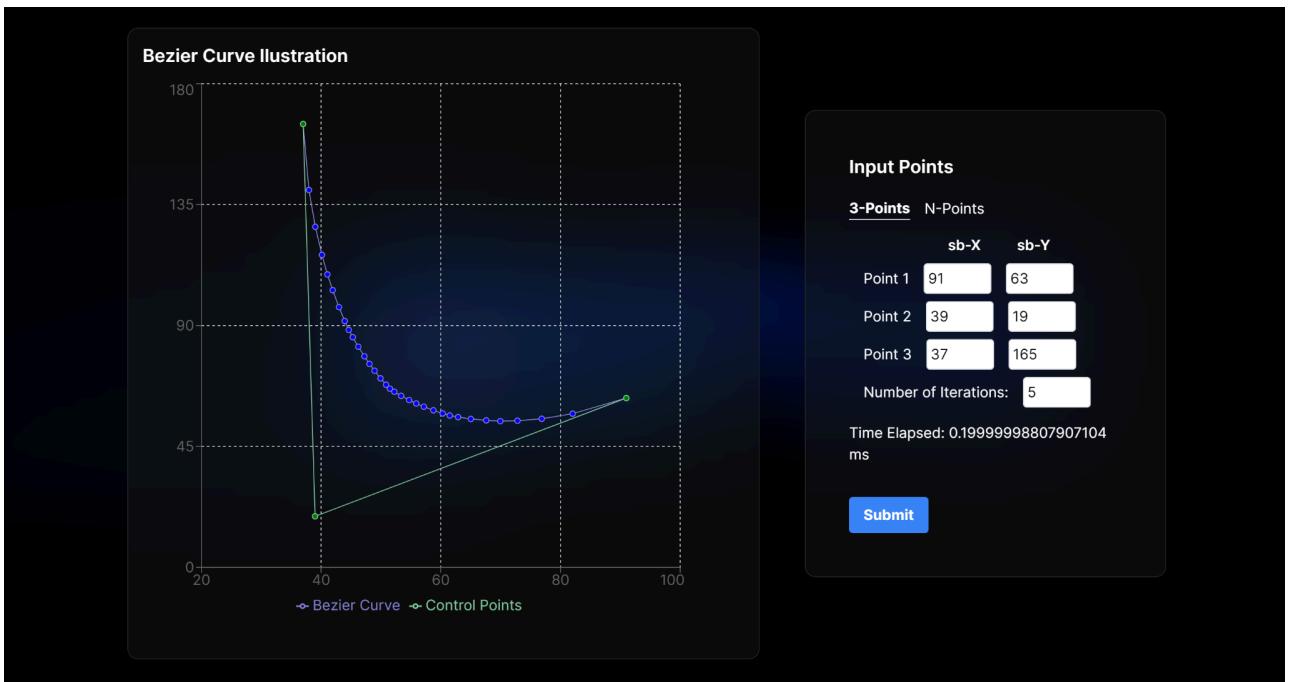


Gambar 17 Hasil kurva set percobaan ketiga algoritma brute force dengan 5 iterasi

- b) Algoritma Divide and Conquer
3 iterasi



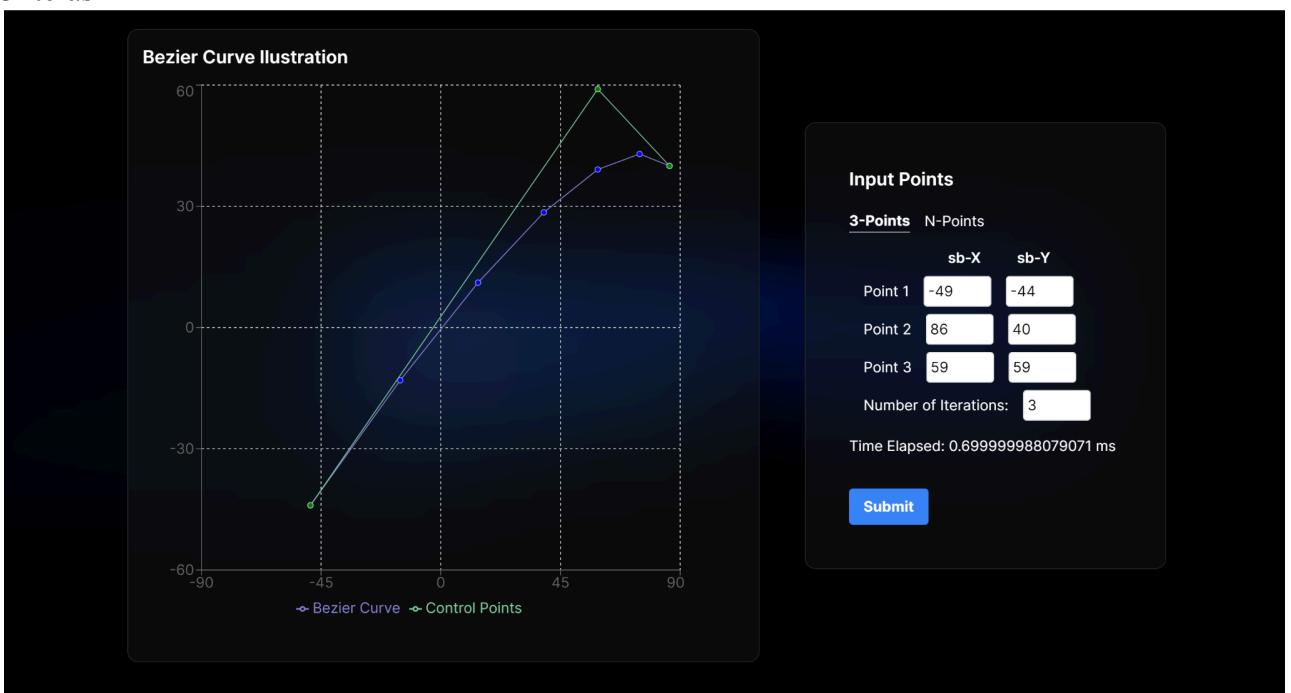
Gambar 18 Hasil kurva set percobaan ketiga algoritma divide and conquer dengan 3 iterasi
5 iterasi



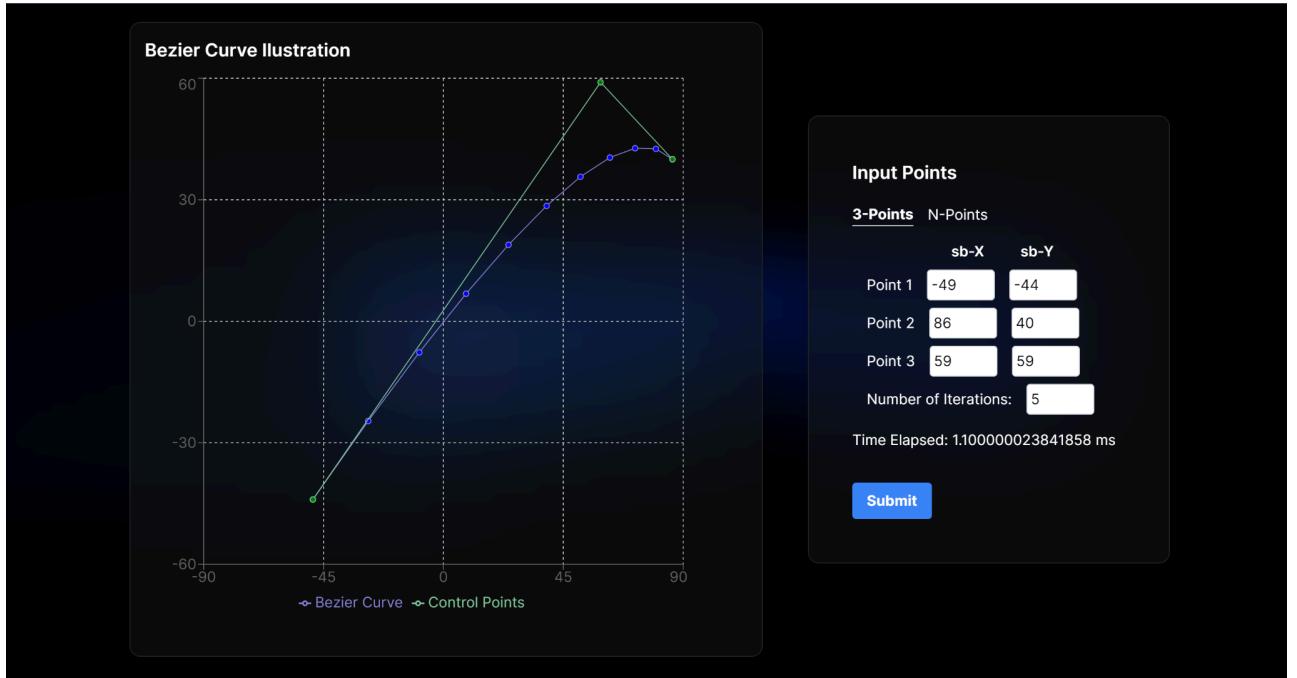
Gambar 19 Hasil kurva set percobaan ketiga algoritma divide and conquer dengan 5 iterasi

4.4 Set Percobaan Keempat

- a) Algoritma Brute Force
- 3 iterasi

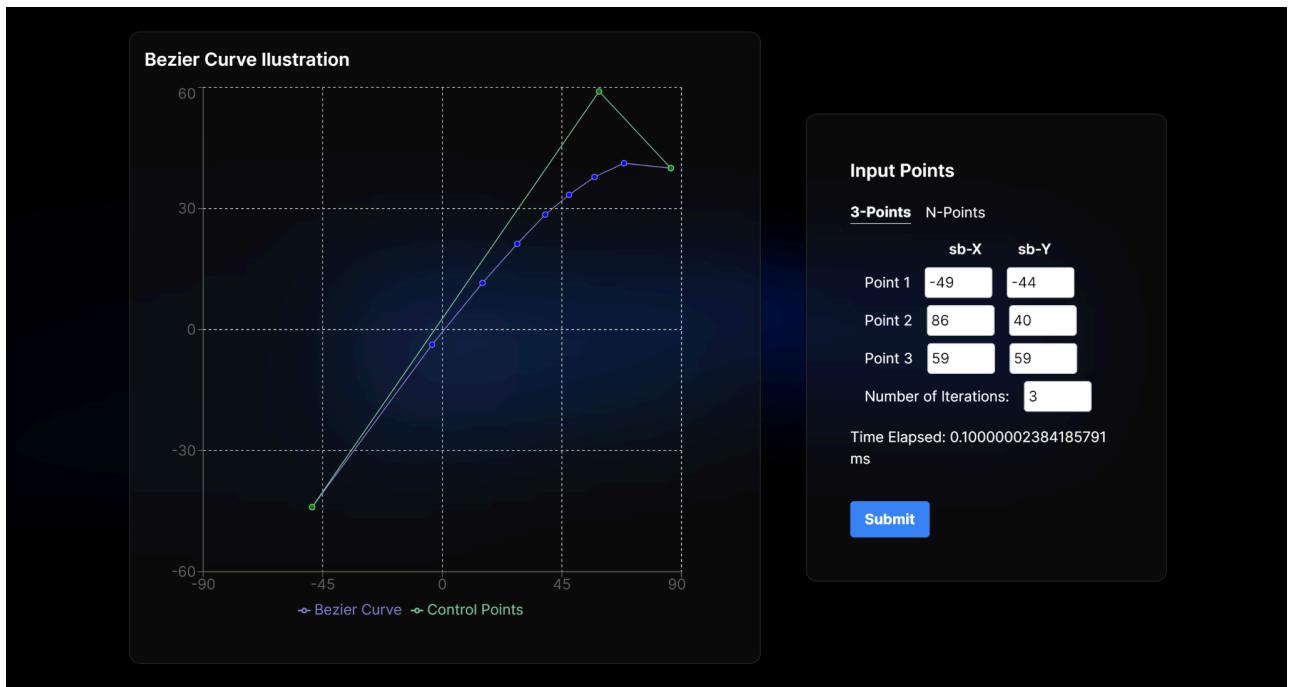


Gambar 20 Hasil kurva set percobaan keempat algoritma brute force dengan 3 iterasi
5 iterasi

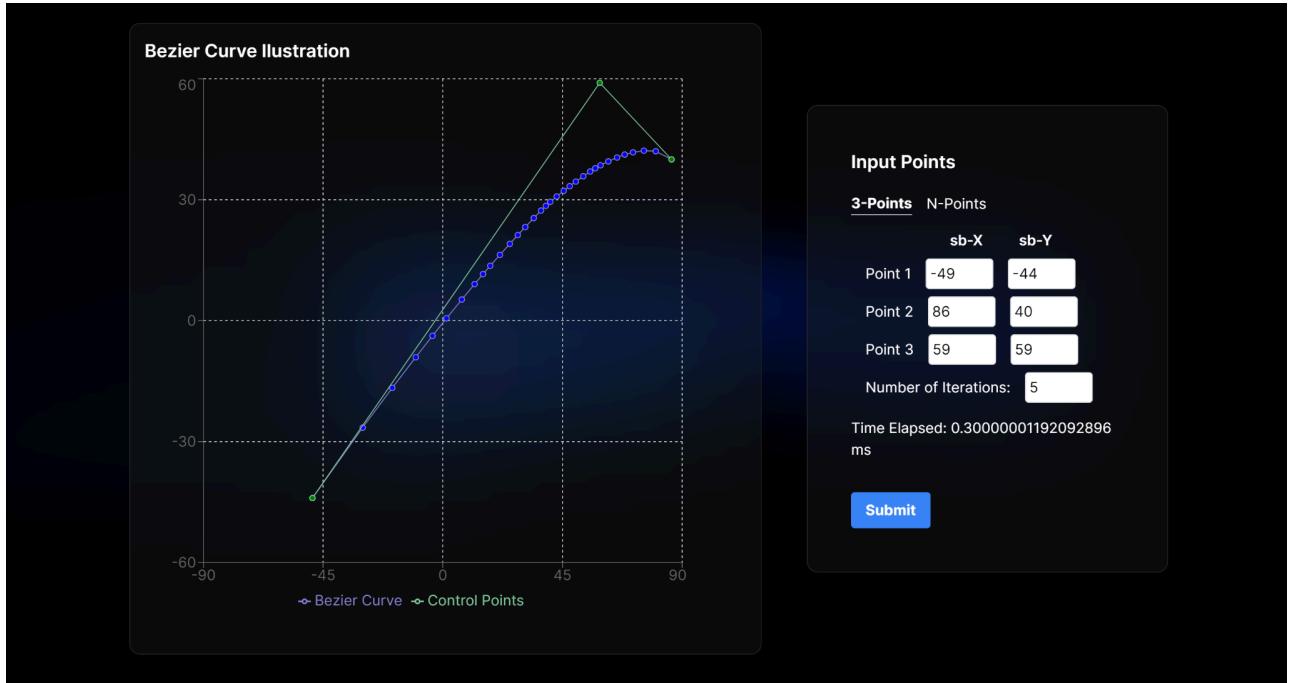


Gambar 21 Hasil kurva set percobaan keempat algoritma brute force dengan 5 iterasi

- b) Algoritma Divide and Conquer
3 iterasi



Gambar 22 Hasil kurva set percobaan keempat algoritma divide and conquer dengan 3 iterasi
5 iterasi

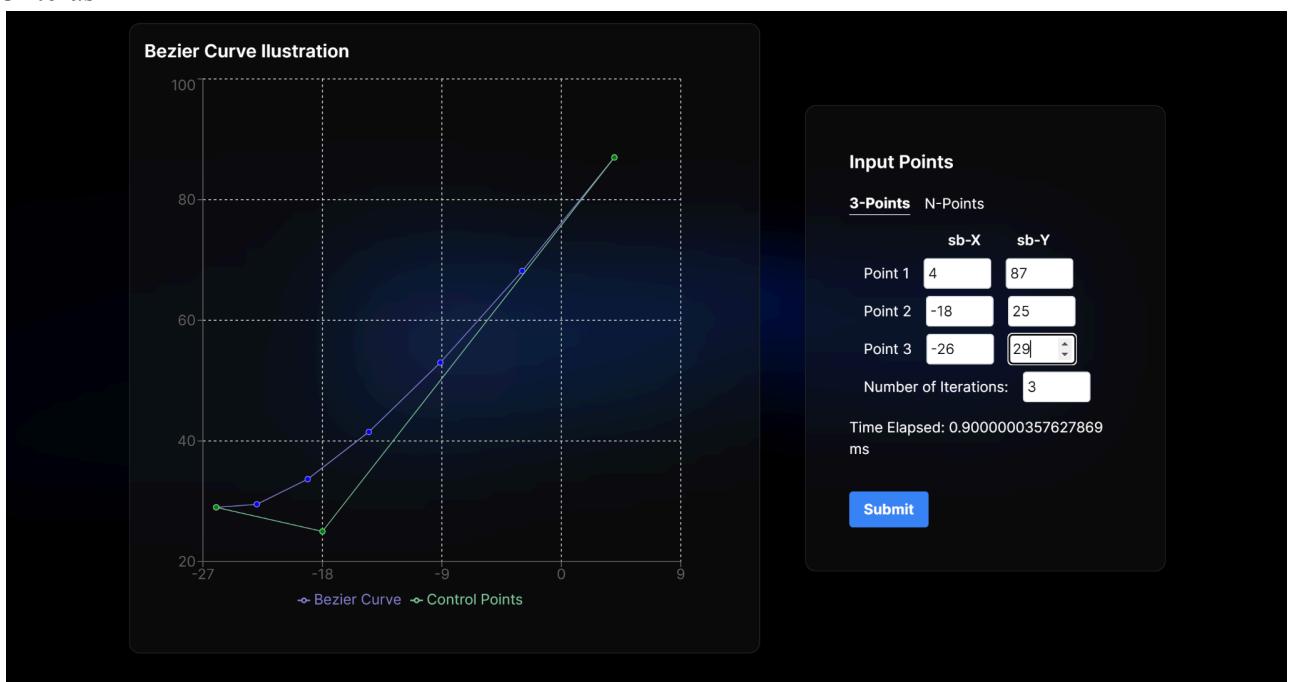


Gambar 23 Hasil kurva set percobaan keempat algoritma divide and conquer dengan 3 iterasi

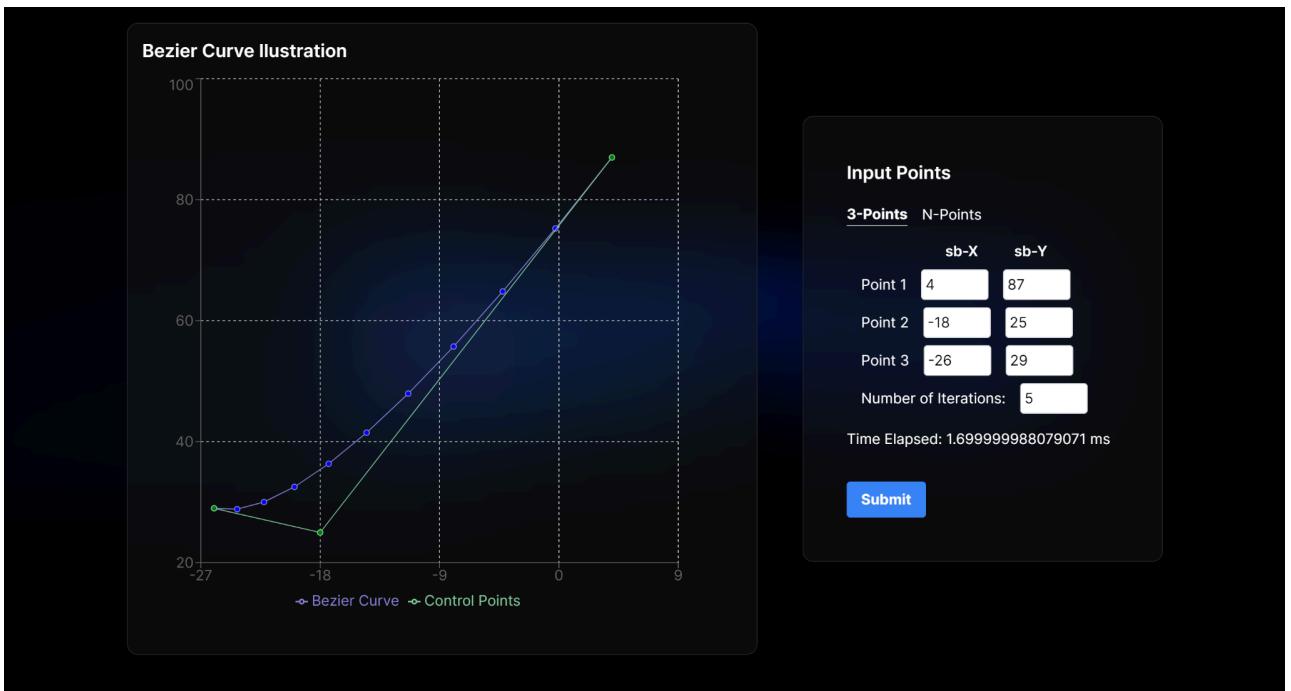
4.5 Set Percobaan Kelima

a) Algoritma Brute Force

3 iterasi

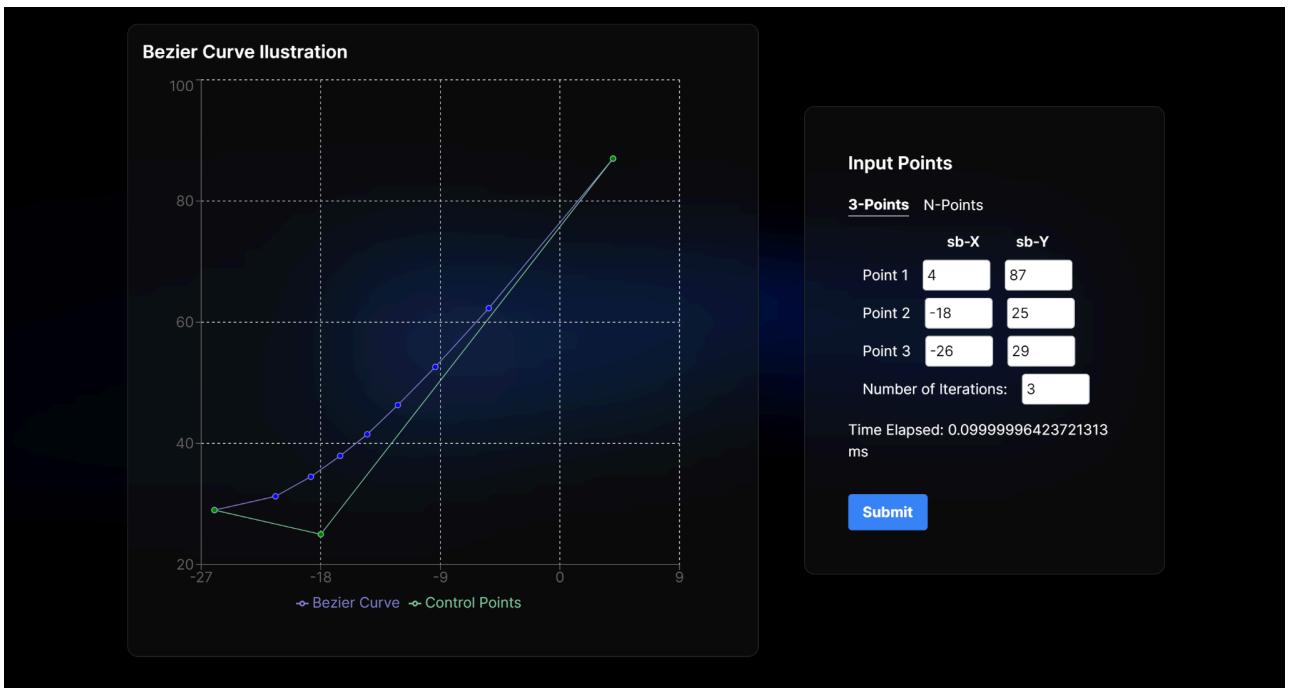


Gambar 24 Hasil kurva set percobaan kelima algoritma brute force dengan 3 iterasi
5 iterasi

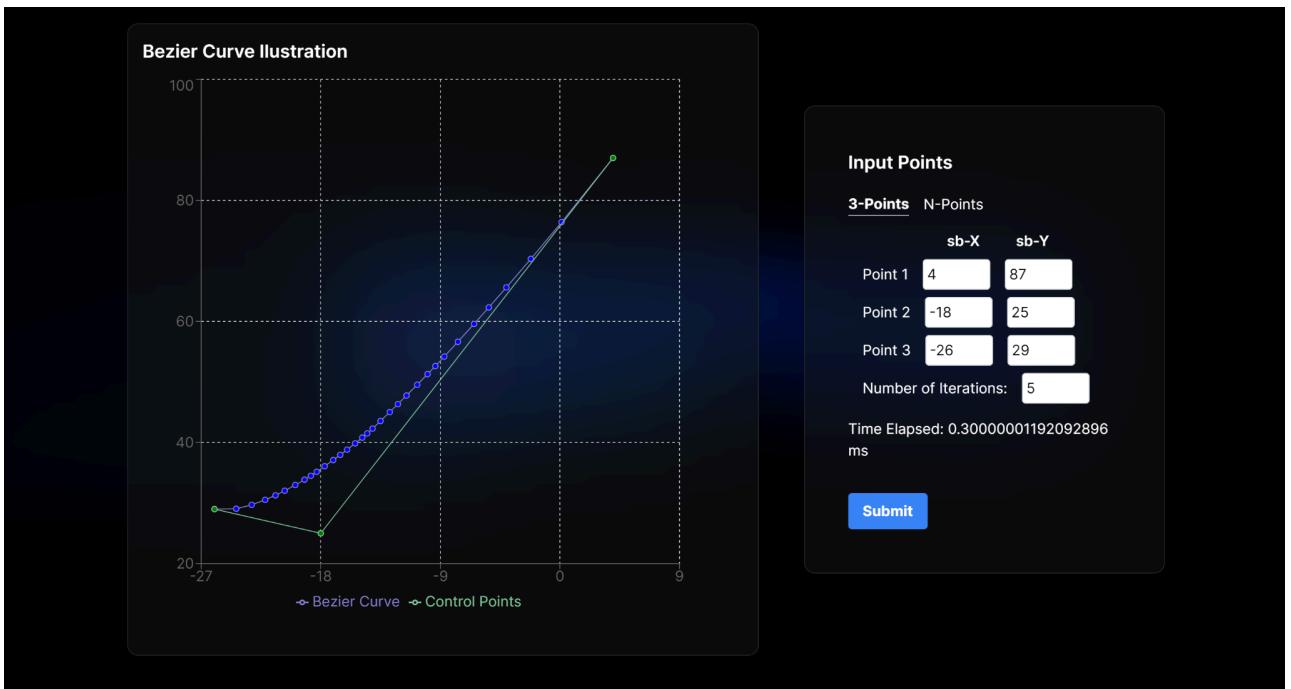


Gambar 25 Hasil kurva set percobaan kelima algoritma brute force dengan 5 iterasi

- b) Algoritma Divide and Conquer
3 iterasi



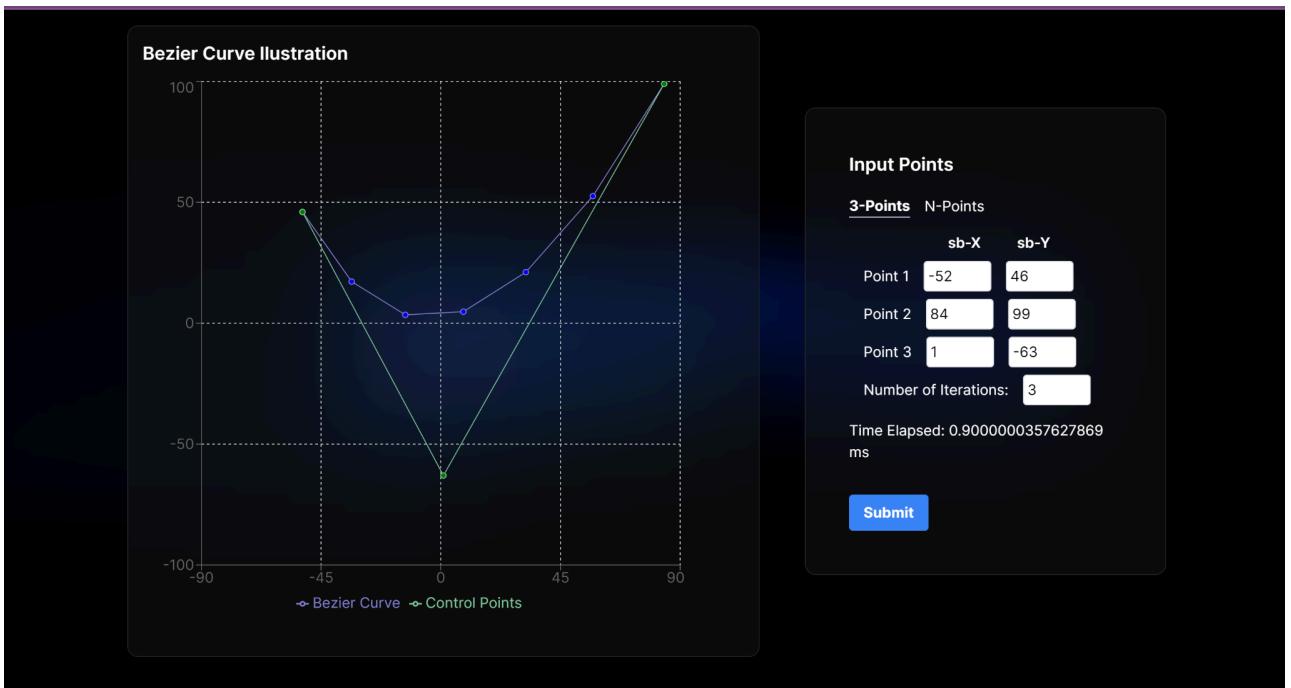
Gambar 26 Hasil kurva set percobaan kelima algoritma divide and conquer dengan 3 iterasi
5 iterasi



Gambar 27 Hasil kurva set percobaan kelima algoritma divide and conquer dengan 5 iterasi

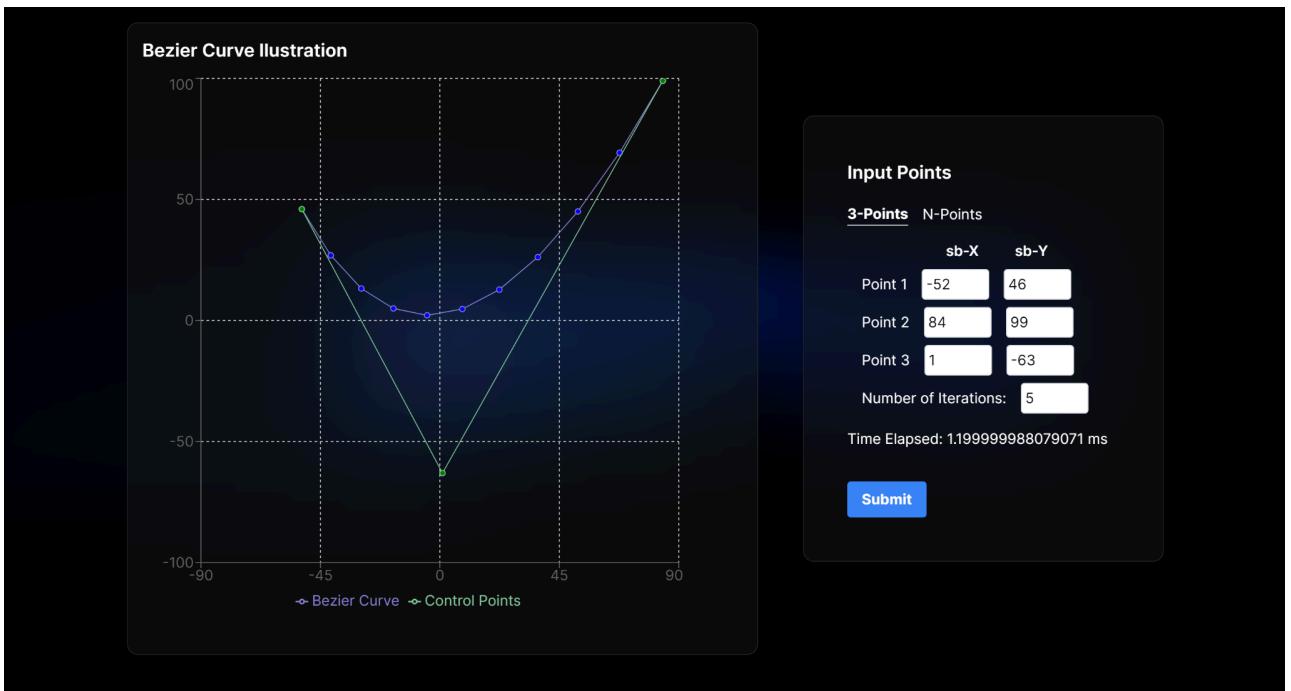
4.6 Set Percobaan Keenam

- a) Algoritma Brute Force
- 3 iterasi



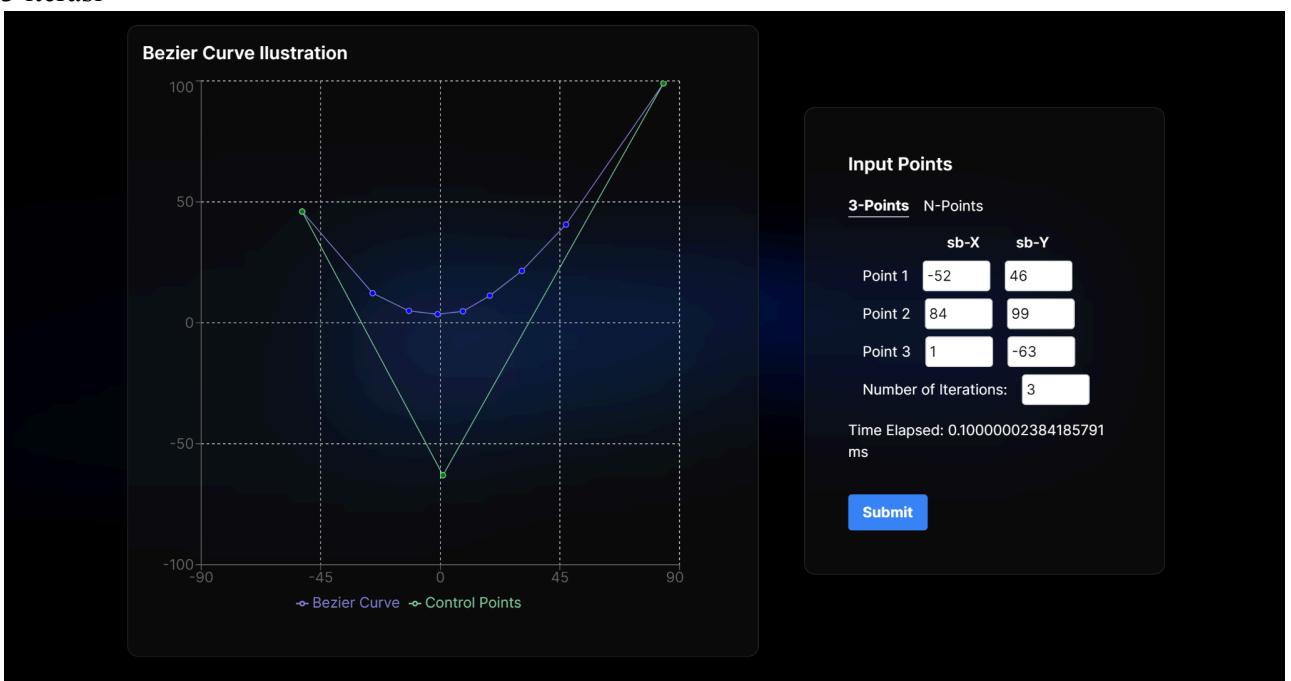
Gambar 28 Hasil kurva set percobaan keenam algoritma brute force dengan 3 iterasi

5 iterasi



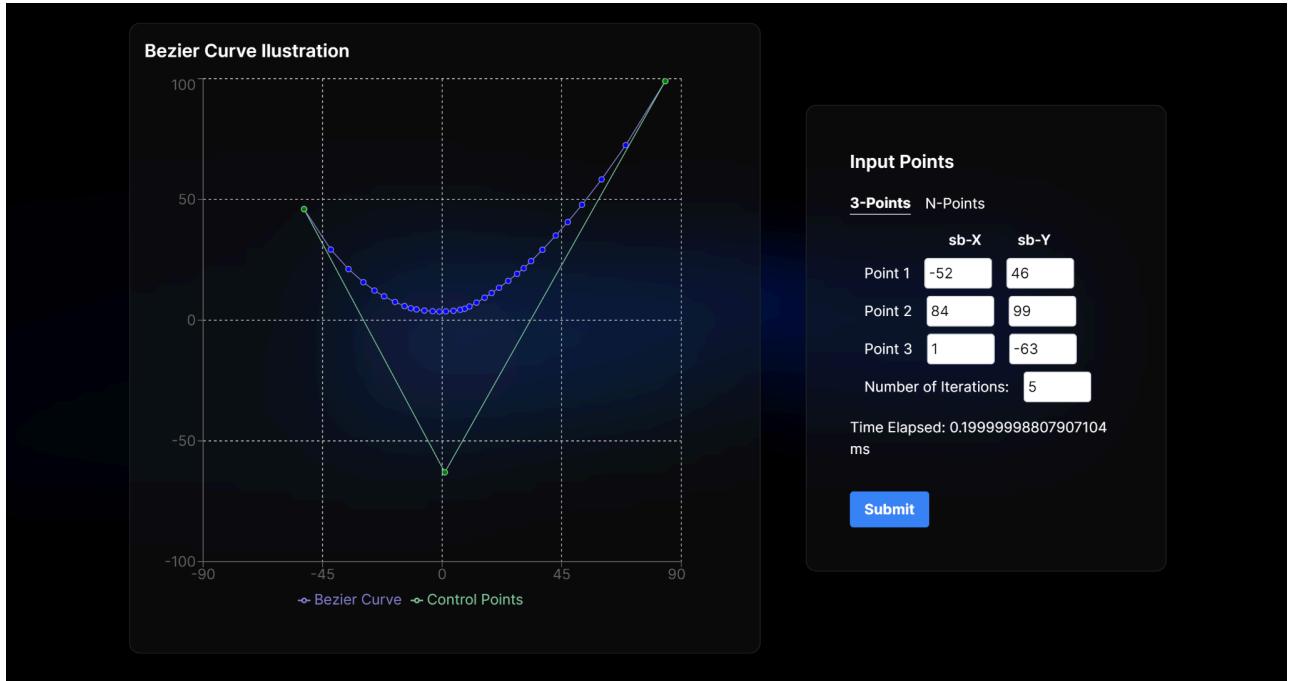
Gambar 29 Hasil kurva set percobaan keenam algoritma brute force dengan 5 iterasi

- b) Algoritma Divide and Conquer
3 iterasi



Gambar 30 Hasil kurva set percobaan keenam algoritma divide and conquer dengan 3 iterasi

5 iterasi



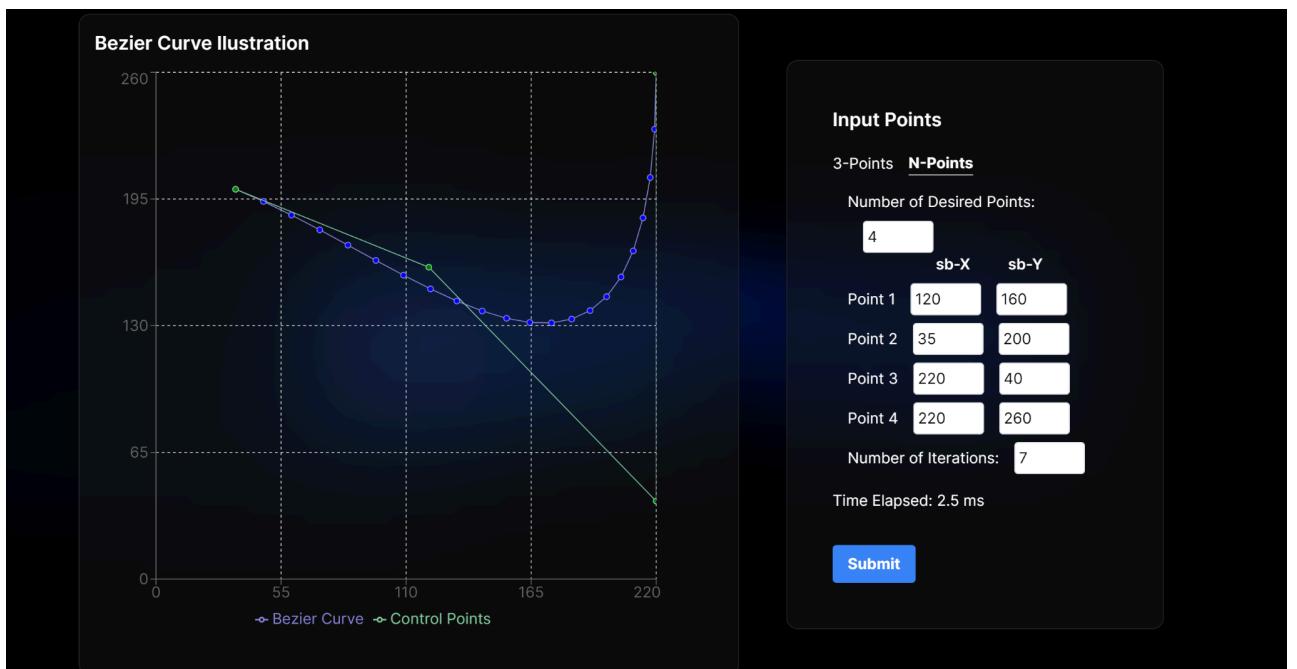
Gambar 31 Hasil kurva set percobaan keenam algoritma divide and conquer dengan 5 iterasi

4.7 Percobaan Kurva Bezier Kubik

Pada percobaan kurva bezier kubik, kita akan menerapkan percobaan sesuai dengan gambar 1 yang berada pada bab 1. Pada kurva bezier kubik tersebut digunakan empat buah titik kontrol yaitu (120,160), (35,200), (220,260), dan juga (220, 40). Pada percobaan kali ini, kita akan menggunakan iterasi yang lebih ekstrim dibanding percobaan sebelumnya. Iterasi yang akan kami gunakan pada percobaan kali ini adalah 7 dan 10 iterasi.

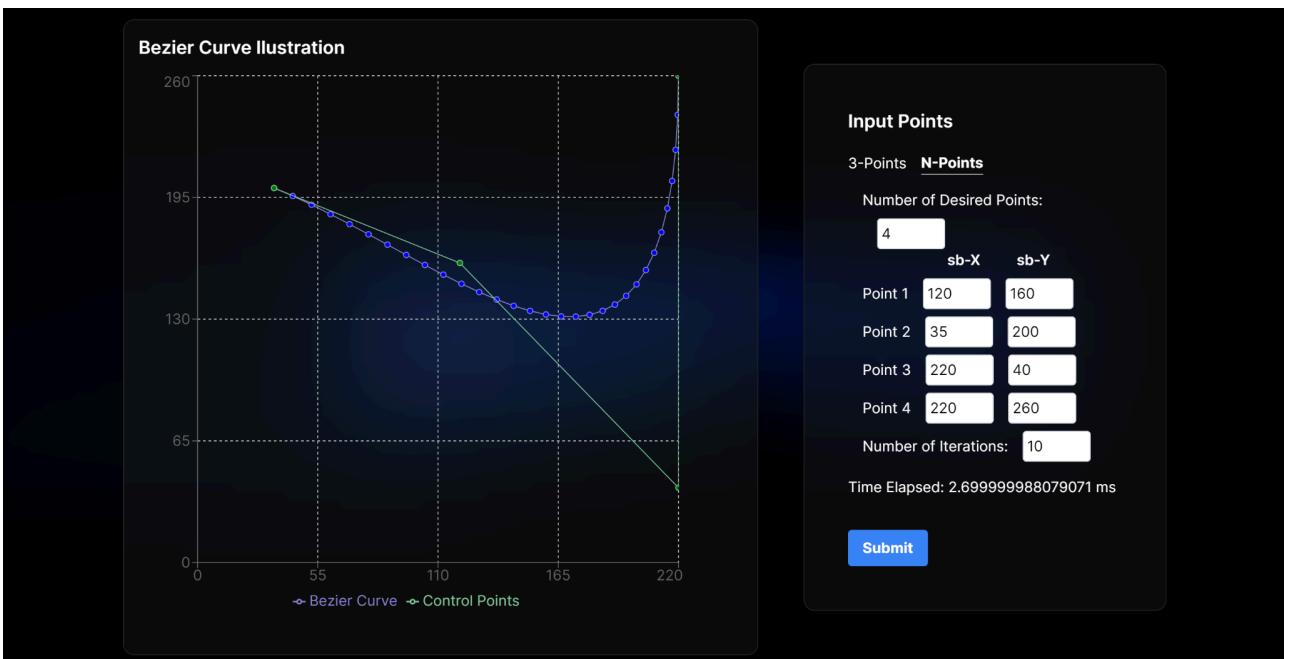
a) Algoritma Brute Force

Iterasi 7



Gambar 32 Hasil kurva percobaan bezier kubik algoritma brute force dengan 7 iterasi

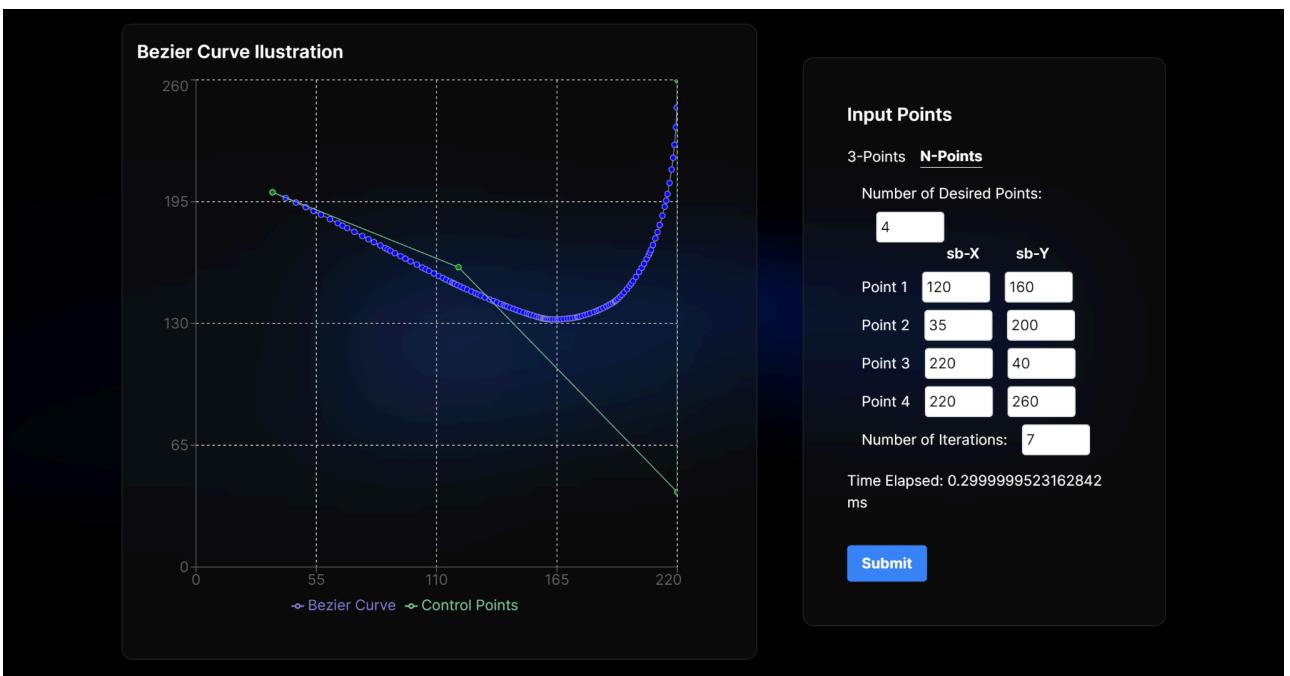
Iterasi 10



Gambar 33 Hasil kurva percobaan bezier kubik algoritma brute force dengan 10 iterasi

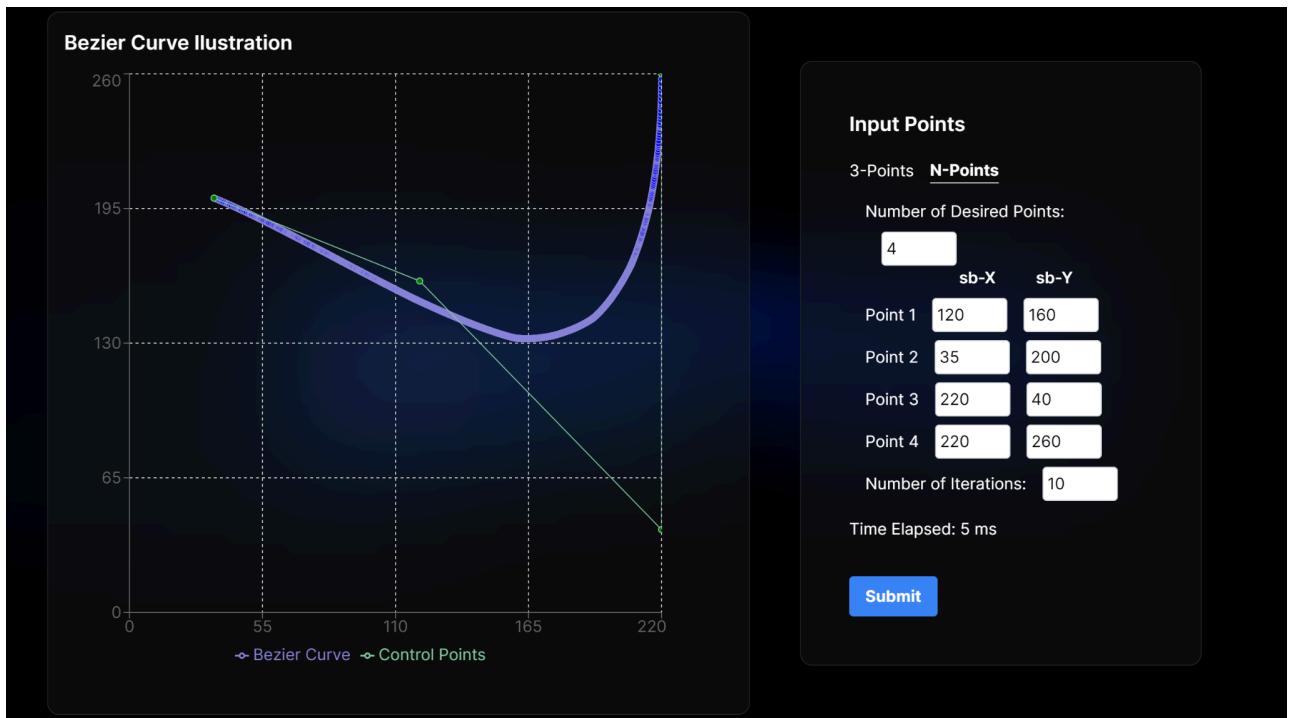
b) Algoritma Divide and Conquer

Iterasi 7



Gambar 34 Hasil kurva percobaan bezier kubik algoritma divide and conquer dengan 7 iterasi

Iterasi 10

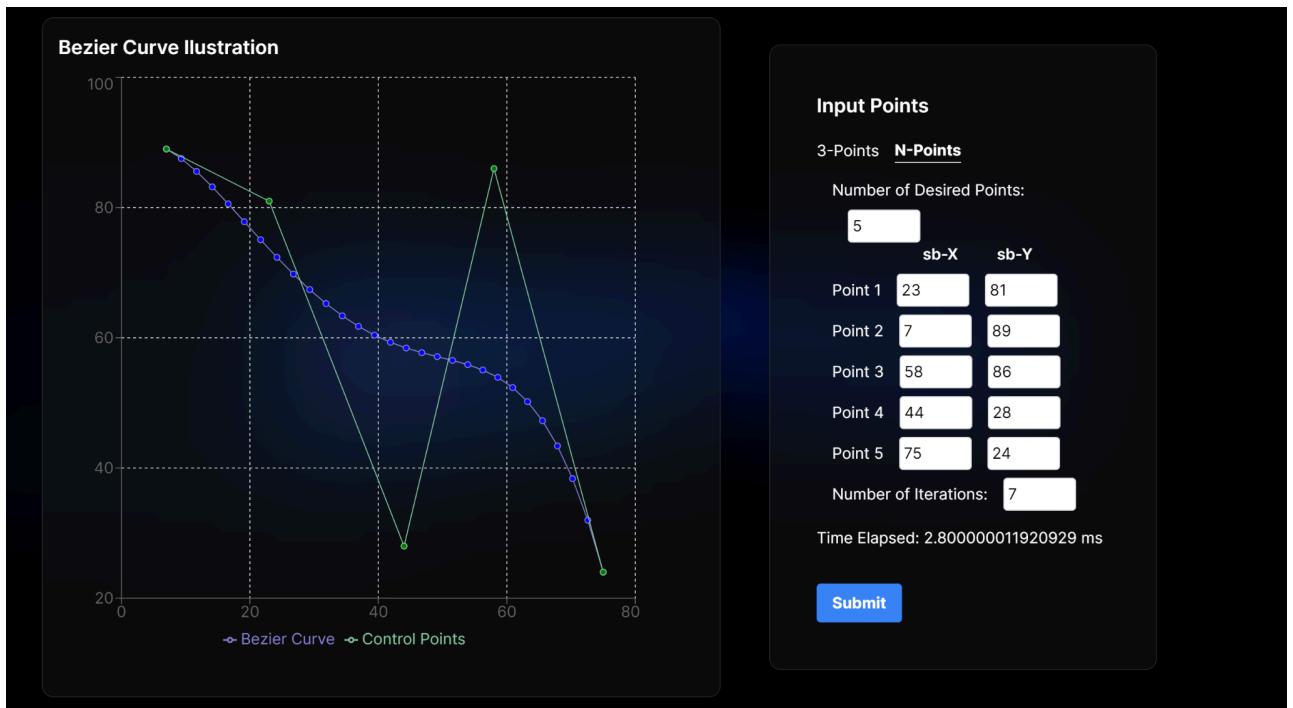


Gambar 35 Hasil kurva percobaan bezier kubik algoritma divide and conquer dengan 10 iterasi

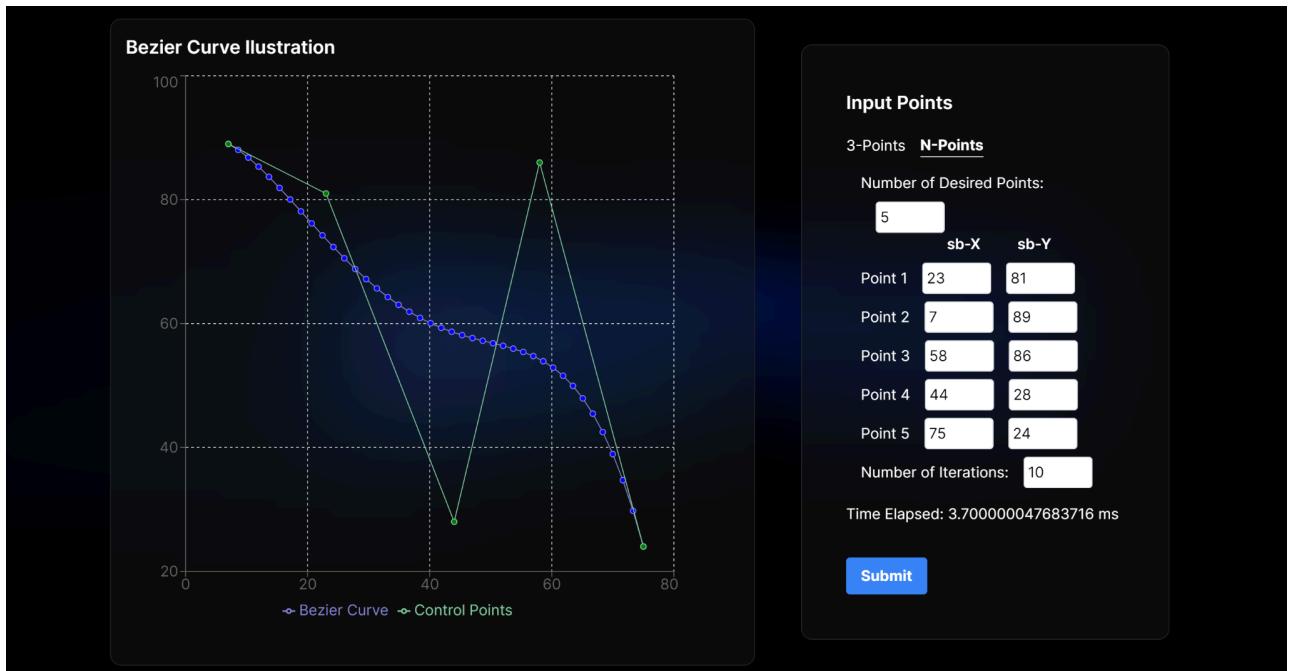
4.8 Percobaan Kurva Bezier Kuartik

Pada percobaan kurva bezier kuartik, sesuai dengan definisi kurva bezier kuartik tersebut kami akan menggunakan empat buah titik kontrol yaitu (23,81), (7,89), (58,86), (44,28) dan juga (75, 24). Pada percobaan kali ini, kita akan menggunakan iterasi yang sama dengan percobaan kurva bezier kubik. Iterasi yang akan kami gunakan pada percobaan kurva bezier kuartik ini adalah 7 dan 10 iterasi.

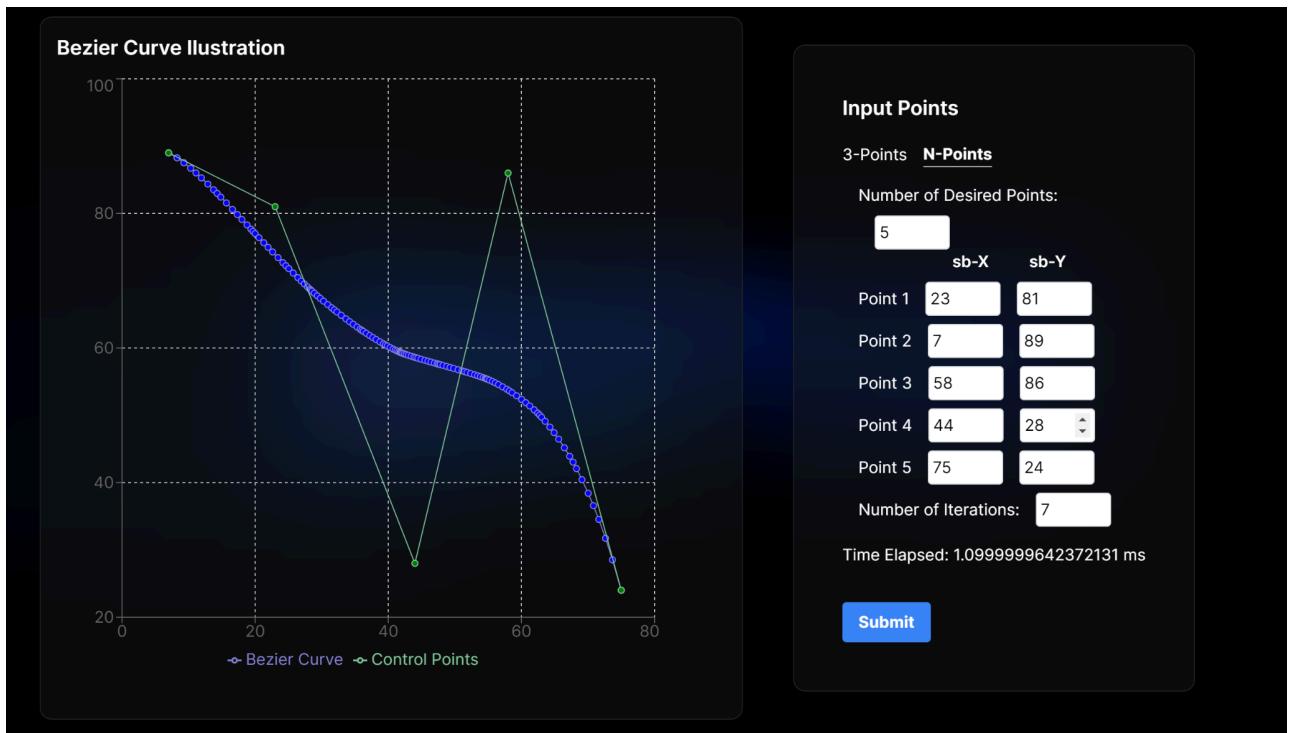
- a) Algoritma Brute Force
7 iterasi



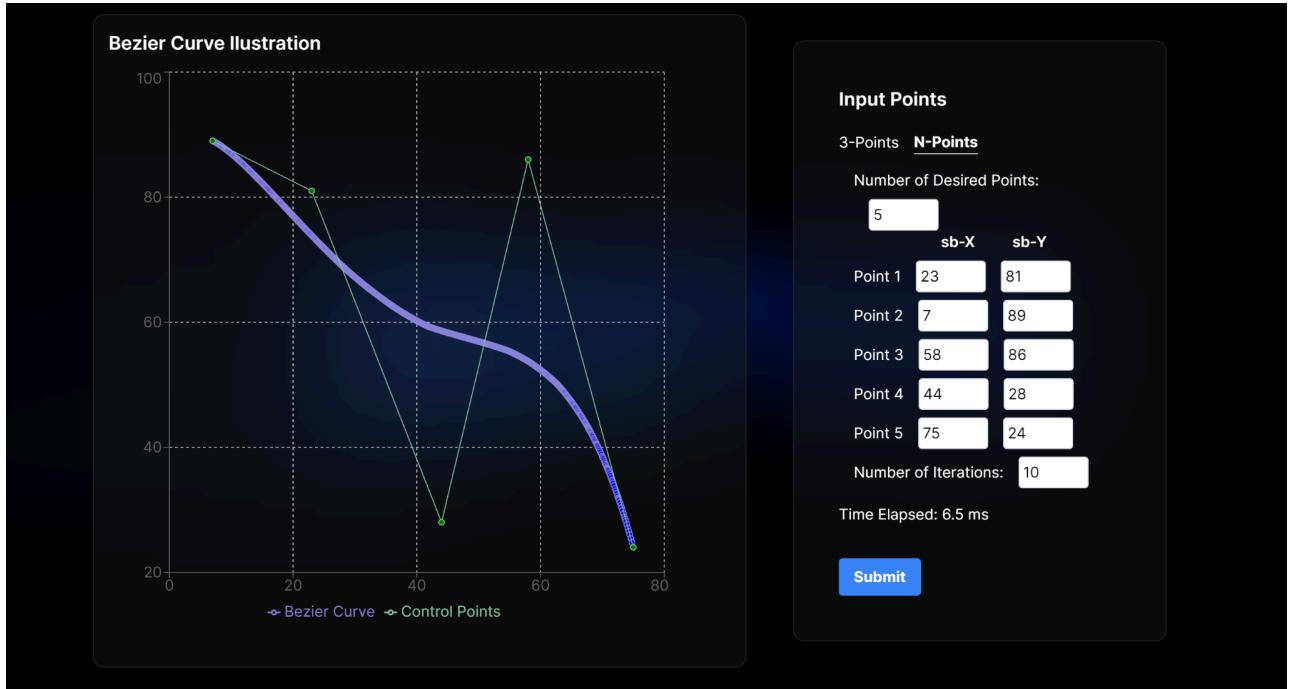
Gambar 36 Hasil kurva percobaan bezier kuartik algoritma brute force dengan 7 iterasi
10 iterasi



Gambar 37 Hasil kurva percobaan bezier kuartik algoritma brute force dengan 10 iterasi
b) Algoritma Divide and Conquer
7 iterasi



Gambar 38 Hasil kurva percobaan bezier kuartik algoritma divide and conquer dengan 7 iterasi
10 iterasi



Gambar 39 Hasil kurva percobaan bezier kuartik algoritma divide and conquer dengan 7 iterasi

BAB V

ANALISIS, SARAN, DAN KESIMPULAN

5.1 Analisis Efektivitas Algoritma

1. Algoritma Brute Force

Algoritma ini menunjukkan keefektifan dalam kasus dengan jumlah titik kontrol yang lebih sedikit. Namun, algoritma brute force memiliki keterbatasan dalam skala menjawab persoalan yang berskala besar dan waktu komputasi menjadi faktor pembatas utamanya. Penggunaan koefisien binomial dalam dalam algoritma brute force membantu mengurangi perhitungan berulang. Berdasarkan program kurva bezier algoritma brute force yang telah kami rancang didapatkan kompleksitas algoritma sebagai berikut:

Fungsi koefisien binomial:

$$T(n) = O(n)$$

n = indeks koefisien

Fungsi utama:

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

n = jumlah titik kontrol

Implementasi pada GUI:

$$T(n) = O(n)$$

n = jumlah iterasi dalam pembentukan kurva bezier

2. Algoritma Divide and Conquer

Signifikan lebih efisien daripada Brute Force, terutama ketika dihadapkan pada iterasi dan jumlah titik kontrol yang lebih banyak. Implementasinya menunjukkan efektivitas dalam menghasilkan kurva yang lebih halus dan kontinu, mendemonstrasikan superioritas dalam pembagian masalah menjadi sub-masalah yang lebih mudah dikelola. Berdasarkan program kurva bezier algoritma divide and conquer yang telah kami rancang didapatkan kompleksitas algoritma sebagai berikut:

Fungsi midpoint:

$$T(n) = O(1)$$

Fungsi untuk melakukan perhitungan matematis

Fungsi utama:

$$T(n) = O(n(\log(n)))$$

n = jumlah titik kontrol

Implementasi pada GUI:

$$T(n) = O(1)$$

Hanya melakukan pemanggilan fungsi

5.2 Saran

1. Optimasi Algoritma:

Mencari dan memaksimalkan metode optimasi lanjutan untuk algoritma Divide and Conquer, termasuk teknik pemrograman dinamis dan pengurangan pemanggilan rekursif.

2. Interaktivitas Pengguna:

Mengembangkan visualisasi pada website serta UI untuk memudahkan pengguna memodifikasi parameter secara dinamis, memberikan visualisasi langsung terhadap perubahan tersebut.

3. Analisis pengujian :

Melakukan analisis komparatif lebih mendalam terhadap kedua algoritma diberbagai skenario pengujian untuk menilai keefektifan mereka dalam berbagai kondisi dan mengevaluasi performa aplikasi di berbagai platform untuk memastikan kestabilan dan efisiensi.

5.3 Kesimpulan

Secara garis besar, tugas kecil Strategi Algoritma yang kedua ini merupakan tugas yang diberikan untuk meningkatkan kemampuan mahasiswa dalam mata kuliah strategi algoritma terutama dalam materi brute force dan divide and conquer. Secara garis besar, algoritma bruteforce merupakan algoritma yang mencoba segala kemungkinan yang ada, sementara untuk devide and conquer merupakan algoritma yang memiliki strategi memecahkan masalah menjadi lebih kecil atau menjadi sub masalah sehingga masalah lebih mudah diselesaikan.

Dalam tugas ini, didapatkan kesimpulan bahwa program dapat diselesaikan dan berjalan dengan baik. Program dengan algoritma brute force dapat menghasilkan kurva yang cukup akurat dengan jumlah titik iterasi yang beragam. Algoritma ini memiliki jumlah titik pada kurva yang lebih kecil bila dibandingkan dengan algoritma divide and conquer. Algoritma divide and conquer memiliki titik pada kurva yang lebih banyak karena algoritmanya yang memiliki rumus eksponensial.

Dengan jumlah titik yang lebih banyak pada kurva, serta kecepatan proses yang lebih cepat algoritma Divide and Conquer merupakan algoritma yang lebih optimal jika dibandingkan dengan algoritma brute force.

5.4 Refleksi

Tugas ini mengajarkan pentingnya pemilihan dan penerapan algoritma yang tepat dalam membuat suatu program. Tugas ini berhasil menunjukkan bagaimana teori algoritma dapat diterapkan dengan sukses dalam praktik, khususnya dalam materi divide and conquer dalam membuat kurva bezier. Pengalaman ini juga menyoroti pentingnya optimasi dan efisiensi dalam membuat suatu program, dan pentingnya memiliki web yang interaktif dan nyaman untuk digunakan.

Secara umum, tugas ini adalah sebuah ilustrasi yang sangat baik tentang bagaimana konsep teori dapat digabungkan dengan penerapannya dalam praktik membuat suatu program, menunjukkan efektivitas pengetahuan algoritma dalam memecahkan masalah praktis secara efisien. Tugas ini juga menekankan pentingnya rekayasa perangkat lunak yang berkualitas, dimana perencanaan yang teliti, desain yang matang, dan eksekusi yang hati-hati menjadi faktor utama dalam menciptakan solusi teknologi yang inovatif dan berpengaruh.

Kami juga memiliki keinginan untuk mengerjakan bonus untuk program dalam melakukan visualisasi proses pembuatan kurva. Terdapat kesalahpahaman dalam mengartikan spesifikasi yang telah diperjelas dalam kolom tanya jawab Tugas Kecil 2. Dalam proses pengerjaan tugas berikutnya, kami akan mengecek kolom tanya jawab dengan rutin agar dapat mengetahui keinginan dari asisten melalui program yang telah dibuat.

LAMPIRAN

Spesifikasi Tugas Kecil 2 IF2211 Strategi Algoritma:

<https://drive.google.com/file/d/1y16HU7ZPqmq7YwP5Y9HJVof8WGv0cNO2/view?usp=sharing>

Checklist Tugas Kecil 2 IF2211 Strategi Algoritma:

Poin	Ya	Tidak
1. Program berhasil dijalankan.	✓	
2. Program dapat melakukan visualisasi kurva Bezier.	✓	
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.		✓