LAPORAN TUGAS KECIL 2
IF2211 STRATEGI ALGORITMA

MEMBANGUN KURVA BÉZIER DENGAN ALGORITMA TITIK TENGAH BERBASIS *DIVIDE AND CONQUER*

Oleh :

Muhammad Fiqri

10023519

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

Daftar Isi

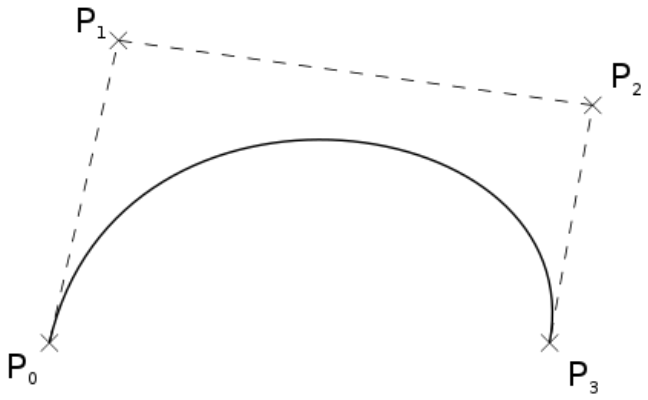
BAB I Deskripsi Program

Kurva Bézier adalah elemen penting dalam grafis komputer, sering digunakan dalam desain grafis animasi dan manufaktur. Program ini bertujuan untuk membangun kurva Bézier kuadratik menggunakan algoritma titik tengah berbasis divide and conquer. Dua metode akan dibandingkan: brute force dan divide and conquer, untuk mengilustrasikan keefektifan pendekatan berbasis divide and conquer dalam pembentukan kurva yang halus dan presisi. Input program melibatkan tiga pasangan titik kontrol dan jumlah iterasi, menghasilkan visualisasi kurva Bézier dan waktu eksekusi kedua metode.

Kurva Bézier dibuat dengan menghubungkan titik-titik kontrol dengan kurva yang menggambarkan bentuk dan arah. Metode divide and conquer memecah masalah pembuatan kurva menjadi sub-proses yang lebih kecil, memungkinkan pembuatan kurva yang lebih efisien dan akurat. Program ini mengimplementasikan kedua metode untuk menunjukkan proses pembentukan kurva Bézier dan membandingkan performa mereka.

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.

(Paragraf di atas dikutip dari : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Tucil2-2024.pdf).



Gambar 1. Kurva Bézier Kubik
(Sumber: https://id.wikipedia.org/wiki/Kurva_B%C3%A9zier)

BAB II Analisis dan Implementasi Algoritma Brute Force

Dalam analisis dan implementasi algoritma brute force pada pembentukan kurva Bézier, pendekatan ini memproses setiap kemungkinan kombinasi titik kontrol untuk menghasilkan kurva. Berikut adalah langkah-langkah implementasi algoritma brute force yang diterapkan dalam program :

1. Program menerima input berupa tiga pasang titik kontrol yang digunakan untuk membentuk kurva Bézier. Setiap pasangan titik kontrol direpresentasikan oleh koordinat (x, y) yang dipisahkan oleh spasi. Pastikan bahwa input sesuai dengan format yang diharapkan.
2. Program mengevaluasi semua kemungkinan kombinasi titik kontrol untuk membentuk kurva Bézier. Ini dilakukan dengan cara memilih setiap kombinasi tiga titik kontrol dari total titik kontrol yang diberikan. Setiap kombinasi dievaluasi untuk membentuk segmen kurva Bézier.
3. Setiap segmen kurva Bézier dihitung menggunakan rumus parametrik Bézier. Dalam pendekatan brute force, semua titik pada kurva dievaluasi secara langsung dengan menggunakan rumus ini. Ini berarti bahwa tidak ada optimisasi yang dilakukan untuk mengurangi jumlah perhitungan yang diperlukan.
4. Setiap hasil evaluasi, yaitu setiap titik pada kurva yang dihasilkan, disimpan dalam struktur data seperti array atau list. Ini memungkinkan program untuk menyimpan semua titik pada kurva tanpa kehilangan informasi.
5. Setelah semua titik pada kurva Bézier dihasilkan, program dapat memvisualisasikan kurva tersebut. Ini dapat dilakukan dengan menggunakan library grafis seperti matplotlib untuk Python, atau dengan menggunakan alat visualisasi lainnya.

Dengan algoritma brute force ini, program secara langsung mengevaluasi semua kemungkinan kombinasi titik kontrol untuk membentuk kurva Bézier, tanpa mempertimbangkan pendekatan optimisasi lainnya. Ini memastikan bahwa semua titik pada kurva dapat dihasilkan, meskipun dengan biaya komputasi yang mungkin tinggi terutama untuk jumlah titik kontrol yang besar.

BAB III Analisis dan Implementasi Algoritma Divide and Conquer

Dalam implementasi algoritma pembentukan kurva Bézier menggunakan metode titik tengah berbasis divide and conquer, kita mengamati beberapa aspek yang relevan. Pertama-tama, kita menggunakan fungsi bezier_curve yang menerima tiga titik kontrol sebagai masukan dan parameter t untuk menentukan posisi pada kurva Bézier. Metode ini secara iteratif menghitung posisi titik pada kurva dengan mengaplikasikan rumus parametrik Bézier pada setiap iterasi. Selain itu, kita memperhitungkan titik tengah dari dua segmen kurva yang dihasilkan dalam algoritma divide and conquer, yang diwakili oleh titik Q_0dan Q_1, serta titik tengah mereka yang disebut B.

Dalam algoritma divide and conquer untuk membentuk kurva Bézier, rumus yang digunakan adalah rumus parametrik untuk titik pada kurva Bézier. Misalkan kita memiliki tiga titik kontrol P_0, P_1,P_2, dan parameter t berkisar antara 0 dan 1, maka rumus untuk menghitung titik pada kurva Bézier adalah sebagai berikut :

$$B(t) = (1-t)^2.P_0 + 2(1-t)t.P_1 + t^2.P_2$$

Dalam rumus ini, B(t) merupakan titik pada kurva Bézier pada parameter t tertentu. P_0, P_1, dan P_2 adalah titik kontrol yang diberikan oleh pengguna. Parameter t berkisar antara 0 dan 1, dan digunakan untuk menentukan posisi pada kurva. Dengan mengubah nilai parameter t dari 0 hingga 1, kita dapat menghasilkan sejumlah titik pada kurva Bézier.

Rumus ini digunakan pada setiap segmen kurva Bézier yang dibangun dalam algoritma divide and conquer. Dengan mengganti nilai parameter t, kita dapat menghitung posisi titik pada kurva pada berbagai titik pada kurva. Ini memungkinkan kita untuk membangun kurva Bézier dengan presisi yang tinggi menggunakan algoritma divide and conquer.

Dari segi visualisasi, kita dapat melihat kurva Bézier yang dihasilkan oleh algoritma pada plot. Kurva ini dibangun berdasarkan titik kontrol yang diberikan oleh pengguna, serta garis dan titik spesifik pada kurva pada nilai t=0.25. Visualisasi ini memberikan pemahaman visual yang lebih baik tentang bentuk dan struktur dari kurva Bézier yang dihasilkan oleh algoritma.

Dengan mempertimbangkan analisis kinerja dan visualisasi hasil, kita dapat mengevaluasi keefektifan algoritma divide and conquer dalam menghasilkan kurva Bézier. Dalam kasus ini, kita dapat mengamati apakah algoritma dapat menghasilkan kurva dengan presisi yang diinginkan dalam waktu yang wajar tergantung pada jumlah iterasi yang diberikan oleh pengguna.

BAB IV Source Code

Berikut adalah tampilan source code nya :

A. Algoritma Brute Force

Berikut adalah potongan kode program yang digunakan untuk implementasi algoritma pembentukan kurva Bézier menggunakan metode brute force :

D: > Campus > PMM ITB > Strategi Algoritma > Tucil2_10023519 > src > Bezier Curve Bruteforce.py > {} plt

```python
import matplotlib.pyplot as plt
import time

Codeium: Refactor | Explain | Generate Docstring | ×
def binomial(n, k):
    if k < 0 or k > n:
        return 0
    if k == 0 or k == n:
        return 1
    k = min(k, n - k)  # Optimisasi untuk mengurangi perhitungan
    result = 1
    for i in range(k):
        result *= (n - i)
        result //= (i + 1)
    return result

Codeium: Refactor | Explain | Generate Docstring | ×
def brute_force_quadratic_bezier(points, iterations):
    curve_points = []
    n = len(points)  # Jumlah titik kontrol

    for i in range(iterations + 1):
        t = i / iterations  # Parameter t berjalan dari 0 sampai 1
        x, y = 0, 0
        for j, (px, py) in enumerate(points):
            # Menghitung koefisien binomial
            b = binomial(n - 1, j) * (1 - t)**(n - 1 - j) * t**j
            x += px * b
            y += py * b
        curve_points.append((x, y))

    return curve_points

Codeium: Refactor | Explain | Generate Docstring | ×
def plot_curve(points, curve_points):
    # Memplot titik kontrol
    plt.plot([p[0] for p in points], [p[1] for p in points], 'ro-', label='Control Points')
    # Memplot kurva Bézier
    plt.plot([p[0] for p in curve_points], [p[1] for p in curve_points], 'b-', label='Quadratic Bezier Curve')
    plt.legend()
    plt.title('Bezier Curve')
    plt.show()
```

```python
    return curve_points

Codeium: Refactor | Explain | Generate Docstring | ×
def plot_curve(points, curve_points):
    # Memplot titik kontrol
    plt.plot([p[0] for p in points], [p[1] for p in points], 'ro-', label='Control Points')
    # Memplot kurva Bézier
    plt.plot([p[0] for p in curve_points], [p[1] for p in curve_points], 'b-', label='Quadratic Bezier Curve')
    plt.legend()
    plt.title('Bezier Curve')
    plt.show()

Codeium: Refactor | Explain | Generate Docstring | ×
def main():
    # Meminta pengguna memasukkan koordinat untuk tiga titik kontrol
    points = []
    for i in range(3):
        point = tuple(map(float, input(f"Enter coordinates for point {i+1} (x y): ").split()))
        points.append(point)

    iterations = int(input("Enter the number of iterations: "))

    # Mulai mengukur waktu
    start_time = time.time()

    # Menghitung titik-titik pada kurva Bézier dan memplotnya
    curve_points = brute_force_quadratic_bezier(points, iterations)

    # Selesai mengukur waktu
    end_time = time.time()

    # Memplot kurva
    plot_curve(points, curve_points)

    # Menampilkan waktu eksekusi
    print(f"Execution time: {end_time - start_time:.4f} seconds")

if __name__ == "__main__":
    main()
```

B. Algoritma Divide and Conquer

Berikut adalah potongan kode program yang digunakan untuk implementasi algoritma pembentukan kurva Bézier menggunakan metode divide and conquer :

D: > Campus > PMM ITB > Strategi Algoritma > Tucil2_10023519 > src > Bezier Curve Divide and Conquer.py > {} plt

```python
import matplotlib.pyplot as plt
import numpy as np
import time

# Fungsi Kurva Bezier
Codeium: Refactor | Explain | Generate Docstring | ×
def bezier_curve(points, t):
    P0, P1, P2 = points
    Q0 = ((1-t) * P0[0] + t * P1[0], (1-t) * P0[1] + t * P1[1])
    Q1 = ((1-t) * P1[0] + t * P2[0], (1-t) * P1[1] + t * P2[1])
    B = ((1-t) * Q0[0] + t * Q1[0], (1-t) * Q0[1] + t * Q1[1])
    return Q0, Q1, B

# Masukan titik kontrol input dan jumlah iterasi secara manual
P0 = tuple(map(float, input("Enter coordinates for point P0 (x y): ").split()))
P1 = tuple(map(float, input("Enter coordinates for point P1 (x y): ").split()))
P2 = tuple(map(float, input("Enter coordinates for point P2 (x y): ").split()))
iterations = int(input("Enter the number of iterations: "))

# Menghitung Kurva Poin
t_values = np.linspace(0, 1, iterations)
start_time = time.time()
curve_points = [bezier_curve([P0, P1, P2], t)[2] for t in t_values]
end_time = time.time()

# Mengekstrak titik spesifik pada t = 0,25
Q0, Q1, B = bezier_curve([P0, P1, P2], 0.25)

# Plotting
plt.figure(figsize=(14, 6))

# Control points
plt.plot([p[0] for p in [P0, P1, P2]], [p[1] for p in [P0, P1, P2]], 'ko-', label='Control Points')

# Kurva
plt.plot([p[0] for p in curve_points], [p[1] for p in curve_points], 'r-', label='Bezier Curve')

# Garis pada t = 0,25
plt.plot([Q0[0], Q1[0]], [Q0[1], Q1[1]], 'b-', label='Line Q0-Q1 at t=0.25')
plt.plot(B[0], B[1], 'go', label='Point B at t=0.25')

plt.title('Bezier Curve')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.grid(True)
plt.show()

# Waktu Eksekusi
print(f"Execution time: {end_time - start_time:.4f} seconds")
```

BAB V Eksperimen dan Hasil

A. Input dan Output

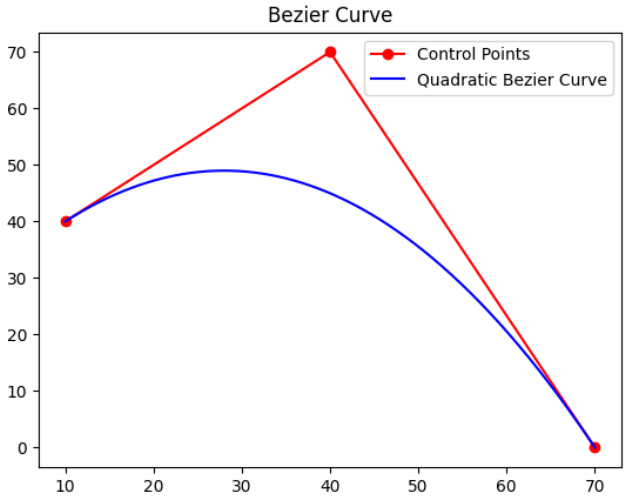
Dalam eksperimen ini, kami melakukan berbagai percobaan dengan menggunakan kedua algoritma, yaitu brute force dan divide and conquer, untuk membentuk kurva Bézier. Kami memberikan berbagai titik kontrol sebagai input untuk kedua algoritma dan memperhatikan hasil kurva yang dihasilkan.

a) Algoritma Brute Force

Input 1 :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS                    Python
PS C:\Users\Acer> & C:/Python312/python.exe "d:/Campus/PMM ITB/Strategi Algoritma/Tucil2_10023519/src/Bezier Curve Bruteforce.py"
Enter coordinates for point 1 (x y): 10 40
Enter coordinates for point 2 (x y): 40 70
Enter coordinates for point 3 (x y): 70 0
Enter the number of iterations: 1000
Execution time: 0.0187 seconds
PS C:\Users\Acer>
```

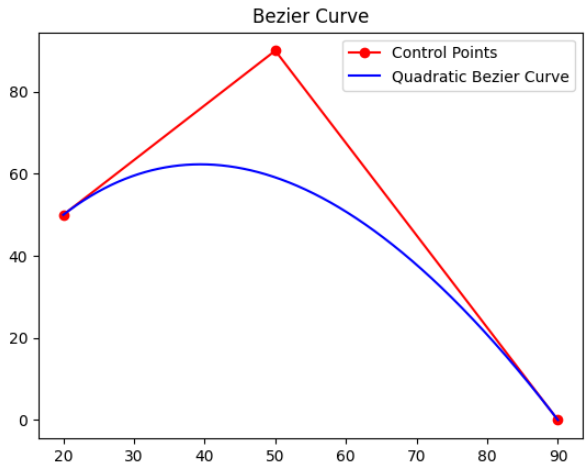
Output 1 :



Input 2 :

```
PS C:\Users\Acer> & C:/Python312/python.exe "d:/Campus/PMM ITB/Strategi Algoritma/Tucil2_10023519/src/Bezier Curve Bruteforce.py"
Enter coordinates for point 1 (x y): 20 50
Enter coordinates for point 2 (x y): 50 90
Enter coordinates for point 3 (x y): 90 0
Enter the number of iterations: 2000
Execution time: 0.0167 seconds
PS C:\Users\Acer>
```

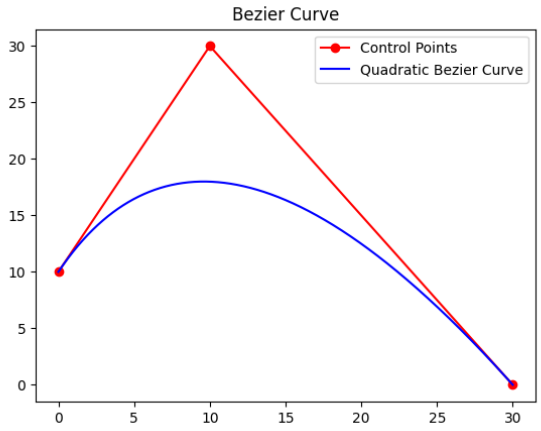
Output 2 :



Input 3 :

```
PS C:\Users\Acer> & C:/Python312/python.exe "d:/Campus/PMM ITB/Strategi Algoritma/Tucil2_10023519/src/Bezier Curve Bruteforce.py"
Enter coordinates for point 1 (x y): 0 10
Enter coordinates for point 2 (x y): 10 30
Enter coordinates for point 3 (x y): 30 0
Enter the number of iterations: 2500
Execution time: 0.0575 seconds
PS C:\Users\Acer>
```

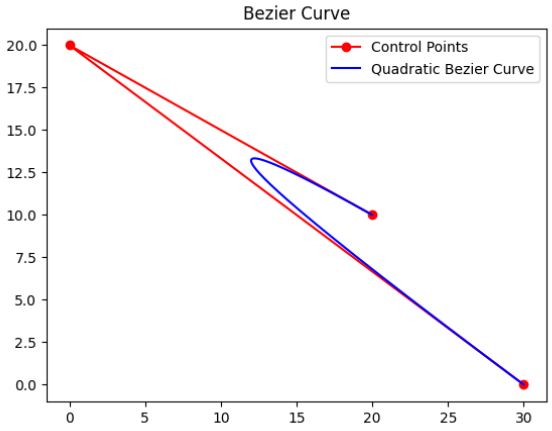
Output 3 :



Input 4 :

```
PS C:\Users\Acer> & C:/Python312/python.exe "d:/Campus/PMM ITB/Strategi Algoritma/Tucil2_10023519/src/Bezier Curve Bruteforce.py"
Enter coordinates for point 1 (x y): 30 0
Enter coordinates for point 2 (x y): 0 20
Enter coordinates for point 3 (x y): 20 10
Enter the number of iterations: 500
Execution time: 0.0249 seconds
PS C:\Users\Acer>
```

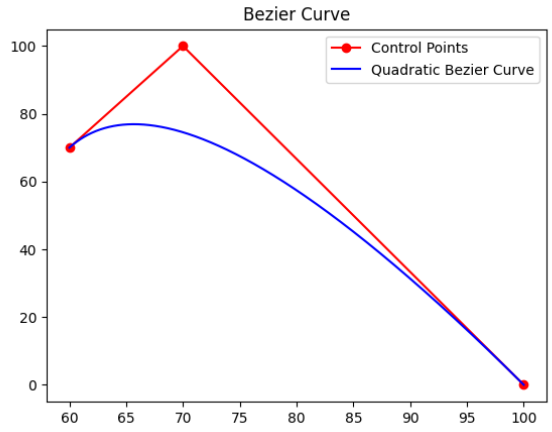
Output 4 :



Input 5 :

```
PS C:\Users\Acer> & C:/Python312/python.exe "d:/Campus/PMM ITB/Strategi Algoritma/Tucil2_10023519/src/Bezier Curve Bruteforce.py"
Enter coordinates for point 1 (x y): 60 70
Enter coordinates for point 2 (x y): 70 100
Enter coordinates for point 3 (x y): 100 0
Enter the number of iterations: 5000
Execution time: 0.0507 seconds
PS C:\Users\Acer>
```

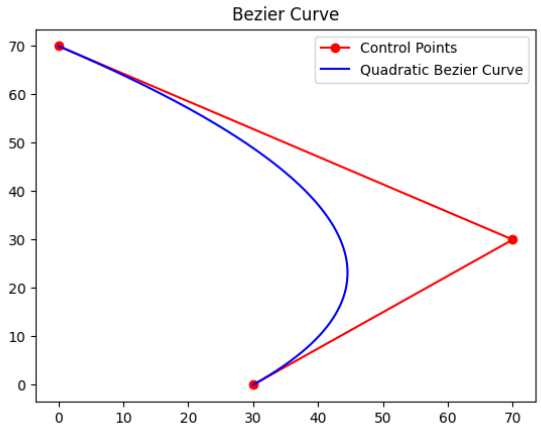
Output 5 :



Input 6 :

```
PS C:\Users\Acer> & C:/Python312/python.exe "d:/Campus/PMM ITB/Strategi Algoritma/Tucil2_10023519/src/Bezier Curve Bruteforce.py"
Enter coordinates for point 1 (x y): 0 70
Enter coordinates for point 2 (x y): 70 30
Enter coordinates for point 3 (x y): 30 0
Enter the number of iterations: 10000
Execution time: 0.0627 seconds
PS C:\Users\Acer>
```

Output 6 :

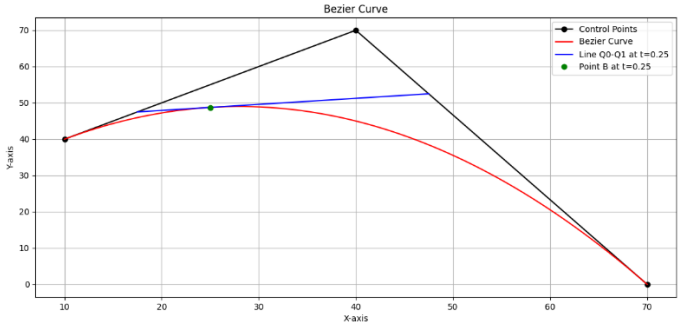


b) Algoritma Divide and Conquer

Input 1 :

```
PS C:\Users\Acer> & C:/Python312/python.exe "d:/Campus/PMM ITB/Strategi Algoritma/Tucil2_10023519/src/Bezier Curve Divide and Conquer.py"
Enter coordinates for point P0 (x y): 10 40
Enter coordinates for point P1 (x y): 40 70
Enter coordinates for point P2 (x y): 70 0
Enter the number of iterations: 1000
Execution time: 0.0431 seconds
PS C:\Users\Acer>
```

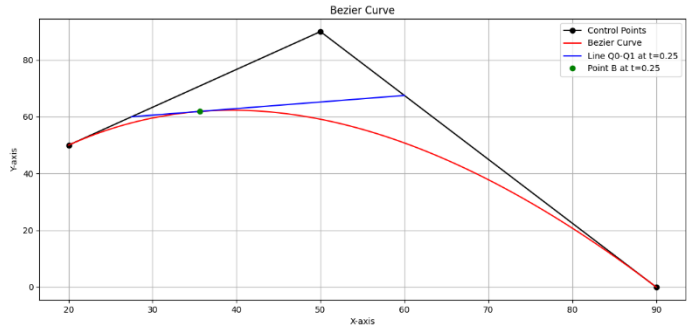
Output 1 :



Input 2 :

```
PS C:\Users\Acer> & C:/Python312/python.exe "d:/Campus/PMM ITB/Strategi Algoritma/Tucil2_10023519/src/Bezier Curve Divide and Conquer.py"
Enter coordinates for point P0 (x y): 20 50
Enter coordinates for point P1 (x y): 50 90
Enter coordinates for point P2 (x y): 90 0
Enter the number of iterations: 2000
Execution time: 0.0593 seconds
PS C:\Users\Acer>
```

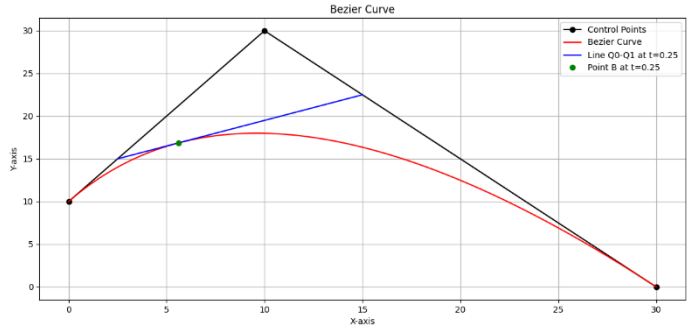
Output 2 :



Input 3 :

```
PS C:\Users\Acer> & C:/Python312/python.exe "d:/Campus/PMM ITB/Strategi Algoritma/Tucil2_10023519/src/Bezier Curve Divide and Conquer.py"
Enter coordinates for point P0 (x y): 0 10
Enter coordinates for point P1 (x y): 10 30
Enter coordinates for point P2 (x y): 30 0
Enter the number of iterations: 2500
Execution time: 0.0471 seconds
PS C:\Users\Acer>
```

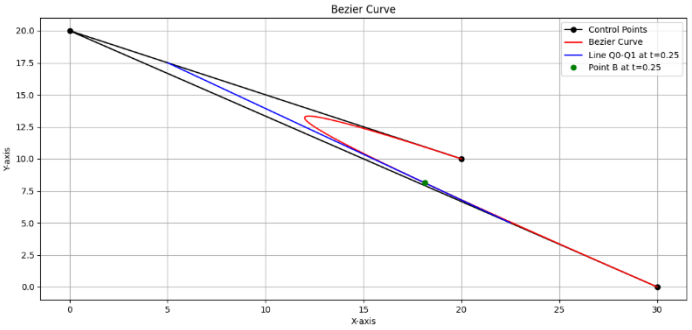
Output 3 :



Input 4 :

```
PS C:\Users\Acer> & C:/Python312/python.exe "d:/Campus/PMM ITB/Strategi Algoritma/Tucil2_10023519/src/Bezier Curve Divide and Conquer.py"
Enter coordinates for point P0 (x y): 30 0
Enter coordinates for point P1 (x y): 0 20
Enter coordinates for point P2 (x y): 20 10
Enter the number of iterations: 500
Execution time: 0.0233 seconds
PS C:\Users\Acer>
```

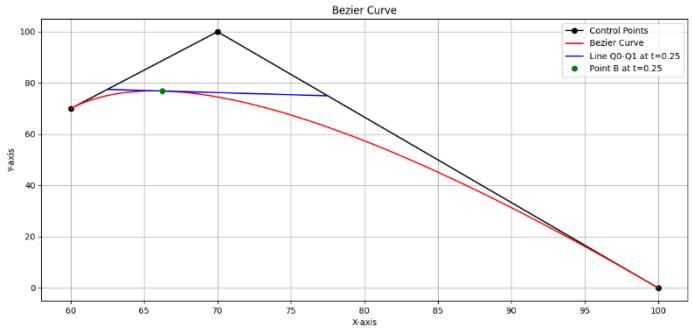
Output 4 :



Input 5 :

```
PS C:\Users\Acer> & C:/Python312/python.exe "d:/Campus/PMM ITB/Strategi Algoritma/Tucil2_10023519/src/Bezier Curve Divide and Conquer.py"
Enter coordinates for point P0 (x y): 60 70
Enter coordinates for point P1 (x y): 70 100
Enter coordinates for point P2 (x y): 100 0
Enter the number of iterations: 5000
Execution time: 0.0371 seconds
PS C:\Users\Acer>
```

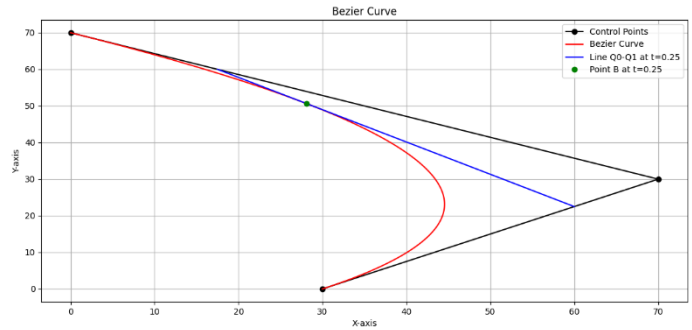
Output 5 :



Input 6 :

```
PS C:\Users\Acer> & C:/Python312/python.exe "d:/Campus/PMM ITB/Strategi Algoritma/Tucil2_10023519/src/Bezier Curve Divide and Conquer.py"
Enter coordinates for point P0 (x y): 0 70
Enter coordinates for point P1 (x y): 70 30
Enter coordinates for point P2 (x y): 30 0
Enter the number of iterations: 10000
Execution time: 0.0684 seconds
PS C:\Users\Acer>
```

Output 6 :



B. Analisis Perbandingan

Hasil analisis perbandingan antara solusi brute force dengan divide and conquer menunjukkan perbedaan yang signifikan dalam kinerja dan kompleksitas kedua algoritma.

Dalam konteks algoritma, brute force dan divide and conquer adalah dua pendekatan yang berbeda untuk menyelesaikan masalah. Brute force umumnya berarti mencoba setiap kemungkinan solusi sampai menemukan solusi yang bekerja, sedangkan divide and conquer melibatkan membagi masalah menjadi bagian-bagian yang lebih kecil, menyelesaikan setiap bagian, dan kemudian menggabungkan hasilnya untuk mendapatkan solusi akhir.

Jika kita mengasumsikan bahwa Anda mencoba menyelesaikan suatu masalah yang melibatkan kurva Bezier (misalnya, menemukan titik pada kurva atau mengoptimalkan beberapa aspek dari kurva tersebut), maka :

- Brute force bisa melibatkan menghitung nilai kurva di banyak titik yang sangat rapat untuk menemukan solusi yang diinginkan (misalnya, titik terdekat dengan koordinat tertentu).
- Divide and conquer bisa melibatkan menggunakan algoritma seperti de Casteljau untuk secara rekursif membagi kurva menjadi bagian yang lebih kecil dan lebih mudah dihitung sampai titik yang diinginkan ditemukan.

Kompleksitas algoritma brute force biasanya $O(n^k)$, di mana n adalah jumlah kemungkinan solusi untuk setiap dimensi, dan k adalah jumlah dimensi dalam masalah. Pendekatan ini bisa menjadi sangat tidak efisien untuk masalah dengan ruang solusi yang besar.

Di sisi lain, algoritma divide and conquer sering memiliki kompleksitas yang jauh lebih baik karena cara mereka memecah masalah. Misalnya, algoritma seperti de Casteljau untuk kurva Bezier memiliki kompleksitas $O(n \log n)$ karena setiap pembagian memotong jumlah titik kontrol menjadi separuh. Ini secara signifikan lebih cepat daripada brute force untuk ruang solusi yang besar atau masalah yang kompleks.

Dalam praktiknya, pendekatan divide and conquer sering kali lebih disukai karena efisiensi dan skalabilitasnya yang lebih baik. Namun, ada situasi di mana brute force bisa lebih sederhana untuk diimplementasikan atau digunakan ketika efisiensi tidak menjadi masalah utama atau ruang solusi terbatas.

Dengan demikian, berdasarkan analisis tersebut, algoritma divide and conquer menunjukkan keunggulan dalam hal kinerja dan kompleksitas algoritma dibandingkan dengan algoritma brute force. Meskipun kedua algoritma menghasilkan hasil yang serupa secara visual, pemilihan algoritma harus memperhitungkan faktor-faktor kinerja dan kompleksitas yang terlibat, tergantung pada kebutuhan spesifik dari aplikasi dan ukuran masalah yang dihadapi.

BAB VI Kesimpulan

Kesimpulan mengenai perbandingan pendekatan brute force dengan divide and conquer dalam pemecahan masalah algoritmik dapat dirumuskan sebagai berikut :

Brute force adalah metode pemecahan masalah yang didasarkan pada pencarian eksaustif dan lengkap melalui semua kemungkinan kasus tanpa menggunakan strategi yang cerdik untuk memotong ruang pencarian. Ini berarti bahwa kompleksitas waktu seringkali sangat tinggi, terutama ketika ruang pencarian berkembang. Dalam konteks masalah dengan banyak variabel atau dimensi, ini sering diwakili dengan kompleksitas $O(n^k)$, yang menunjukkan bahwa waktu yang diperlukan untuk menjalankan algoritma meningkat secara eksponensial terhadap jumlah variabel k, ukuran ruang pencarian n.

Sebaliknya, divide and conquer adalah teknik yang mengurangi masalah menjadi sub-masalah yang lebih kecil, menyelesaikannya secara independen, dan kemudian menggabungkan hasilnya untuk mendapatkan solusi untuk masalah asli. Pendekatan ini biasanya lebih efisien karena dapat memotong ruang pencarian dan memecah masalah menjadi komponen-komponen yang dapat ditangani dengan lebih efisien, seringkali menghasilkan kompleksitas waktu seperti $O(n \log n)$ atau bahkan $O(n \log n)$ untuk beberapa masalah tertentu.

Dalam praktek, pendekatan divide and conquer umumnya lebih disukai atas brute force ketika kita berurusan dengan masalah skala besar atau ketika efisiensi sangat penting. Namun, brute force mungkin masih cocok untuk masalah dengan ruang pencarian yang terbatas, ketika kesederhanaan implementasi lebih diutamakan daripada efisiensi, atau ketika solusi awal yang cepat diperlukan tanpa memerlukan solusi optimal.

BAB VII Referensi

Munir, R. (2024). Algoritma Divide and Conquer. Diambil dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian2.pdf.

Munir, R. (2024). Tugas Kecil 2 Algoritma dan Struktur Data. Diambil dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Tucil2-2024.pdf.

Wikipedia contributors. (n.d.). Bézier curve. Wikipedia, The Free Encyclopedia. Diambil dari https://en.wikipedia.org/wiki/Bézier_curve.

Wikipedia kontributor. (t.t.). Kurva Bézier. Wikipedia, Ensiklopedia Bebas. Diambil dari https://id.wikipedia.org/wiki/Kurva_Bézier.

Halliday, S. (2017, February 15). Quadratic Bézier Curve Demo. Diambil dari https://simonhalliday.com/2017/02/15/quadratic-bezier-curve-demo/.

BAB VIII 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.		✓

Pranala Repository : Tucil2_10023519