Laporan Tugas Kecil 2 IF2211 Strategi Algoritma

Semester II tahun 2023/2024

Membangun Kurva bézier dengan Algoritma Titik Tengah berbasis *Divide and Conquer*

Disusun Oleh:

Maulana Muhamad Susetyo - 13522127

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Daftar Isi

1. Analisis dan Implementasi Algoritma Kurva bézier

1.1. Brute Force

Implementasi Brute Force Algoritma Kurva bézier dilakukan dengan mencari titik-titik kurva menggunakan rumus kurva bézier kuadratik yang merupakan substitusi lanjutan dari rumus interpolasi linier.

$$B(t) = (1-t)^2 P0 + 2(1-t)tP1 + t^2 P2,\ t \in (0,1)$$

Titik-titk dihasilkan dengan mencari *increment* berdasarkan jumlah iterasi yang diinginkan. Karena per iterasi jumlah titik yang didapatkan dua kali lipat iterasi sebelumnya, maka *increment* dapat dicari dengan rumus berikut

$$increment = \left(\frac{1}{2}\right)^t$$

Setelah *increment* ditemukan, selanjutnya melakukan iterasi sebanyak 2^t kali. Pada setiap iterasi, nilai t ditambah sebanyak nilai *increment* sampai nilai t = 1. Hasil iterasi kemudian ditambahkan ke list titik.

1.2. Divide and Conquer

Implementasi Algoritma Divide and Conquer untuk Kurva bézier dilakukan dengan melakukan interpolasi linier terhadap titik-titik kontrol untuk menghasilkan dua titik tengah. Kemudian dilakukan interpolasi linier terhadap kedua titik tersebut untuk menghasilkan titik tengah.

Setelah mendapatkan ketiga titik tambahan, akan dilakukan rekursi. Rekursi dibagi menjadi dua, yaitu pada sebelah ‘kiri’ titik tengah, dan sebelah ‘kanan’ titik tengah (kiri dan kanan merujuk ke titik kontrol pertama dan terakhir). Langkah yang diambil pada tiap rekursi sama, melakukan interpolasi linier terhadap titik kontrol baru sehingga menghasilkan titik pada kurva.

Rekursi dijalankan sampai jumlah iterasi sama dengan 1, dimana algoritma akan mengembalikan nilai tengah. Hasil rekursi akan ditambahkan ke list rekursi dengan titik kontrol pertama dan terakhir untuk membuat graf.

2. Source Code dan Tangkapan Layar Program

2.1. Tangkapan Layar Source Code

```
def input_menu():
    list_point = []
    for i in range (0,3):
        while (True):
            point_str = input("Masukkan titik (Format x,y): ")
            try:
                x, y = map(float, point_str.split(','))
                list_point.append([x, y])
                break
            except ValueError:
                print("Input tidak valid.")
    while True:
        print("Pilihan:")
        print("1. Divide and Conquer")
        print("2. Brute Force")
        choice = int(input("Masukkan Pilihan: "))
        print(choice)
        if choice != 1 and choice != 2:
            print("Input Tidak Valid.")
        else:
            break

    while True:
        iterasi = int(input("Jumlah Iterasi: "))
        if choice <1:
            print("Jumlah Iterasi Minimal 1.")
        else:
            break
```

Gambar 2.1.1 Fungsi Input Menggunakan Command Line

```
dnc_start_time = process_time()

res_list = [list_point[0]]
res_list += (bezier_dnc(list_point[0],list_point[1],list_point[2],0.5, iterasi))
res_list += [list_point[2]]
res_list = np.array(res_list)

dnc_end_time = process_time()
time_elapsed = (dnc_end_time - dnc_start_time)*1000
print("Waktu Algoritma:",time_elapsed, "ms")

plot_curve(list_point, res_list)
```

Gambar 2.1.2 Menghitung Waktu dan Menampilkan Hasil Algoritma Divide and Conquer

```
bruteforce_start_time = process_time()

res_list = []
bezier_bruteforce(list_point,res_list,iterasi)
res_list = np.array(res_list)

bruteforce_end_time = process_time()
time_elapsed = (bruteforce_end_time - bruteforce_start_time)*1000
print("Waktu Algoritma:",time_elapsed, "ms")

plot_curve(list_point, res_list)
```

Gambar 2.1.3 Menghitung Waktu dan Menampilkan Hasil Algoritma Brute Force

```
def bezier_dnc(P0, P1, P2, t, iter_cnt):
    Left = [((1-t)*P0[0] + t*P1[0]), ((1-t)*P0[1] + t*P1[1])]
    Right = [((1-t)*P1[0] + t*P2[0]), ((1-t)*P1[1] + t*P2[1])]
    Middle = [((1-t)*Left[0] + t*Right[0]), ((1-t)*Left[1] + t*Right[1])]
    if iter_cnt == 1:
        return [Middle]
    else:
        left_bez = bezier_dnc(P0,Left,Middle,t,iter_cnt-1)
        right_bez = bezier_dnc(Middle,Right,P2,t,iter_cnt-1)
        return left_bez + [Middle] + right_bez
```

Gambar 2.1.4 Algoritma Divide and Conquer Untuk Membangun Kurva bézier

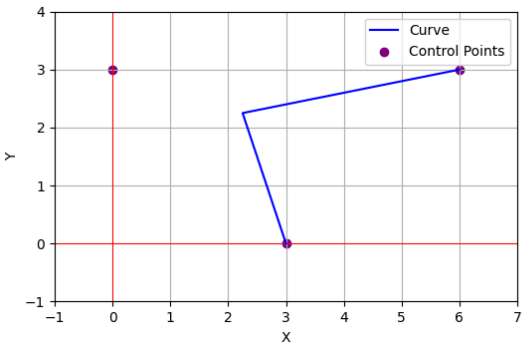
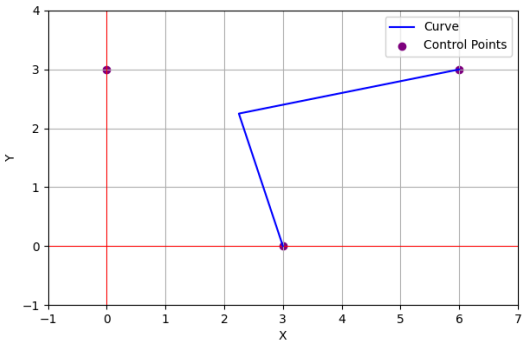
```
def bezier_bruteforce(points_list, result_list, iter_cnt: int):
    result_list.append(points_list[0])
    divv = 1/(2**iter_cnt)
    t = 0
    for i in range(1,2**(iter_cnt)):
        t+=divv
        x=((1-t)**2)*points_list[0][0] + 2*(1-t)*t*points_list[1][0] + (t**2)*points_list[2][0]
        y=((1-t)**2)*points_list[0][1] + 2*(1-t)*t*points_list[1][1] + (t**2)*points_list[2][1]
        result_list.append([x,y])
    result_list.append(points_list[2])
```

Gambar 2.1.5 Algoritma Brute Force Untuk Membangun Kurva bézier

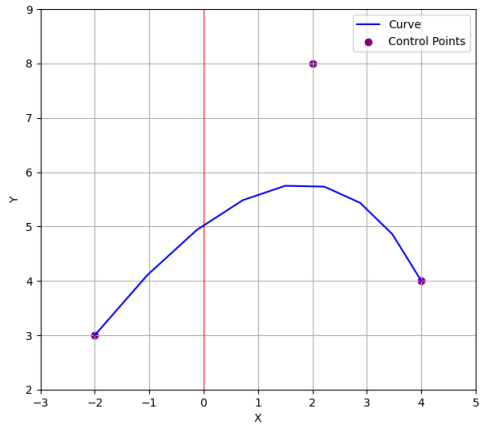
```python
def plot_curve(control ,points):
    plt.figure(figsize=(7,7))
    plt.axhline(0, color='red', linewidth=0.75)
    plt.axvline(0, color='red', linewidth=0.75)
    plt.plot(points[:,0], points[:,1], label="Curve", color="blue")
    plt.scatter([control[0][0], control[1][0], control[2][0]], [control[0][1], control[1][1], control[2][1]], color="purple",label="Control Points")
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.axis('on')
    plt.legend()
    plt.gca().set_aspect('equal', adjustable='box')
    plt.grid(True)
    plt.xlim(min(control[0][0], control[1][0], control[2][0]) - 1, max(control[0][0], control[1][0], control[2][0]) + 1)
    plt.ylim(min(control[0][1], control[1][1], control[2][1]) - 1, max(control[0][1], control[1][1], control[2][1]) + 1)
    plt.show()
```

Gambar 2.1.6 Fungsi Menampilkan Plot Kurva bézier

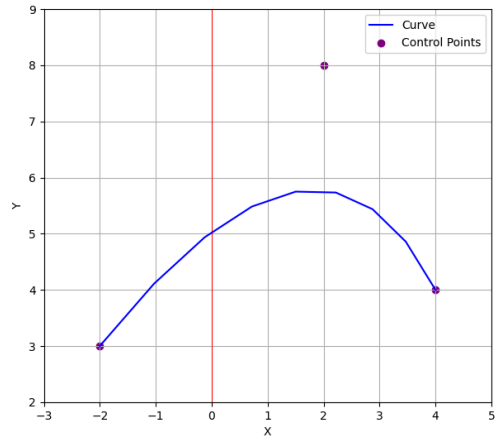
2.2. Tangkapan Layar Input dan Output

Input & Waktu	Output
Masukkan titik (Format x,y): 3,0 Masukkan titik (Format x,y): 0,3 Masukkan titik (Format x,y): 6,3 Pilihan: 1. Divide and Conquer 2. Brute Force Masukkan Pilihan: 1 1 Jumlah Iterasi: 1 Waktu Algoritma DnC: 0.0 ms	
Masukkan titik (Format x,y): 3,0 Masukkan titik (Format x,y): 0,3 Masukkan titik (Format x,y): 6,3 Pilihan: 1. Divide and Conquer 2. Brute Force Masukkan Pilihan: 2 2 Jumlah Iterasi: 1 Waktu Algoritma BF: 0.0 ms	

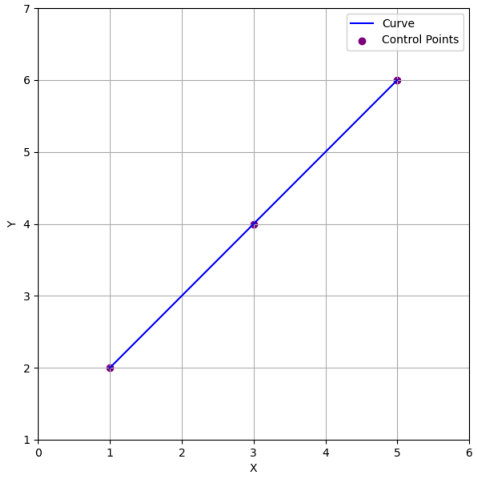
```
Masukkan titik (Format x,y): -2,3
Masukkan titik (Format x,y): 2,8
Masukkan titik (Format x,y): 4,4
Pilihan:
1. Divide and Conquer
2. Brute Force
Masukkan Pilihan: 1
1
Jumlah Iterasi: 3
Waktu Algoritma DnC: 0.0 ms
```



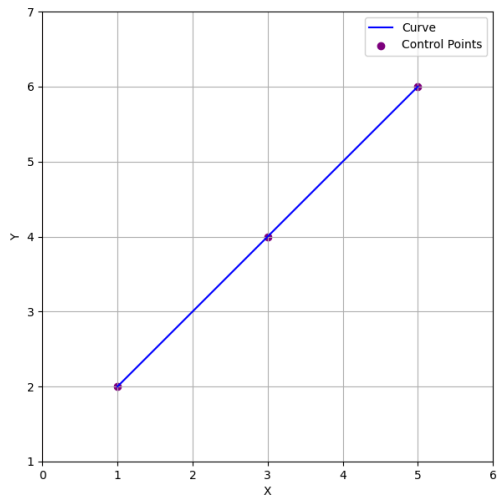

```
Masukkan titik (Format x,y): -2,3
Masukkan titik (Format x,y): 2,8
Masukkan titik (Format x,y): 4,4
Pilihan:
1. Divide and Conquer
2. Brute Force
Masukkan Pilihan: 2
2
Jumlah Iterasi: 3
Waktu Algoritma BF: 0.0 ms
```



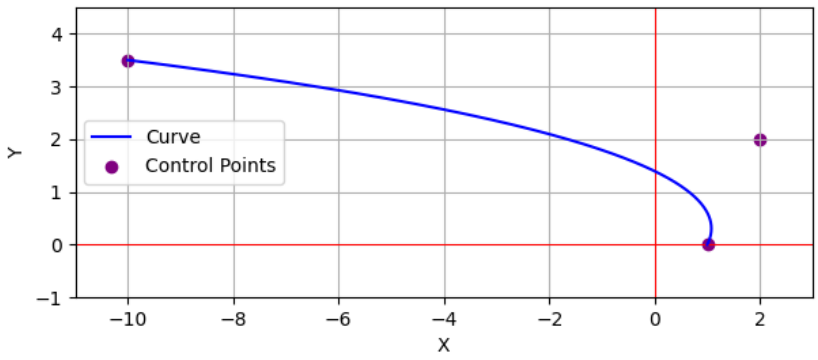

```
Masukkan titik (Format x,y): 1,2
Masukkan titik (Format x,y): 3,4
Masukkan titik (Format x,y): 5,6
Pilihan:
1. Divide and Conquer
2. Brute Force
Masukkan Pilihan: 1
1
Jumlah Iterasi: 6
Waktu Algoritma DnC: 0.0 ms
```



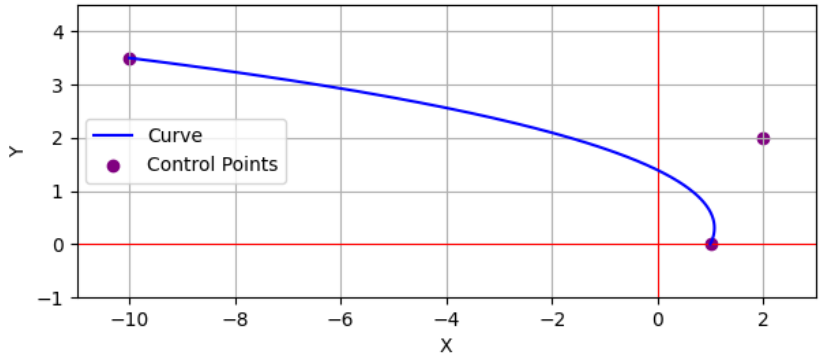
```
Masukkan titik (Format x,y): 1,2
Masukkan titik (Format x,y): 3,4
Masukkan titik (Format x,y): 5,6
Pilihan:
1. Divide and Conquer
2. Brute Force
Masukkan Pilihan: 2
2
Jumlah Iterasi: 6
Waktu Algoritma BF: 0.0 ms
```



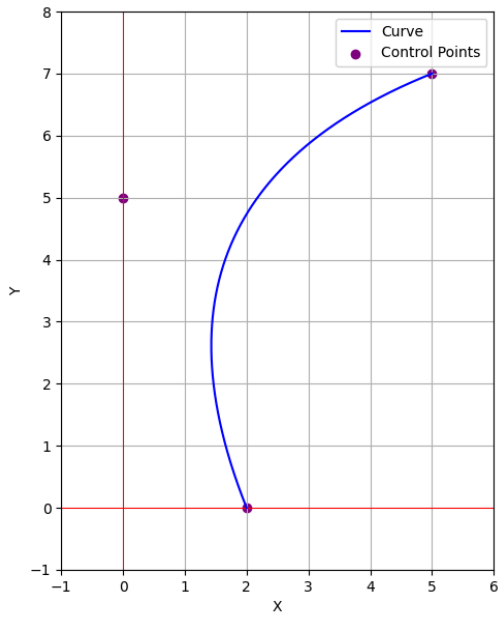

```
Masukkan titik (Format x,y): 1,0
Masukkan titik (Format x,y): 2,2
Masukkan titik (Format x,y): -10,3.5
Pilihan:
1. Divide and Conquer
2. Brute Force
Masukkan Pilihan: 1
1
Jumlah Iterasi: 16
Waktu Algoritma DnC: 31.25 ms
```



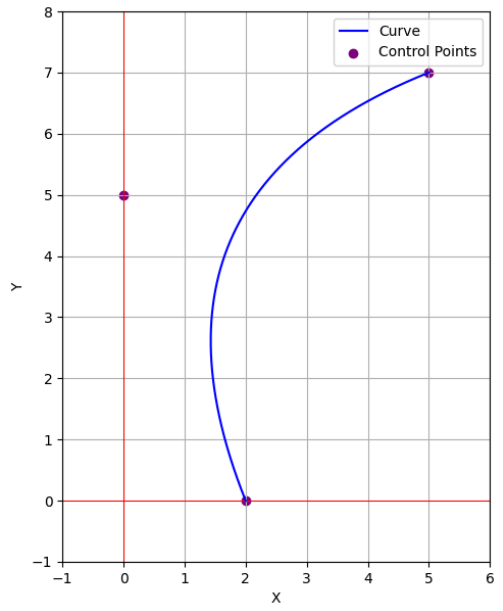

```
Masukkan titik (Format x,y): 1,0
Masukkan titik (Format x,y): 2,2
Masukkan titik (Format x,y): -10,3.5
Pilihan:
1. Divide and Conquer
2. Brute Force
Masukkan Pilihan: 2
2
Jumlah Iterasi: 16
Waktu Algoritma BF: 15.625 ms
```

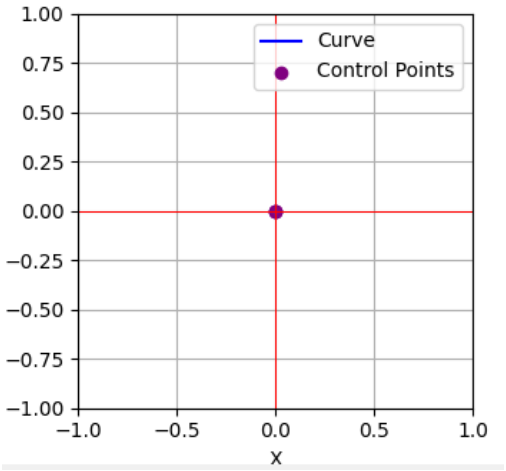
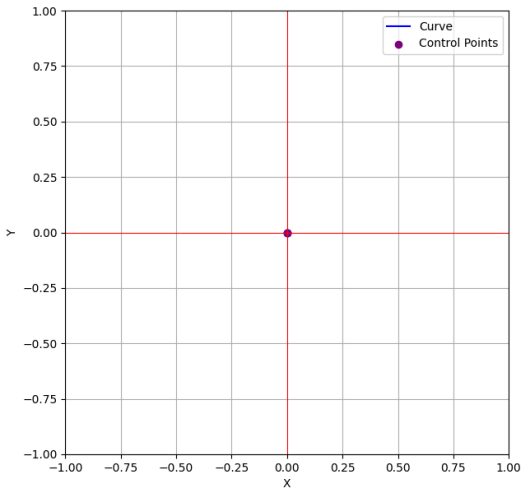


```
Masukkan titik (Format x,y): 5,7
Masukkan titik (Format x,y): 0,5
Masukkan titik (Format x,y): 2,0
Pilihan:
1. Divide and Conquer
2. Brute Force
Masukkan Pilihan: 1
1
Jumlah Iterasi: 18
Waktu Algoritma DnC: 140.625 ms
```




```
Masukkan titik (Format x,y): 5,7
Masukkan titik (Format x,y): 0,5
Masukkan titik (Format x,y): 2,0
Pilihan:
1. Divide and Conquer
2. Brute Force
Masukkan Pilihan: 2
2
Jumlah Iterasi: 18
Waktu Algoritma BF: 78.125 ms
```



Masukkan titik (Format x,y): 0,0 Masukkan titik (Format x,y): 0,0 Masukkan titik (Format x,y): 0,0 Pilihan: 1. Divide and Conquer 2. Brute Force Masukkan Pilihan: 1 1 Jumlah Iterasi: 5 Waktu Algoritma DnC: 0.0 ms	
Masukkan titik (Format x,y): 0,0 Masukkan titik (Format x,y): 0,0 Masukkan titik (Format x,y): 0,0 Pilihan: 1. Divide and Conquer 2. Brute Force Masukkan Pilihan: 2 2 Jumlah Iterasi: 5 Waktu Algoritma BF: 0.0 ms	

3. Hasil Analisis Perbandingan Solusi *Brute Force* Dengan *Divide and Conquer*

Algoritma Brute Force pembangunan Kurva bézier pertama melakukan perhitungan *increment*, kemudian mencari titik sebanyak 2^{iterasi}. Jadi kompleksitas algoritma Brute Force pembangunan Kurva bézier adalah sebagai berikut (jumlah iterasi sebagai n)

$$T(n) = 1 + 19 * 2^n, O(n) = 2^n$$

Untuk algoritma Divide and Conquer pembangunan Kurva bézier, pertama melakukan interpolasi linier untuk mencari titik tengah dua kontrol poin pertama dan titik tengah dua kontrol poin terakhir, kemudian melakukan interpolasi linier terhadap kedua titik tersebut untuk menghasilkan titik pada Kurva bézier. Rekursi kemudian dilakukan untuk mencari titik kiri dan titik kanan dari titik tengah pertama. Ini dilakukan hingga jumlah iterasi 1, dimana hasil akan dikembalikan sebagai list. Karena per iterasi jumlah titik yang dicari digandakan, maka kompleksitas algoritma Divide and Conquer adalah sebagai berikut

$$T(n) = 18 * 2^n, O(n) = 2^n$$

Karena bilangan Big O kedua algoritma tersebut sama, maka seharusnya waktu penyelesaian kedua algoritma hampir sama, tetapi pada kasus dimana jumlah iterasi tinggi, contohnya tes 7-8 dan 9-10, waktu penyelesaian algoritma Divide and Conquer dua kali lipat waktu penyelesaian algoritma Brute Force pada titik-titik dan dengan jumlah iterasi yang sama. Padahal kompleksitas algoritma DnC lebih kecil dibandingkan Brute Force. Hal ini dikarenakan program menggunakan bahasa python, sehingga tidak ada optimasi untuk fungsi rekursif.

4. Lampiran dan Pranala

Link Github: https://github.com/LastPrism7/Tucil2_13522127
Checklist Program:

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.		✓