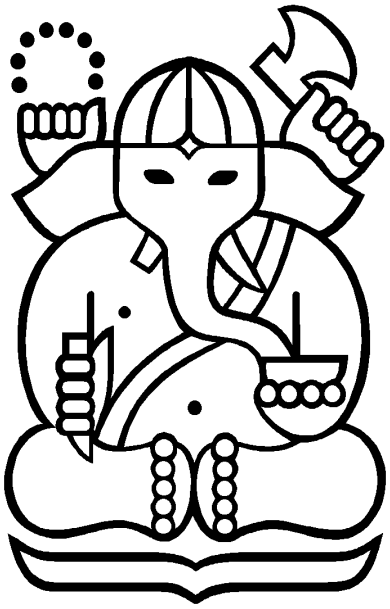
Tugas Kecil 2 IF2211 Strategi Algoritma

Semester II tahun 2023/2024

Membangun Kurva Bézier dengan Algoritma Titik Tengah berbasis Divide and Conquer



Oleh :

Muhammad Zaki 13522136

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA INSTITUT TEKNOLOGI BANDUNG TAHUN AJARAN 2023/2024

BAB I

Landasan Teori

1.1 Kurva Bezier

Kurva Bézier adalah kurva halus yang sering digunakan dalam desain grafis, animasi, dan manufaktur. Kurva ini dibuat dengan menghubungkan beberapa titik kontrol, yang menentukan bentuk dan arah kurva. Cara membuatnya cukup mudah, yaitu dengan menentukan titik-titik kontrol dan menghubungkannya dengan kurva. Kurva Bézier memiliki banyak kegunaan dalam kehidupan nyata, seperti pen tool, animasi yang halus dan realistis, membuat desain produk yang kompleks dan presisi, dan membuat font yang indah dan unik. Keuntungan menggunakan kurva Bézier adalah kurva ini mudah diubah dan dimanipulasi, sehingga dapat menghasilkan desain yang presisi dan sesuai dengan kebutuhan.

Sebuah kurva Bézier didefinisikan oleh satu set titik kontrol P0 sampai Pn , dengan n disebut order (n = 1 untuk linier, n = 2 untuk kuadrat, dan seterusnya). Titik kontrol pertama dan terakhir selalu menjadi ujung dari kurva, tetapi titik kontrol antara (jika ada) umumnya tidak terletak pada kurva. Pada gambar 1 diatas, titik kontrol pertama adalah P0, sedangkan titik kontrol terakhir adalah P3. Titik kontrol P1 dan P2 disebut sebagai titik kontrol antara yang tidak terletak dalam kurva yang terbentuk. Mengulas lebih jauh mengenai bagaimana sebuah kurva Bézier bisa terbentuk, misalkan diberikan dua buah titik P0 dan P1 yang menjadi titik kontrol, maka kurva Bézier yang terbentuk adalah sebuah garis lurus antara dua titik. Kurva ini disebut dengan kurva Bézier linier. Misalkan terdapat sebuah titik Q0 yang berada pada garis yang dibentuk oleh P0 dan P1, maka posisinya dapat dinyatakan dengan persamaan parametrik berikut.

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

dengan t dalam fungsi kurva Bézier linier menggambarkan seberapa jauh B(t) dari P0 ke P1. Misalnya ketika t = 0.25, maka B(t) adalah seperempat jalan dari titik P0 ke P1. sehingga seluruh rentang variasi nilai t dari 0 hingga 1 akan membuat persamaan B(t) membentuk sebuah garis lurus dari P0 ke P1.

1.2 Algoritma Brute Force

Algoritma *brute force* adalah salah satu algoritma yang digunakan untuk menyelesaikan persoalan komputasi. Pendekatan yang dilakukan oleh algoritma *brute force* adalah secara *straight forward* atau secara langsung. Biasanya algoritma ini digunakan untuk menyelesaikan persoalan yang sederhana, langsung, dan jelas persoalannya. Contoh penggunaan algoritma *brute force* adalah untuk mencari elemen terbesar/terkecil pada senarai, mencari elemen yang ada pada senarai, menghitung faktorial, dan lain sebagainya, termasuk untuk menyelesaikan persoalan perkalian polinomial.

1.3 Algoritma Divide and Conquer

Algoritma *divide and conquer* adalah salah satu strategi algoritma yang menyelesaikan persoalan dengan cara membagi persoalan yang besar menjadi persoalan yang lebih kecil, sehingga lebih mudah diselesaikan. Langkah yang dilakukan dalam algoritma *divide and conquer* ada 3 yaitu :

- *Divide* : membagi persoalan besar menjadi sub-persoalan yang lebih kecil
- *Conquer* : menyelesaikan sub-persoalan dengan konsep rekursif
- *Combine* : menggabungkan solusi masing-masing sub-persoalan, sehingga membentuk kembali solusi semula

Biasanya objek yang dibagi menjadi sub-persoalan adalah berupa larik (*array*) berukuran n. Tiap sub-persoalan memiliki karakteristik yang sama dengan persoalan awal, sehingga sub-solusi dapat digunakan untuk memecahkan persoalan awal. Maka dari itu solusi dari algoritma *divide and conquer* dapat dicari dengan cara rekursif. Persoalan yang dapat diselesaikan oleh algoritma *divide and conquer* antara lain perpangkatan, mengurutkan larik (*quick sort, merge sort, insertion sort,* dan *selection sort*), dan perkalian polinom.

BAB II
ANALISIS DAN IMPLEMENTASI

2. 1 Algoritma Brute – Force

Brute Force yang saya gunakan disini adalah metode brute – force dengan menggunakan rumus untuk membentuk kurva kuadratik bezier yaitu :

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

Ide nya adalah untuk menginterasi setiap t dimana t merupakan nilai yang mempunyai range dari 0 sampai dengan 1. Iterasi melalui setiap nilai t memungkinkan pembentukan titik-titik pada kurva Bézier dengan resolusi yang sesuai dengan jumlah iterasi yang ditentukan oleh pengguna. Dengan cara ini, titik-titik pada kurva Bézier kuadratik dapat dihasilkan dengan pendekatan brute-force.

Implementasi Brute - Force :

```
import numpy as np
import matplotlib.pyplot as plt
from Point import Point
from showCurve import showCurve
import time

# BRUTE FORCE
def quadratic_bezier(p0: Point, p1: Point, p2: Point, t_values):
        return [Point((1 - t) * (1 - t) * p0.x + 2 * (1 - t) * t *
p1.x + t * t * p2.x,
                      (1 - t) * (1 - t) * p0.y + 2 * (1 - t) * t *
p1.y + t * t * p2.y) for t in t_values]

def bruteForce():
    P = []
    for i in range(3):
        while True:
            try:
```

```
                    x, y = map(float, input("Masukkan koordinat x dan y
titik ke-{} : ".format(i)).split())
                    P.append(Point(x, y))
                    break
                except ValueError:
                    print("Invalid input. Please enter valid numerical
values.")

        iterations = int(input("Masukkan jumlah iterasi: "))

        start_time = time.perf_counter()
        curve_points = quadratic_bezier(P[0], P[1], P[2], t_values)
        end_time = time.perf_counter()
        elapsed_time = end_time - start_time
        print("Waktu eksekusi:", elapsed_time * 1000, "ms")
        showCurve(P, curve_points,i)
        input("Press anything to continue...")
```

Untuk contoh misal saya mempunyai 3 titik kontrol yaitu pada titik (1,1) , (5,5) , (8,3). Dengan iterasi 1 , kita akan mendapat kan titik di (4.75,3.5)

$$P(0.5) = (1-0.5)^2 \cdot (1,1) + 2(1-0.5)0.5 \cdot (5,5) + 0.5^2 \cdot (8,3)$$

$$P(0.5) = (0.5)^2 \cdot (1,1) + 2 \cdot 0.5 \cdot 0.5 \cdot (5,5) + (0.5)^2 \cdot (8,3)$$

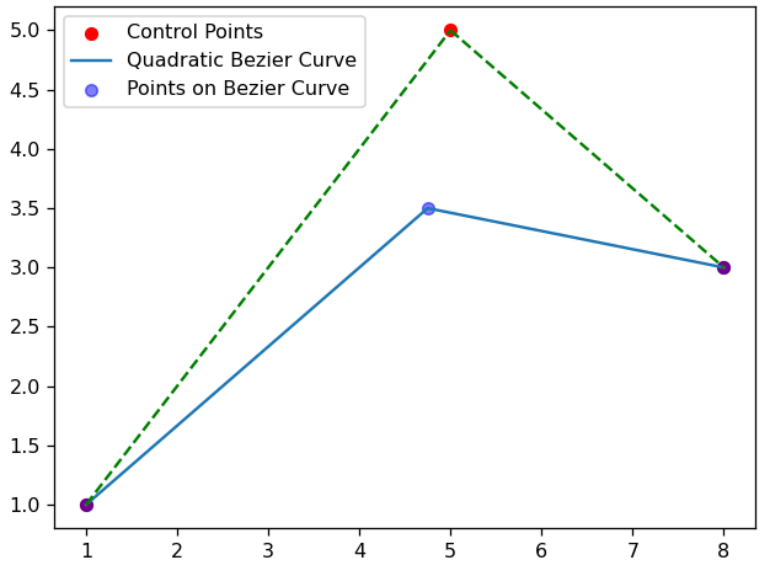
$$P(0.5) = 0.25 \cdot (1,1) + 0.5 \cdot (5,5) + 0.25 \cdot (8,3)$$

$$P(0.5) = (0.25, 0.25) + (2.5, 2.5) + (2, 0.75)$$

$$P(0.5) = (0.25 + 2.5 + 2, 0.25 + 2.5 + 0.75)$$

$$P(0.5) = (4.75, 3.5)$$

Jadi, hasilnya adalah P(0.5)=(4.75,3.5) sesuai dengan yang diharapkan. Ini menunjukkan bahwa dengan satu iterasi, kita dapat mendapatkan nilai tengah pada kurva Bézier dengan menggunakan rumus yang tepat.



Gambar 1 : Hasil Brute - Force

2.1 Algoritma Divide and Conquer

Algoritma yang digunakan untuk menyelesaikan Kurva Bezier kuadratik ini adalah divide and conquer dengan pendekatan mencari titik tengah, algoritma ini memiliki pendekatan dengan membagi masalah menjadi submasalah yang lebih kecil, menyelesaikan masing-masing submasalah secara terpisah, dan kemudian menggabungkan solusi-solusi tersebut untuk mendapatkan solusi akhir.

Dalam algoritma ini, kurva Bézier kuadratik dibagi menjadi dua bagian dengan menggunakan titik tengah. Setiap bagian kemudian dibagi lagi menjadi dua bagian yang lebih kecil. Proses ini terus diulang sampai kita mencapai titik di mana kurva sudah cukup halus atau mencapai kedalaman rekursi yang ditentukan sebelumnya atau bisa juga dikatakan saya menggunakan iterasi sebagai basis dari rekursi yang digunakan.

Ketika kita telah menghitung semua titik pada kedua bagian, kita akan menggabungkan semua titik tersebut untuk membentuk kurva Bézier kuadratik yang lengkap. Proses penggabungan ini menghasilkan kurva Bézier yang halus dan terdefinisi dengan baik. Dengan cara ini, algoritma Divide and Conquer membantu kita memecahkan masalah kompleks menjadi bagian-bagian yang lebih sederhana, dan kemudian menggabungkan solusi-solusi tersebut untuk mendapatkan hasil akhir.

Implementasi Algoritma Divide and Conquer :

```
import matplotlib.pyplot as plt
import numpy as np
from Point import Point
from showCurve import showCurve
import time

# DIVIDE AND CONQUER
def midpoint(p1: Point, p2: Point):
    return Point((p1.x + p2.x) / 2, (p1.y + p2.y) / 2)

def bezierCurve(P1,P2,P3,currIteration):
    curvePoint = []
    if(currIteration < iterations):

        midPoint1 = midpoint(P1,P2)
        midPoint2 = midpoint(P2,P3)
        midPoint3 = midpoint(midPoint1,midPoint2)

        currIteration += 1

curvePoint.extend(bezierCurve(P1,midPoint1,midPoint3,currIteration))
        curvePoint.append(midPoint3)

curvePoint.extend(bezierCurve(midPoint3,midPoint2,P3,currIteration))
    else:
        curvePoint.append(P1)
        curvePoint.append(P3)

    return curvePoint



def divideandConquer():
    P = []
```

```
    # Masukkan
    for i in range(3):
        while True:
            try:
                x, y = map(float, input("Masukkan koordinat x dan y titik
ke-{} : ".format(i)).split())
                P.append(Point(x, y))
                break
            except ValueError:
                print("Invalid  input.  Please  enter  valid  numerical
values.")

    global iterations
    iterations = int(input("Masukkan jumlah iterasi: "))

    curve_points = []



    total_time = 0
    for i in range(0, iterations+1):
        start_time = time.perf_counter()
        curve_points = bezierCurve(P[0], P[1], P[2], iterations - i)
        end_time = time.perf_counter()
        iteration_time = end_time - start_time
        total_time += iteration_time
        showCurve(P, curve_points)

    plt.title(f'Iteration {i}')
```

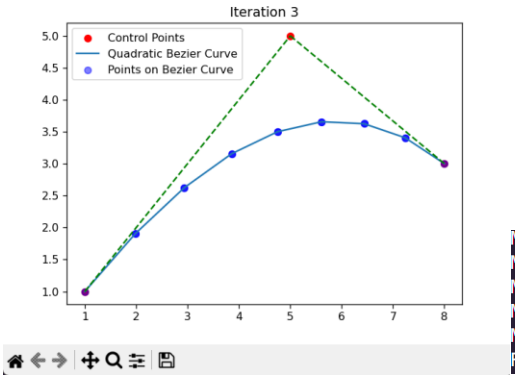
```
    average_time = total_time / (iterations + 1
Masukkan pilihan: 2
Masukkan koordinat x dan y titik ke-0 : 1 1
Masukkan koordinat x dan y titik ke-1 : 5 5
Masukkan koordinat x dan y titik ke-2 : 8 3
Masukkan jumlah iterasi: 3
Waktu eksekusi: 0.05030000011174707 ms
)
    print("Rata-rata waktu eksekusi:", average_time * 1000, "ms")


    # for point in curve_points:
    #     print(point.x, point.y)
```

2.3 Uji Coba

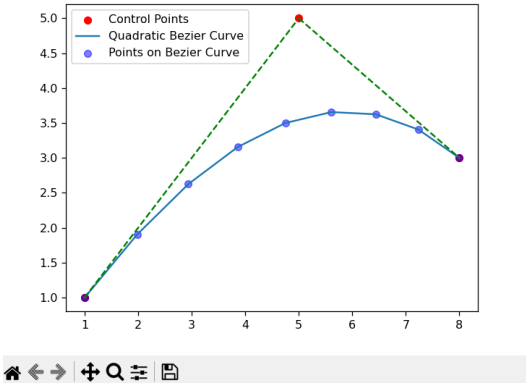
2.3.1 3 titik kontrol dengan titik : (1,1) , (5,5) , (8,3) , 3 Buah iterasi

Divide and Conquer :




```
Masukkan pilihan: 1
Masukkan koordinat x dan y titik ke-0 : 1 1
Masukkan koordinat x dan y titik ke-1 : 5 5
Masukkan koordinat x dan y titik ke-2 : 8 3
Masukkan jumlah iterasi: 3
Rata-rata waktu eksekusi: 0.026900000079876918 ms
```

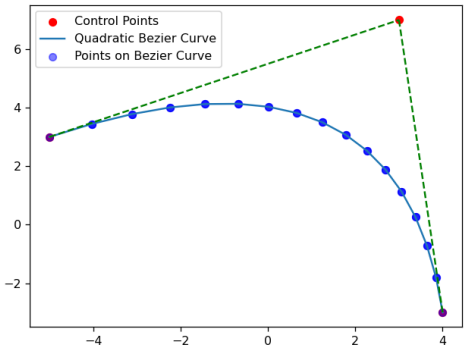
Brute – Force :




```
Masukkan pilihan: 2
Masukkan koordinat x dan y titik ke-0 : 1 1
Masukkan koordinat x dan y titik ke-1 : 5 5
Masukkan koordinat x dan y titik ke-2 : 8 3
Masukkan jumlah iterasi: 3
Waktu eksekusi: 0.05030000011174707 ms
```

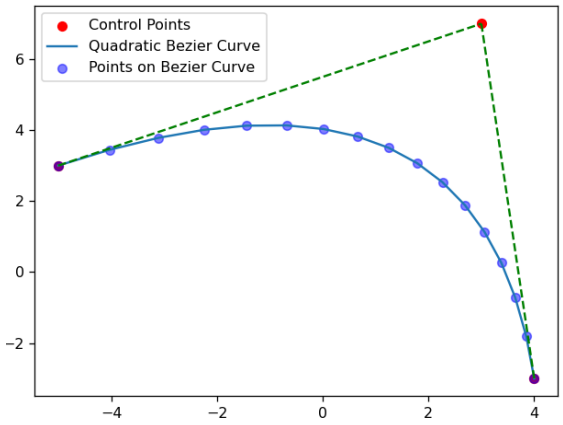
2.3.2 3 Titik kontrol dengan Titik : (-5,3) , (3,7) , (4,-3) dengan 4 iterasi

Divide and Conquer :




```
Masukkan pilihan: 1
Masukkan koordinat x dan y titik ke-0 : -5 3
Masukkan koordinat x dan y titik ke-1 : 3 7
Masukkan koordinat x dan y titik ke-2 : 4 -3
Masukkan jumlah iterasi: 4
Rata-rata waktu eksekusi: 0.037079999992783996 ms
```

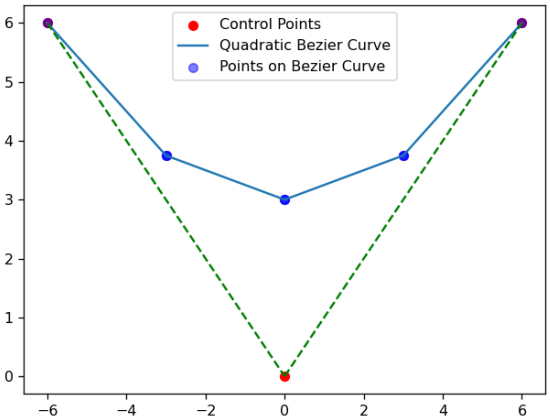
Brute – Force :




```
Masukkan pilihan: 2
Masukkan koordinat x dan y titik ke-0 : -5 3
Masukkan koordinat x dan y titik ke-1 : 3 7
Masukkan koordinat x dan y titik ke-2 : 4 -3
Masukkan jumlah iterasi: 4
Waktu eksekusi: 0.06119999989095959 ms
```

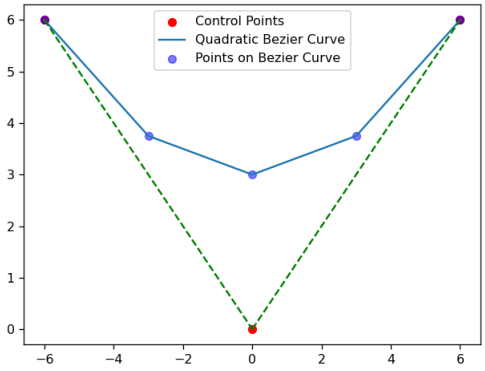
2.3.3 3 Titik kontrol dengan titik : (-6,6) , (0,0) , (6,6) dengan 2 iterasi

Divide and Conquer :




```
Masukkan pilihan: 1
Masukkan koordinat x dan y titik ke-0 : -6 6
Masukkan koordinat x dan y titik ke-1 : 0 0
Masukkan koordinat x dan y titik ke-2 : 6 6
Masukkan jumlah iterasi: 2
Rata-rata waktu eksekusi: 0.013799999957579226 ms
```

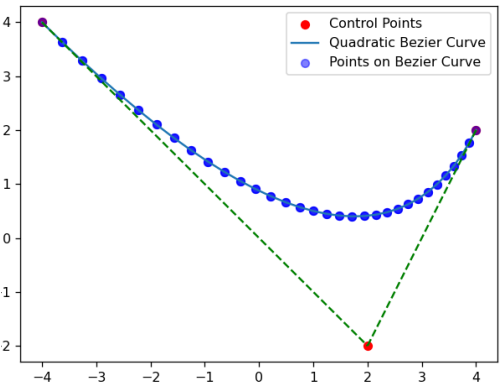
Brute – Force :




```
Masukkan koordinat x dan y titik ke-0 : -6 6
Masukkan koordinat x dan y titik ke-1 : 0 0
Masukkan koordinat x dan y titik ke-2 : 6 6
Masukkan jumlah iterasi: 2
Waktu eksekusi: 0.043000000005122274 ms
```

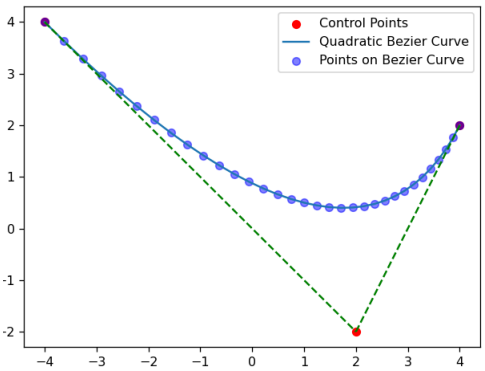
2.3.4 3 titik kontrol dengan titik : (-4,4) , (2,-2), (4,2) dengan 5 iterasi

Divide and Conquer :




```
Masukkan pilihan: 1
Masukkan koordinat x dan y titik ke-0 : -4 4
Masukkan koordinat x dan y titik ke-1 : 2 -2
Masukkan koordinat x dan y titik ke-2 : 4 2
Masukkan jumlah iterasi: 5
Rata-rata waktu eksekusi: 0.036049999835086055 ms
```

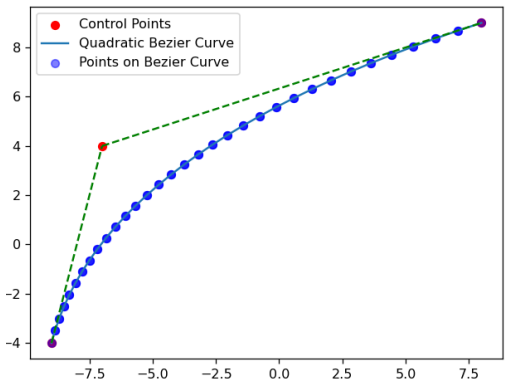
Brute – Force :




```
Masukkan pilihan: 2
Masukkan koordinat x dan y titik ke-0 : -4 4
Masukkan koordinat x dan y titik ke-1 : 2 -2
Masukkan koordinat x dan y titik ke-2 : 4 2
Masukkan jumlah iterasi: 5
Waktu eksekusi: 0.09210000007442432 ms
```

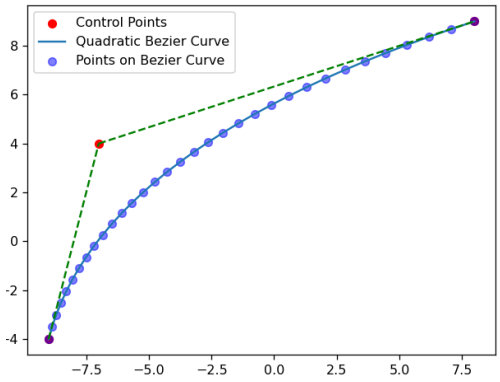
2.3.5 3 Titik kontrol dengan titik (-9,-4) , (-7,4) , (8,9) dengan 5 iterasi

Divide and Conquer :




```
Masukkan pilihan: 1
Masukkan koordinat x dan y titik ke-0 : -9 -4
Masukkan koordinat x dan y titik ke-1 : -7 4
Masukkan koordinat x dan y titik ke-2 : 8 9
Masukkan jumlah iterasi: 5
Rata-rata waktu eksekusi: 0.032516666578885633 ms
```

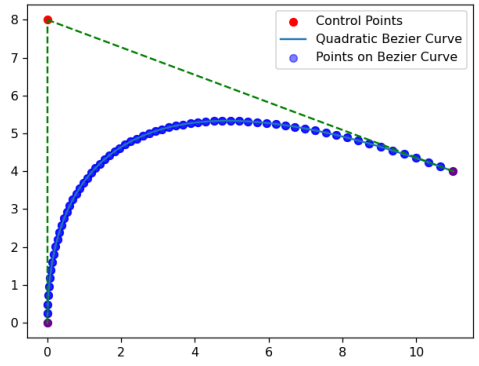
Brute – Force:




```
Masukkan pilihan: 2
Masukkan koordinat x dan y titik ke-0 : -9 -4
Masukkan koordinat x dan y titik ke-1 : -7 4
Masukkan koordinat x dan y titik ke-2 : 8 9
Masukkan jumlah iterasi: 5
Waktu eksekusi: 0.08709999974598759 ms
```

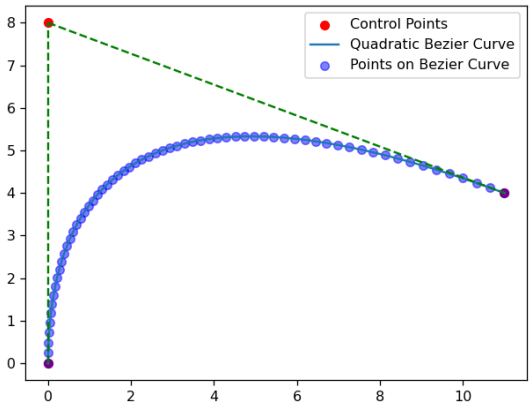
2.3.6 3 Titik kontrol dengan titik : (0,0) , (0,8) , (11,4) dengan 6 iterasi

Divide and Conquer :




```
Masukkan pilihan: 1
Masukkan koordinat x dan y titik ke-0 : 0 0
Masukkan koordinat x dan y titik ke-1 : 0 8
Masukkan koordinat x dan y titik ke-2 : 11 4
Masukkan jumlah iterasi: 6
Rata-rata waktu eksekusi: 0.05220000002736924 ms
```

Brute – Force :




```
Masukkan pilihan: 2
Masukkan koordinat x dan y titik ke-0 : 0 0
Masukkan koordinat x dan y titik ke-1 : 0 8
Masukkan koordinat x dan y titik ke-2 : 11 4
Masukkan jumlah iterasi: 6
Waktu eksekusi: 0.12800000013157842 ms
```

BAB III

PEMBAHASAN

3.1 Perbandingan Kompleksitas Algoritma

Pada kedua algoritma yang digunakan kompleksitas waktu yang dimiliki kedua algoritma tersebut adalah O(2 ^ N) dimana N adalah jumlah iterasi, Pada algoritma brute force, kita menghasilkan semua kemungkinan titik pada kurva Bezier dengan menggunakan pendekatan yang sederhana namun langsung. Setiap iterasi dalam algoritma ini memerlukan evaluasi untuk titik pada kurva Bezier dengan mempertimbangkan semua kombinasi nilai t yang mungkin. Sedangkan pada algoritma divide and conquer, kita membagi masalah menjadi dua submasalah yang lebih kecil, menghitung titik-titik pada kurva Bezier untuk setiap submasalah, dan kemudian menggabungkan hasilnya. Dengan menggunakan pendekatan rekursif ini, algoritma membagi masalah menjadi submasalah yang lebih kecil sampai mencapai kasus dasar. Kompleksitas waktu algoritma divide and conquer juga adalah O(2^N), di mana N adalah jumlah iterasi. Ini karena meskipun kita mengurangi ukuran masalah menjadi dua setiap rekursi, kita masih memiliki total N iterasi yang perlu dieksekusi.

3.2 Time Execution

Dari pengujian yang telah dilakukan time execution dari algoritma divide and conquer mempunyai waktu yang lebih cepat meskipun memiliki kompleksitas algoritma yang sama, ini mungkin terjadi dikarenakan beberapa faktor yaitu :

1. Strategi yang lebih efisien , dikarenakan divide and conquer membagi permasalahan menjadi 2 sub permasalahan yang lebih kecil lalu digabungkan akan lebih mempermudah pemecahan dan mungkin mempercepat permasalahan tersebut
2. Karakteristik Masalah Spesifik: Algoritma Divide and Conquer mungkin lebih cocok untuk menangani jenis masalah ini daripada algoritma brute force. Jika struktur masalah memungkinkan untuk eksploitasi pemisahan dan penggabungan masalah yang efisien, maka algoritma Divide and Conquer dapat memberikan kinerja yang lebih baik.

3.3 Kurva yang dihasilkan

Dari pengujian yang dilakukan bisa dilihat bahwa kurva yang dihasilkan oleh metode Brute – Force maupun Divide and Conquer menghasilkan kurva yang sama. Hal ini menunjukkan bahwa kedua metode tersebut mengimplementasikan algoritma yang benar dan memberikan hasil yang konsisten.

Meskipun keduanya mungkin memiliki pendekatan yang berbeda dalam implementasi, baik Brute Force maupun Divide and Conquer harus menghasilkan kurva yang identik jika diberikan titik kontrol yang sama.

DAFTAR PUSTAKA

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/stima23-24.htm

LAMPIRAN

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.	✓	

Link Repository :

https://github.com/mzaki9/Tucil2_13522136