LAPORAN TUGAS KECIL 2

IF2211 STRATEGI ALGORITMA

Disusun oleh:

Tazkia Nizami (13522032)

Daniel Mulia Putra Manurung (13522043)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2024

DAFTAR ISI

BAB I

DESKRIPSI MASALAH

1.1 Membangun Kurva Bézier

Kurva Bézier adalah kurva halus yang banyak digunakan dalam berbagai bidang seperti desain grafis, animasi, dan manufaktur. Kurva ini dihasilkan dengan menghubungkan titik-titik kontrol yang menentukan bentuk dan arahnya.

Misalkan diberikan dua buah titik P0 dan P1 yang menjadi titik kontrol, maka kurva Bézier yang terbentuk adalah sebuah garis lurus antara dua titik. Kurva ini disebut dengan kurva Bézier linier. Misalkan terdapat sebuah titik Q0 yang berada pada garis yang dibentuk oleh P0 dan P1, maka posisinya dapat dinyatakan dengan persamaan parametrik berikut.

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

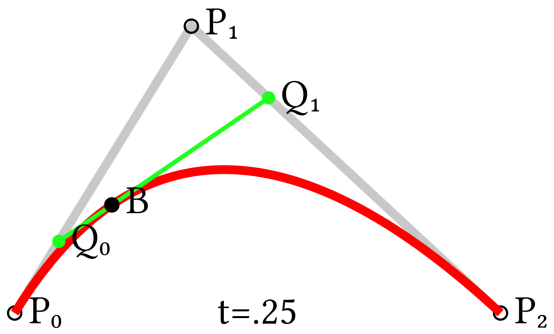
Kemudian, selain dua titik sebelumnya ditambahkan sebuah titik baru, sebut saja P2, dengan P0 dan P2 sebagai titik kontrol awal dan akhir, dan P1 menjadi titik kontrol antara. Dengan menyatakan titik Q1 terletak diantara garis yang menghubungkan P1 dan P2, dan membentuk kurva Bézier linier yang berbeda dengan kurva letak Q0 berada, maka dapat dinyatakan sebuah titik baru, R0 yang berada diantara garis yang menghubungkan Q0 dan Q1 yang bergerak membentuk kurva Bézier kuadratik terhadap titik P0 dan P2. Berikut adalah uraian persamaannya.

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

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

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

Berikut adalah ilustrasi dari kasus diatas.



Gambar 2. Pembentukan Kurva Bézier Kuadratik.
(Sumber: https://simonhalliday.com/2017/02/15/quadratic-bezier-curve-demo/)

1.2 Spesifikasi Program

Tugas ini berfokus pada pembuatan program yang dapat menghasilkan kurva Bézier dengan tingkat kelengkungan yang bisa diatur (linear, kuadrat, kubik, dan lain-lain). Program akan

menggunakan algoritma titik tengah berbasis divide and conquer untuk mencapai efisiensi dalam perhitungan.

1.2.1 Input

Format masukan dibebaskan, dengan catatan dijelaskan pada README dan laporan. Komponen yang perlu menjadi masukan yaitu

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

1.2.2 Output

Berikut adalah luaran dari program yang diekspektasikan.

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

1.2.3 Bonus

- Melakukan generalisasi algoritma, ide, serta melakukan implementasinya sehingga program dapat membentuk kurva Bézier kubik, kuartik, dan selanjutnya dengan 4, 5, 6, hingga n titik kontrol.
- Memberikan visualisasi proses pembentukan kurva, sehingga tidak hanya hasil akhirnya saja. Tentu saja proses visualisasinya perlu melibatkan Graphical User Interface (GUI) yang kakasnya dibebaskan.

BAB II
ANALISIS ALGORITMA

Pencarian solusi optimal dalam membangun bezier curve dalam tugas ini akan dilakukan dengan 2 pendekatan yaitu dengan pendekatan Brute Force dan pendekatan Divide and Conquer.

2.1 Analisis Algoritma Brute Force

Dalam Algoritma Brute Force yang kami gunakan, pada awalnya akan terdapat sebuah senarai titik kontrol yang didapatkan melalui input, dan sebuah nilai iterasi yang menandakan banyak iterasi yang akan dilakukan. Algoritma Brute Force ini akan melakukan perhitungan titik tengah sesuai dengan persamaan perhitungan titik Bezier Curve yang terdapat dalam spesifikasi yaitu:

$$\mathbf{B}(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i$$
$$= (1-t)^n \mathbf{P}_0 + \binom{n}{1}(1-t)^{n-1} t \mathbf{P}_1 + \cdots + \binom{n}{n-1}(1-t) t^{n-1} \mathbf{P}_{n-1} + t^n \mathbf{P}_n, \qquad 0 \leqslant t \leqslant 1$$

Gambar 3. Persamaan Titik Bezier Curve

Dikarenakan dalam kasus ini titik yang akan dicari dalam tiap iterasi adalah titik tengah, maka nilai t untuk persamaan diatas akan selalu digantikan oleh 0.5 untuk setiap iterasinya, hanya saja untuk mempermudah perhitungan, nilai t akan diubah menjadi $1 / 2^{(iterasi)} - 1$ sehingga dengan meningkatnya nilai iterasi titik tengan dalam iterasi selanjutnya juga akan diperhitungkan.

- **Perulangan 1:** Pada awalnya didapatkan sebuah nilai t yaitu $1 / 2^{(iterasi)}$, untuk tiap perulangan nilai dari t sekarang akan ditambahkan nilai dari t awal, dan perulangan akan berhenti ketika nilai t mencapai 1, sehingga perulangan yang akan dilakukan adalah sebanyak $2^{(iterasi)} - 1$ kali. Dengan k adalah jumlah iterasi maka perulangan ini memiliki kompleksitas $O(2^k - 1)$.
- **Perulangan 2:** Untuk setiap perulangan 1, akan dihitung koordinat titik sesuai dengan jumlah iterasi yang dilakukan dengan cara melakukan perulangan sebanyak jumlah titik, sesuai dengan persamaan titik Bezier Curve. Dengan n adalah jumlah titik kontrol maka perulangan ini memiliki kompleksitas $O(n)$.

Untuk setiap perulangan 2, titik yang telah dihitung akan dimasukkan dalam senarai hasil, yang mana senarai hasil ini nantinya akan diproses untuk menampilkan visualisasi Bezier Curve yang diapatkan.

Secara keseluruhan Algoritma Brute Force ini memiliki kompleksitas sebesar $O(n(2^k - 1))$.

2.2 Analisis Algoritma Divide and Conquer

Divide and conquer adalah strategi algoritmik yang memecah masalah kompleks menjadi sub-masalah yang lebih kecil dan serupa. Sub-masalah ini dipecah-pecah lagi secara rekursif sampai ukurannya cukup mudah dipecahkan secara langsung. Solusi sub-masalah kemudian digabungkan untuk memberikan solusi terhadap masalah aslinya

Pada permasalahan pencarian titik-titik Kurva Bezier, algoritma Divide and Conquer dapat diterapkan sebagai berikut

1. Membagi (Divide): Sebuah fungsi dapat dipanggil secara rekursif untuk membagi kumpulan titik kontrol asli menjadi segmen-segmen yang lebih kecil. Pada setiap iterasi/langkah, fungsi tersebut menghitung titik tengah antara pasangan titik yang berurutan, menciptakan kumpulan titik baru yang mewakili segmen Bezier yang lebih kecil.
2. Menaklukkan (Conquer): Ketika sub-masalah menjadi cukup sederhana (yaitu, ketika iterasi saat ini sudah mencapai iterasi yang dibutuhkan), titik tengah dikembalikan secara langsung ke saat fungsi dipanggil (kasus dasar rekursi).
3. Menggabungkan (Combine): Garis-garis yang terbentuk dari titik-titik tengah ini kemudian digunakan untuk membangun perkiraan kurva Bezier yang halus dalam fungsi yang dipanggil pertama. Setiap iterasi akan menghasilkan kumpulan titik yang lebih banyak, sehingga memberikan representasi kurva yang lebih halus dan detail.

Dalam Algoritma Divide and Conquer yang kami gunakan, akan terdapat sebuah senarai yang berisi titik - titik kontrol yang didapatkan dari input user, nilai iterasi sekarang, dan nilai iterasi maksimum, yaitu nilai iterasi yang diinginkan user. Algoritma Divide and Conquer yang kami gunakan akan menggunakan pendekatan rekursi.

- **Basis:** Iterasi pada proses sekarang lebih besar atau sama dengan nilai iterasi maksimal yang diinginkan user, jika syarat ini terpenuhi, maka fungsi akan mengembalikan nilai dari senarai titik kontrol.
- **Rekurens:** Iterasi pada proses sekarang lebih kecil dari nilai iterasi maksimal yang diinginkan user, jika syarat ini terpenuhi, maka rekurens akan tereksekusi. Rekurens pada fungsi ini adalah sebagai berikut.
    a. **Langkah 1 (Perhitungan Titik Tengah Control Points)**
    Langkah ini bertujuan untuk mendapatkan titik tengah dari titik - titik kontrol yang digunakan. Hal ini dilakukan menggunakan persamaan perhitungan titik tengah dari Bezier Curve yang telah diberikan sebelumnya, dengan melakukan loop sebanyak n kali dengan n adalah jumlah titik kontrol sehingga memiliki kompleksitas $O(n)$.
    b. **Langkah 2 (Perhitungan Titik - Titik Tengah Bagian Kiri dan Kanan)**
    Langkah ini bertujuan untuk mendapatkan seluruh titik-titik tengah yang digunakan untuk mendapatkan titik tengah yang digunakan pada langkah pertama, dan membaginya menjadi titik - titik tengah kiri, dan titik - titik - tengah kanan, proses ini

adalah proses **DIVIDE** dalam Algoritma Divide and Conquer ini. Dengan melakukan loop sebanyak n kali di mana n adalah jumlah titik kontrol awal, untuk setiap loop ini, akan dilakukan loop sebanyak i kali di mana i adalah nilai iterasi dalam loop sebelumnya untuk menghitung titik - titik tengah yang digunakan. Hal ini diperlukan dikarenakan titik - titik tengah yang digunakan pada langkah 1 adalah hasil dari perhitungan titik - titik kontrol dengan jumlah yang berbeda beda, mulai dari 2 hingga n - 1 titik. Hasil dari langkah ini adalah dua senarai yang merepresentasikan titik - titik tengah pada kiri kurva dan pada kanan kurva. Kompleksitas pada langkah ini adalah $O(n)$.

c. **Langkah 3 (Perhitungan Rekurens)**
   Langkah ini bertujuan untuk mendapatkan titik untuk iterasi selanjutnya, selama iterasi sekarang masih lebih kecil dari iterasi yang user inginkan. Bagian rekurens ini akan menggunakan fungsi awal dengan parameter yang diganti, yaitu senarai dengan dengan titik tengah bagian kiri kurva, dan titik tengah bagian kanan kurva. Hasil dari bagian ini adalah senarai titik tengah kiri dan titik tengah kanan yang telah diproses sesuai dengan jumlah iterasi yang diinginkan user. Kompleksitas pada langkah ini adalah $T(n,\ k)\ =\ 1$ untuk k = 1 dan $T(n,\ k)\ =\ 2^k$ untuk k > 1 sehingga memiliki kompleksitas $O(2^k)$.
d. **Langkah 4 (Penggabungan Hasil)**
   Pada langkah ini, kita telah memiliki 3 buah senarai, yaitu senarai titik tengah dari keseluruhan titik kontrol, senarai titik tengah bagian kiri sesuai jumlah iterasi user, dan senarai titik tengah bagian kanan sesuai jumlah iterasi user. Ketiga senarai ini pun digabungkan menjadi sebuah senarai yang adalah hasil dari pencarian titik pada Bezier Curve. Bagian ini adalah bagian **COMBINE** dalam Algoritma Divide and Conquer kami. Kompleksitas pada langkah ini adalah $O(n)$.

Secara keseluruhan Algoritma Divide and Conquer ini memiliki kompleksitas $O(n^2 2^k)$.

BAB III
SOURCE CODE PROGRAM

3.1 Algoritma Brute Force

Algoritma pada Bab 2 diimplementasikan dalam bahasa JavaScript. Program Brute Force terletak dalam 1 file BezierBF.js yang melingkupi seluruh fungsi yang diperlukan untuk menyelesaikan permasalahan secara Brute Force.

1. Fungsi Pembantu

```
function makePoints(arrayX, arrayY) {
    let arr = [];
    for (let i = 0; i < arrayX.length; i++) {
      arr.push([arrayX[i], arrayY[i]]);
    }
    return arr;
  }

function pascalRow(order, index) {
    let row = [];
    row[0] = 1;
    for (let i = 1; i <= order; i++) {
      for (let j = i - 1; j > 0; j--) {
        row[j] = row[j] + row[j - 1];
      }
      row.push(1);
    }
    return row[index];
  }

function PascalControlsx(Ctrls, thedi) {
    let n = 0;
    for (let k = 0; k < Ctrls.length; k++) {
      n +=
        (1 - thedi) ** (Ctrls.length - k - 1) *
        thedi ** k *
        Ctrls[k][0] *
        pascalRow(Ctrls.length - 1, k);
    }
    return n;
  }

  function PascalControlsy(Ctrls, thedi) {
```

```
    let n = 0;
    for (let k = 0; k < Ctrls.length; k++) {
      n +=
        (1 - thedi) ** (Ctrls.length - k - 1) *
        thedi ** k *
        Ctrls[k][1] *
        pascalRow(Ctrls.length - 1, k);
    }
    return n;
  }
```

Fungsi makePoints adalah fungsi yang digunakan untuk melakukan konversi yang mengembalikan senarai berisi koordinat titik dalam bentuk [[x, y], [x, y], …] yang dikonversi dari bentuk [x, x, x, x, …] dan [y, y, y, y, ….].

Fungsi pascalRow adalah fungsi yang dapat mengembalikan nilai dari sebuah baris dalam order tertentu dalam segitiga paskal. Dalam fungsi ini, order berlaku sebagai order ke berapa baris paskal yang diinginkan, dan index adalah nilai dari index ke berapa dengan 0 menyatakan indeks pertama.

Fungsi PascalControlsx dan PascalControlsy adalah fungsi untuk melakukan perhitungan titik tengah dari sebuah senarai berisi titik - titik kontrol. Kedua fungsi ini menerapkan persamaan yang digunakan untuk menghitung titik tengah sebuah Bezier Curve yang telah diberikan sebelumnya. Parameter thedi pada kasus ini adalah nilai t pada persamaan tersebut.

2. Fungsi Brute Force Utama

```
function BezierBF(Iterations, Inputx, Inputy) {
  let Fin = [];

  let Controls = makePoints(Inputx, Inputy);

  Fin.push(Controls[0]);

  let x = 0;
  let y = 0;
  let di = 1 / 2 ** (Iterations);
  let dianch = di;
  while (di < 1) {
    x = PascalControlsx(Controls, di);
    y = PascalControlsy(Controls, di);
    Fin.push([x, y]);
    di += dianch;
```

```
  }
  Fin.push(Controls[Controls.length - 1]);
  return Fin;
}

export default BezierBF;
```

Fungsi utama ini memiliki 3 parameter, yang pertama adalah Iterasions yang merepresentasikan jumlah iterasi yang diinginkan user, yang kedua adalah Inputx, yang merepresentasikan senarai koordinat titik kontrol x, dan yang ketiga adalah Inputy yang merepresentasikan senarai koordinat titik kontrol y.

Fungsi ini kemudian membentuk Controls yang adalah senarai koordinat titik kontrol, dan sebuah variabel di, yang merepresentasikan nilai t yang akan digunakan. Nilai di ini adalah $1 - 2^{Iterations}$. Selanjutnya fungsi akan melakukan loop sebanyak $2^{Iterations} - 1$ dengan variabel di dimulai dari awal, ditambahkan dengan nilai awal di, dan diakhiri dengan nilai akhirnya sebelum 1. Untuk setiap perulangan ini, fungsi akan menghitung letak titik dengan persamaan titik Bezier Curve dengan nilai di sebagai nilai t nya. Koordinat yang telah dihitung pun akan dimasukkan ke dalam sebuah array Fin, dan fungsi akan mengembalikan Fin yang telah berisi seluruh titik yang akan dilalui Bezier Curve.

3.2 Algoritma Divide and Conquer

Algoritma pada Bab 2 diimplementasikan dalam bahasa JavaScript. Program Divide and Conquer terletak dalam 1 file BezierLogicv2.js yang melingkupi seluruh fungsi yang diperlukan untuk menyelesaikan permasalahan secara Divide and Conquer.

1. **Fungsi Pembantu**

```
function makePoints(arrayX, arrayY) {
    let arr = [];
    for (let i = 0; i < arrayX.length; i++) {
      arr.push([arrayX[i], arrayY[i]]);
    }
    return arr;
  }

function pascalRow(order, index) {
    let row = [];
    row[0] = 1;
    for (let i = 1; i <= order; i++) {
      for (let j = i - 1; j > 0; j--) {
        row[j] = row[j] + row[j - 1];
```

```
      }
      row.push(1);
    }
    return row[index];
  }

function PascalControlsx(Ctrls, thedi) {
    let n = 0;
    for (let k = 0; k < Ctrls.length; k++) {
      n +=
        (1 - thedi) ** (Ctrls.length - k - 1) *
        thedi ** k *
        Ctrls[k][0] *
        pascalRow(Ctrls.length - 1, k);
    }
    return n;
  }

function PascalControlsy(Ctrls, thedi) {
    let n = 0;
    for (let k = 0; k < Ctrls.length; k++) {
      n +=
        (1 - thedi) ** (Ctrls.length - k - 1) *
        thedi ** k *
        Ctrls[k][1] *
        pascalRow(Ctrls.length - 1, k);
    }
    return n;
  }

function LeftMids(Ctrls) {
    let arr = [];
    for (let i = 1; i < Ctrls.length - 1; i++) {
      arr.unshift([
        PascalControlsx(Ctrls.slice(0, Ctrls.length - i), 0.5),
        PascalControlsy(Ctrls.slice(0, Ctrls.length - i), 0.5),
      ]);
    }
    arr.unshift(Ctrls[0]);
    return arr;
  }

function RightMids(Ctrls) {
    let arr = [];
    for (let i = 1; i < Ctrls.length - 1; i++) {
```

```
      arr.push([
        PascalControlsx(Ctrls.slice(i), 0.5),
        PascalControlsy(Ctrls.slice(i), 0.5),
      ]);
    }
    arr.push(Ctrls[Ctrls.length - 1]);
    return arr;
  }

function EnterLeft(arr, entry) {
    for (let i = entry.length - 1; i >= 0; i--) {
      arr.unshift(entry[i]);
    }
  }

function EnterRight(arr, entry) {
    for (let i = 0; i < entry.length; i++) {
      arr.push(entry[i]);
    }
  }
```

Fungsi makePoints adalah fungsi yang digunakan untuk melakukan konversi yang mengembalikan senarai berisi koordinat titik dalam bentuk [[x, y], [x, y], …] yang dikonversi dari bentuk [x, x, x, x, …] dan [y, y, y, y, ….].

Fungsi pascalRow adalah fungsi yang dapat mengembalikan nilai dari sebuah baris dalam order tertentu dalam segitiga paskal. Dalam fungsi ini, order berlaku sebagai order ke berapa baris paskal yang diinginkan, dan index adalah nilai dari index ke berapa dengan 0 menyatakan indeks pertama.

Fungsi PascalControlsx dan PascalControlsy adalah fungsi untuk melakukan perhitungan titik tengah dari sebuah senarai berisi titik - titik kontrol. Kedua fungsi ini menerapkan persamaan yang digunakan untuk menghitung titik tengah sebuah Bezier Curve yang telah diberikan sebelumnya. Parameter thedi pada kasus ini adalah nilai t pada persamaan tersebut.

Fungsi LeftMids dan RightMids adalah fungsi yang menerima parameter titik - titik kontrol, dan akan mengembalikan seluruh titik tengah bagian kiri dan kanan dari titik tengah utama titik - titik kontrol. Titik tengah bagian kiri dan kanan yang dimaksud adalah titik tengah akhir yang digunakan untuk membentuk titik tengah utama.

Fungsi EnterLeft dan EnterRight adalah prosedur yang digunakan untuk melakukan penyambungan 2 senarai dari arah kiri dan arah kanan.

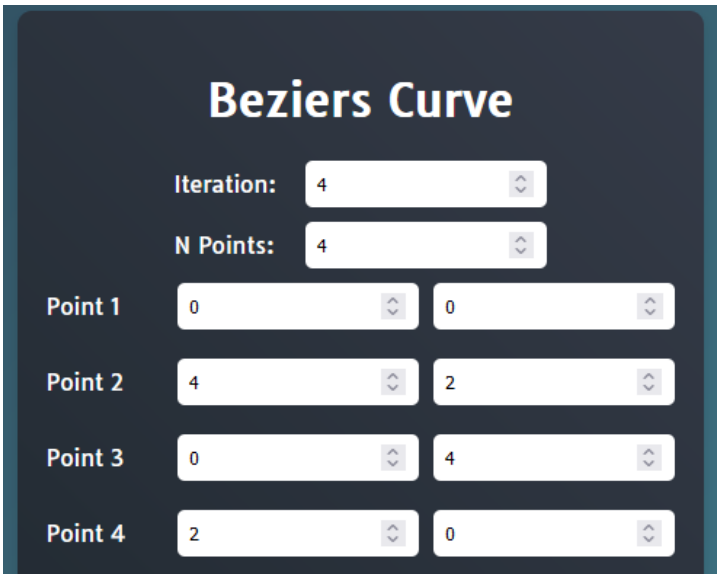
2. Fungsi Utama

```
function recFind(Ctrls, Iter, MaxIter) {
    if (Iter < MaxIter) {
      let mid = [[PascalControlsx(Ctrls, 0.5), PascalControlsy(Ctrls,
0.5)]];
      let midsl = LeftMids(Ctrls);
      let midsr = RightMids(Ctrls);

      midsl.push(mid[0]);
      midsr.unshift(mid[0]);

      if (Iter + 1 < MaxIter) {
        let lefts = recFind(midsl, Iter + 1, MaxIter);
        let rights = recFind(midsr, Iter + 1, MaxIter);
        EnterLeft(mid, lefts);
        EnterRight(mid, rights);

        return mid;
      } else {
        return mid;
      }
    } else {
      return Ctrls;
    }
  }
```

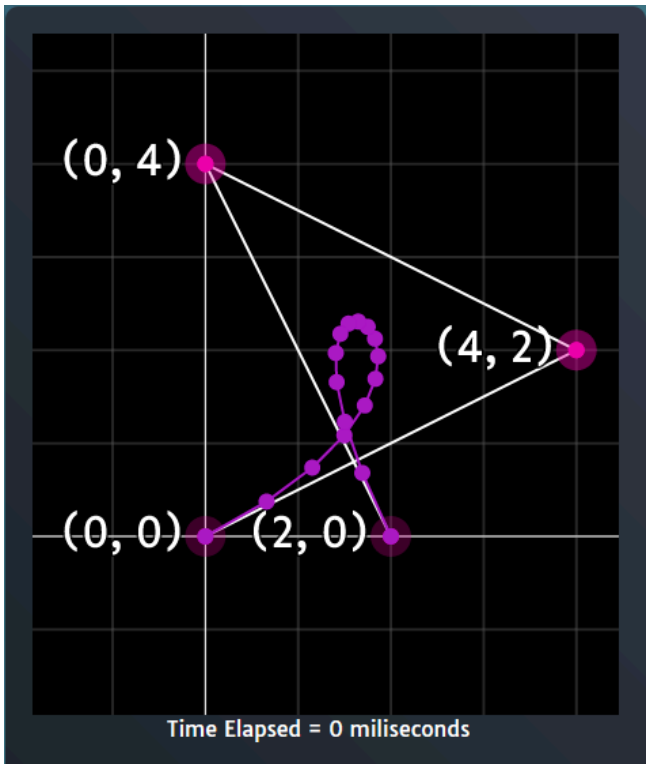
Fungsi utama ini memiliki 3 parameter, parameter yang pertama adalah titik - titik kontrol yang digunakan, parameter yang kedua adalah iterasi yang sedang digunakan, dan parameter ketiga adalah iterasi maksimal sesuai keinginan user. Sesuai dengan penjelasan pada Bab 2, fungsi ini memiliki basis yaitu jika iterasi saat ini lebih besar atau sama dengan iterasi maksimal, maka fungsi ini akan mengembalikan titik kontrol yang digunakan, jika tidak, maka akan memasuki bagian rekurens. Bagian rekurens dari fungsi ini akan membentuk sebuah titik tengah utama dari seluruh titik kontrol yang digunakan, lalu dengan fungsi LeftMids dan RightMids, akan diambil titik tengah yang digunakan untuk membentuk titik tengah utama, di sisi kiri dan sisi kanan dari titik tengah utama. Fungsi kemudian akan melakukan rekurens, dengan memanggil fungsi recFind kembali dengan titik tengah kiri dan kanan sebagai titik kontrol dan iterasi yang telah ditambah 1. Setelah itu fungsi akan menggabungkan seluruh titik yang didapatkan kembali ke titik tengah, dan mengembalikan senarai yang telah berisi seluruh titik - titik yang akan dilalui Bezier Curve.

BAB IV
EKSPERIMEN

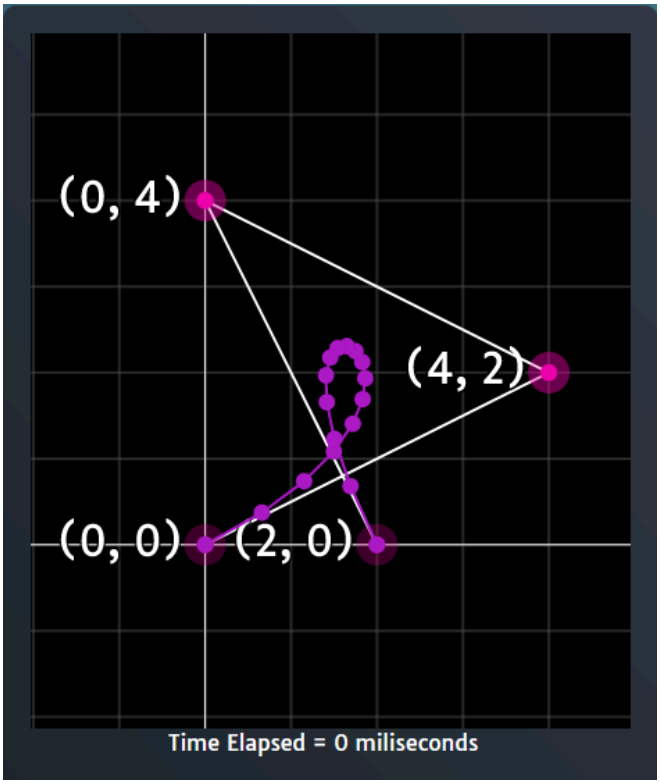
4.1 Tes 1



Gambar 4. Input Tes 1

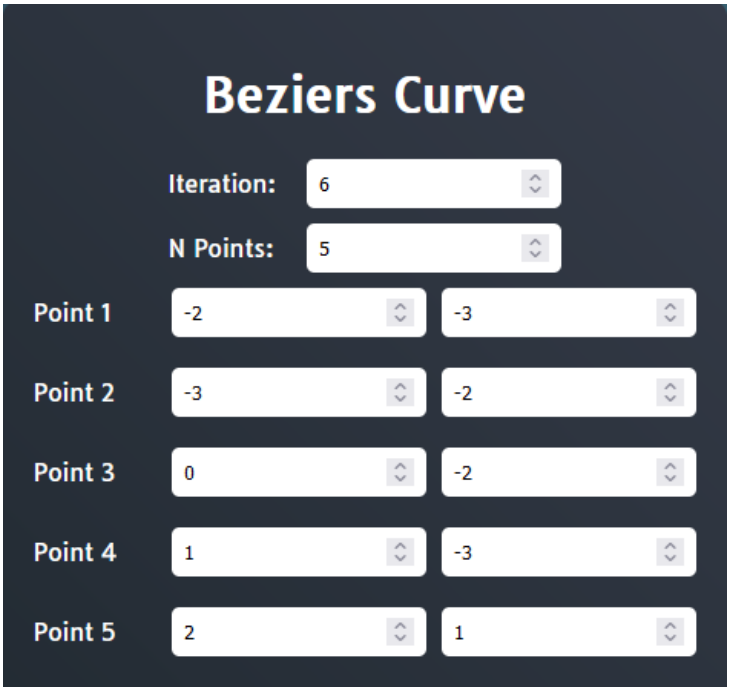


Gambar 5. Tes 1 dengan DnC

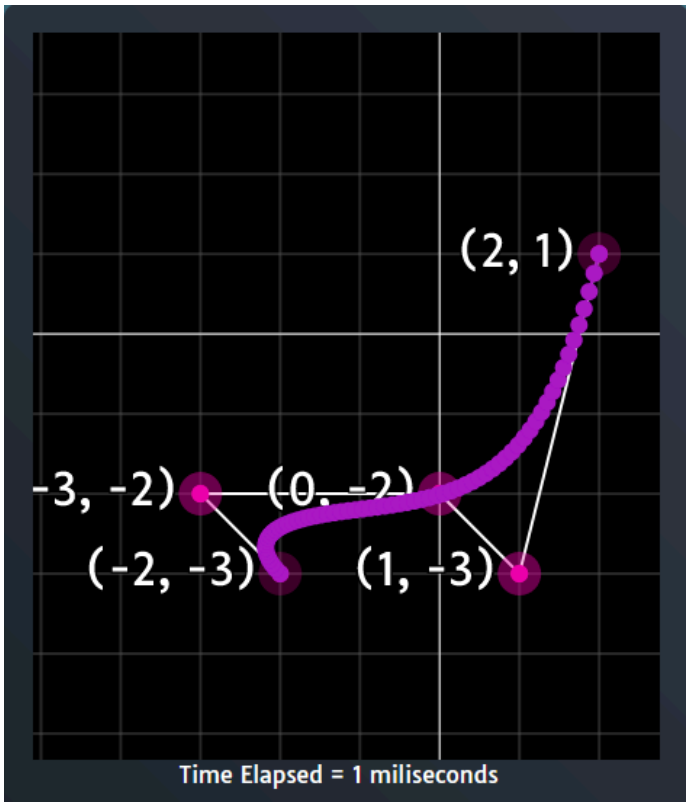


Gambar 6. Tes 1 dengan Brute Force

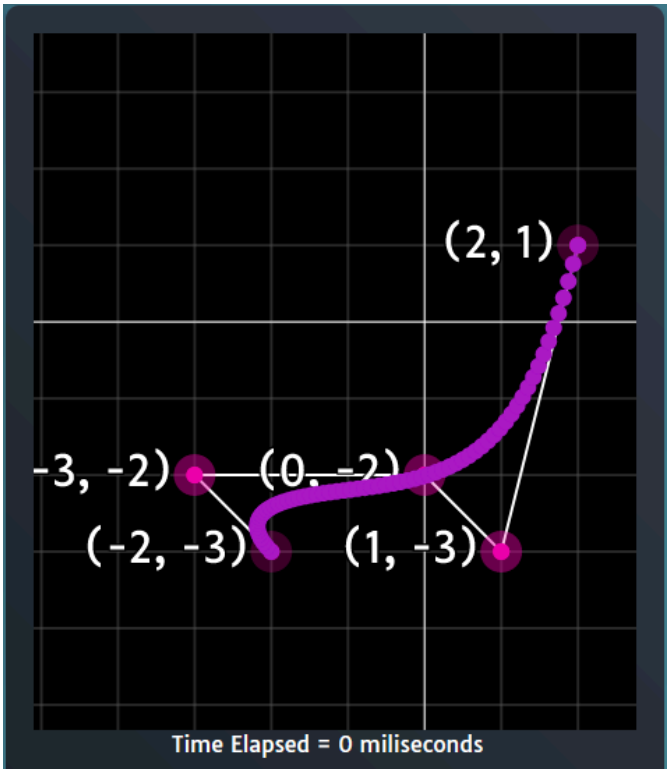
4.2 Tes 2



Gambar 7. Input Tes 2

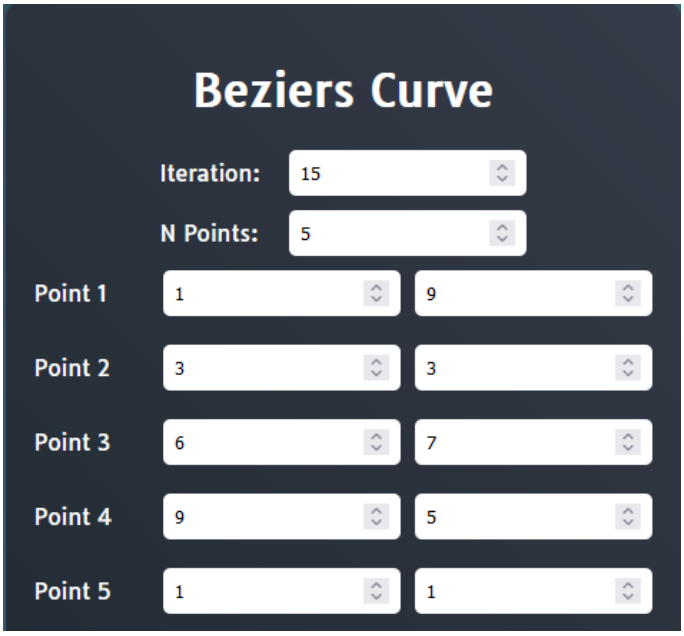


Gambar 8. Tes 2 dengan DnC

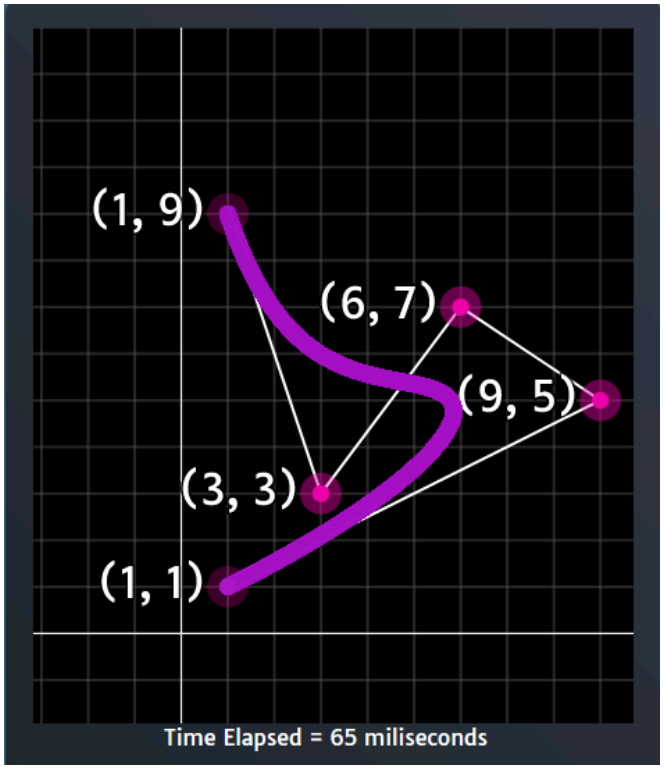


Gambar 9. Tes 2 dengan Brute Force

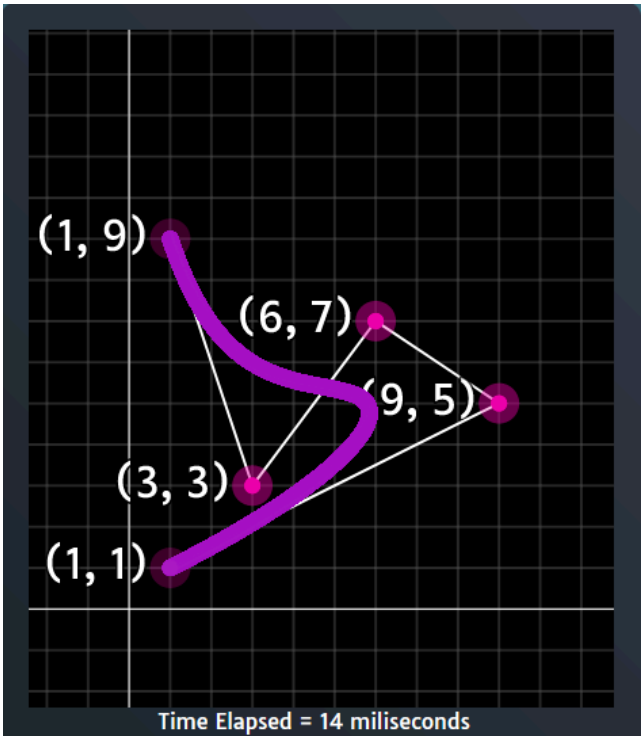
4.3 Tes 3



Gambar 10. Input Tes 3

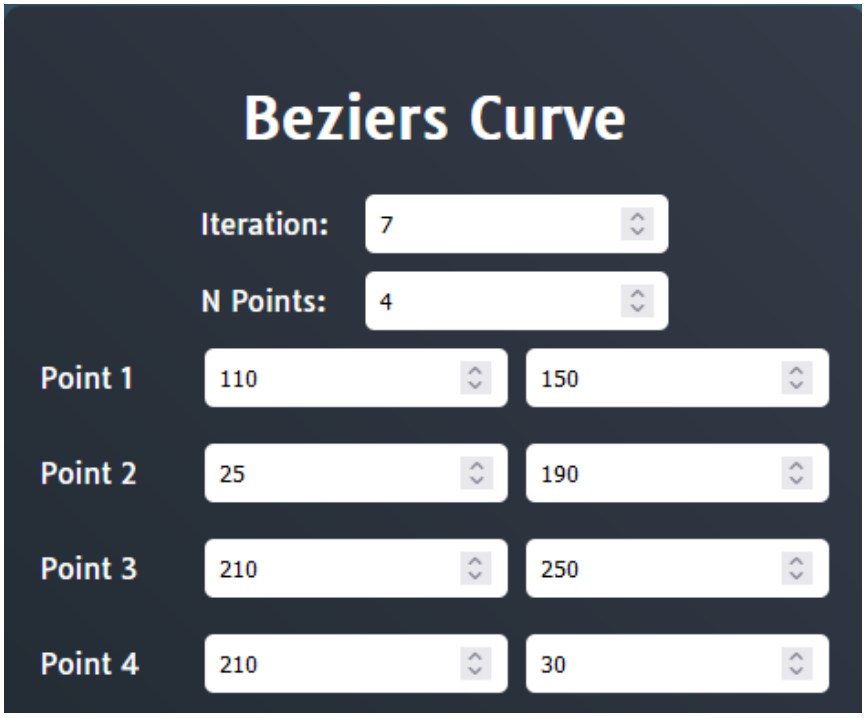


Gambar 11. Tes 3 dengan DnC

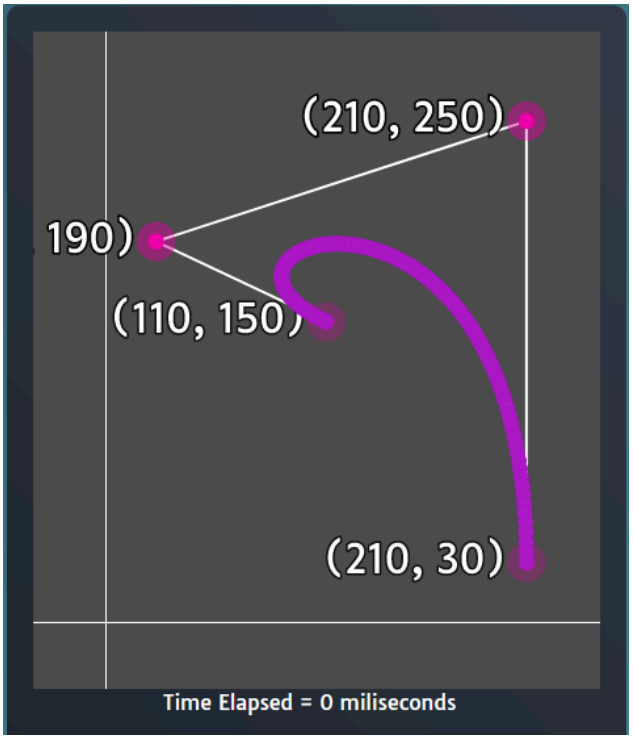


Gambar 12. Tes 3 dengan Brute Force

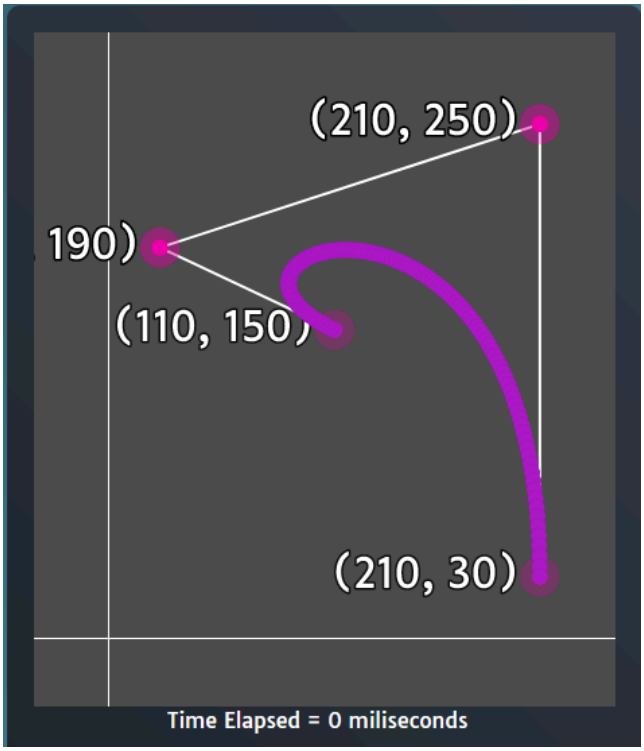
4.4 Tes 4



Gambar 13. Input Tes 4

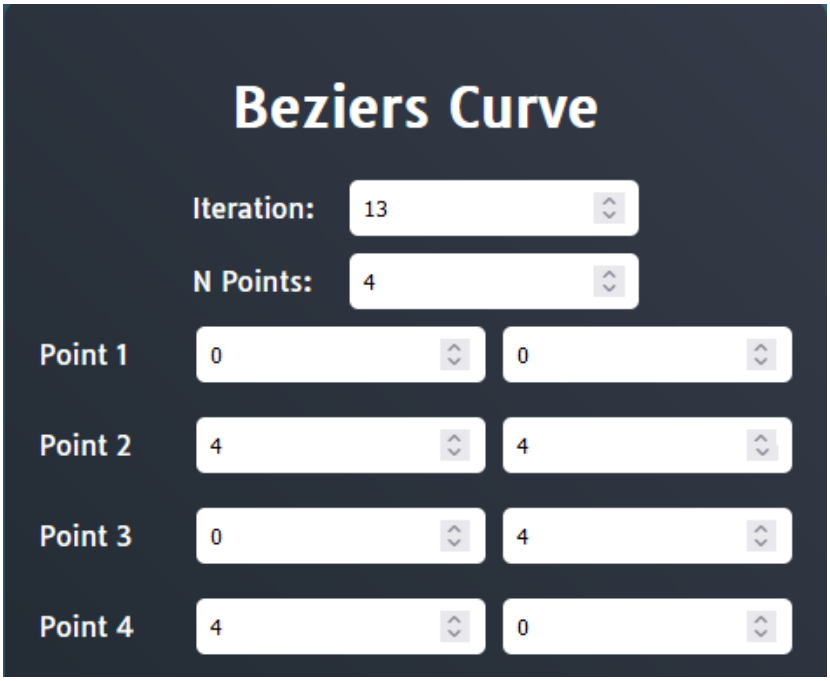


Gambar 14. Tes 4 dengan DnC

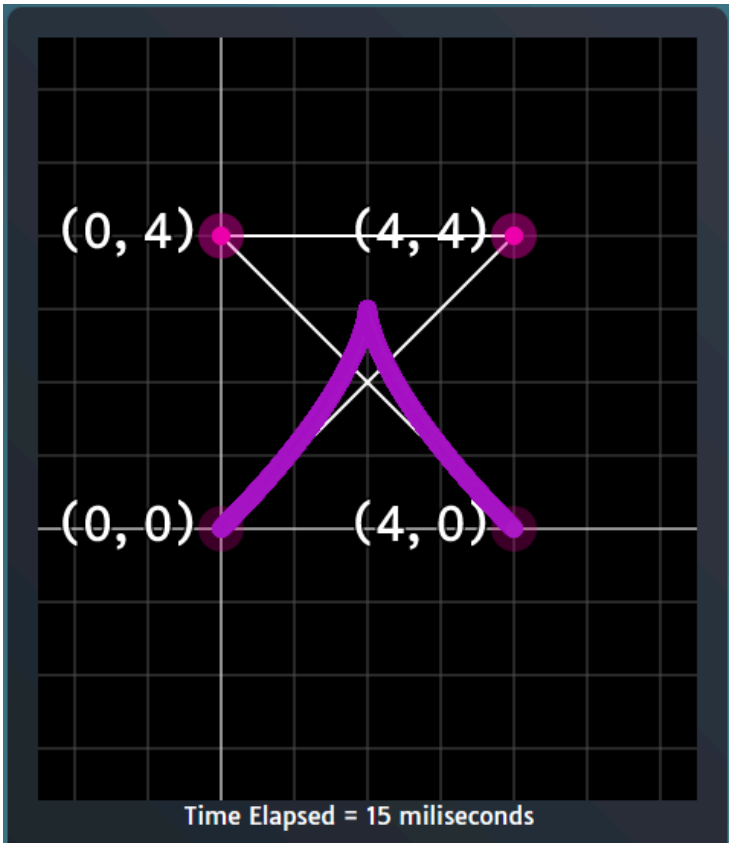


Gambar 15. Tes 4 dengan Brute Force

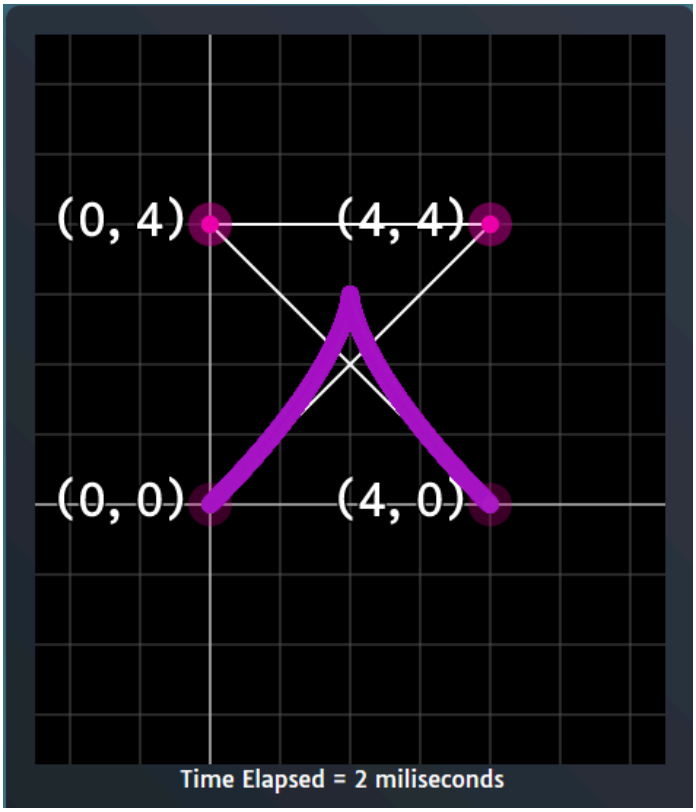
4.5 Tes 5



Gambar 16. Input Tes 5

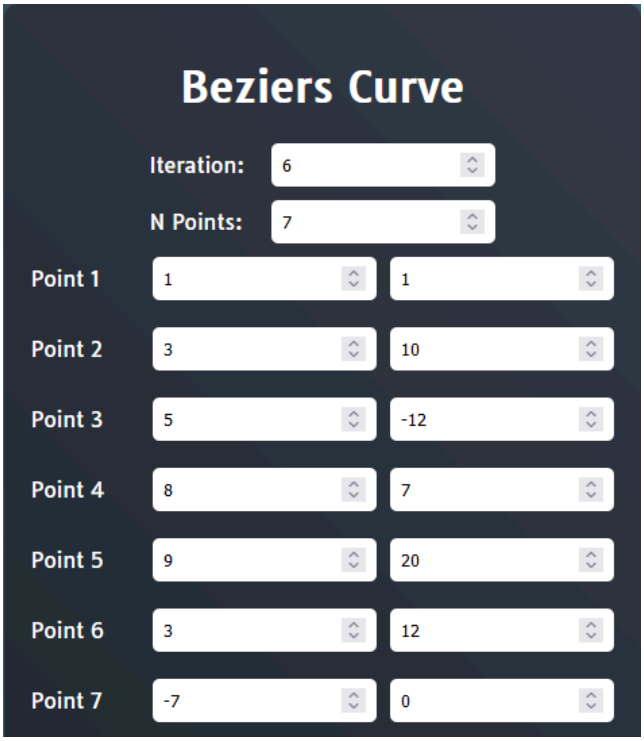


Gambar 17. Tes 5 dengan DnC

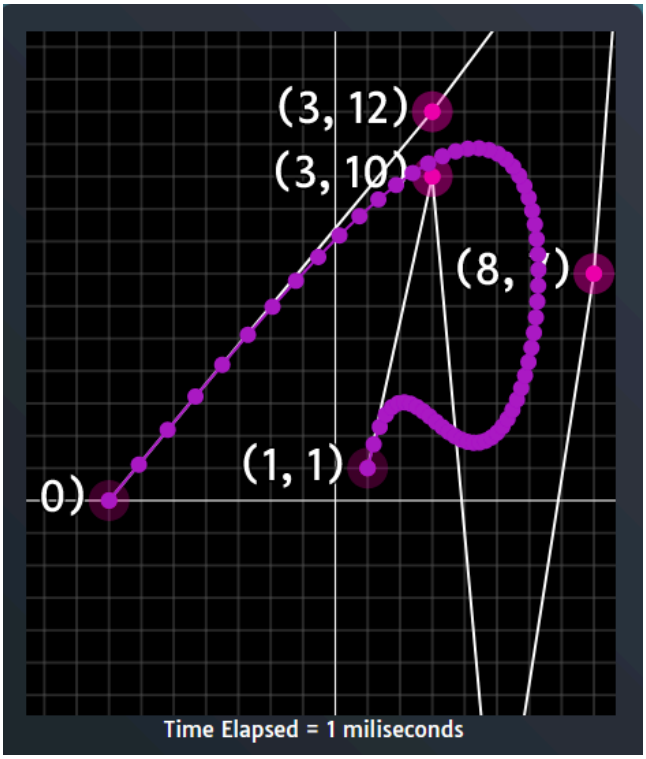


Gambar 18. Tes 5 dengan Brute Force

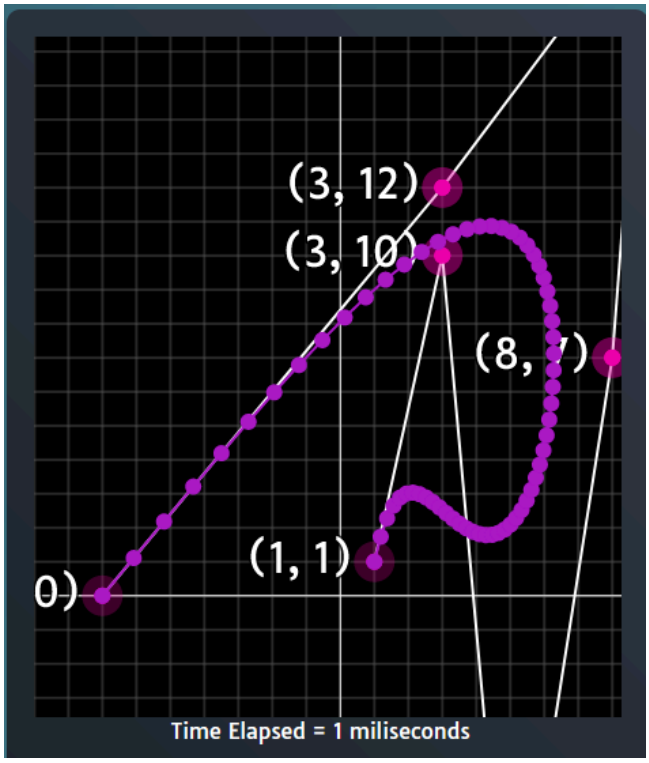
4.6 Tes 6



Gambar 19. Input Tes 6



Gambar 20. Tes 6 dengan DnC



Gambar 21. Tes 6 dengan Brute Force

BAB V
ANALISIS HASIL BF vs DNC

Berdasarkan hasil yang kami dapatkan dari 6 test case, hasil titik - titik yang kami dapatkan dari algoritma Brute Force dan Divide and Conquer adalah sama. Perbedaan yang paling utama adalah waktu yang diperlukan untuk memproses kedua algoritma. Dapat dilihat, terutama dalam iterasi yang cukup besar, algoritma Brute Force memiliki waktu proses yang lebih rendah dibandingkan algoritma Divide and Conquer. Berdasarkan analisis kami, algoritma Divide and Conquer memiliki waktu proses yang lebih lama dikarenakan dalam penerapan algoritma Divide and Conquer terlibat banyak pemotongan senarai dan penggabungan senarai. Dalam setiap rekursi yang digunakan akan terdapat beberapa pembentukan senarai, penggabungan senarai dengan rekursi sebelumnya, dan pemotongan senarai untuk beberapa iterasi. Sedangkan dalam algoritma Brute Force, penggabungan dan pemotongan senarai tidak diterapkan, yang diterapkan hanyalah perhitungan normal dengan persamaan Bezier Curve biasa, dan menambahkan nilai koordinat yang didapatkan dalam 1 array yang pada akhirnya akan digunakan. Maka untuk itu, algoritma Divide and Conquer akan kalah cepat dibandingkan algoritma Brute Force.

LAMPIRAN

Repository Github

Link repository : https://github.com/TazakiN/Tucil2_13522032_13522043

Tabel Spesifikasi

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.		✓