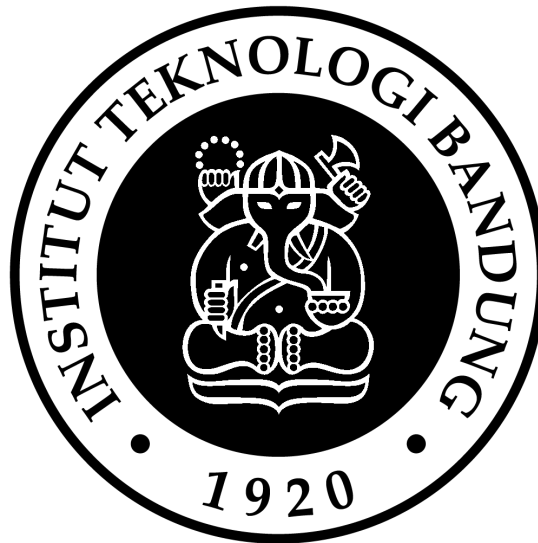


**IF2211-53 STRATEGI ALGORITMA
MEMBANGUN KURVA BÉZIER DENGAN
ALGORITMA TITIK TENGAH BERBASIS DIVIDE
AND CONQUER**

Laporan Tugas Kecil 2



Oleh:

Yosef Rafael Joshua / 13522133

**Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024**

DAFTAR ISI

DAFTAR ISI.....	2
BAB I PENDAHULUAN.....	3
BAB II ISI.....	3
2.1. Algoritma Brute Force.....	3
2.1.1. Implementasi dalam Python.....	3
2.2. Algoritma Divide and Conquer.....	5
2.2.1. Implementasi dalam Python.....	5
2.3. Pengujian.....	7
2.4. Analisis Perbandingan.....	15
BAB III PENUTUP.....	17
LAMPIRAN.....	18
DAFTAR PUSTAKA.....	19

BAB I PENDAHULUAN

Kurva Bézier adalah kurva yang terbentuk dari sekumpulan titik yang disebut control points. Kurva Bézier memiliki ciri khas yaitu kurva yang terbentuk adalah kurva yang mulus dan kontinu. Oleh karena keunikannya, kurva Bézier telah banyak diaplikasikan dalam bidang animasi, grafik komputer, modelling, serta aplikasi seperti Adobe Illustrator.

Terdapat banyak variasi dari kurva Bézier. Salah satu dari variasi tersebut adalah kurva Bézier kuadratik (*Quadratic Bézier Curve*). Pada versi ini, kurva Bézier dibentuk dengan tiga titik kontrol. P0 sebagai titik awal, P2 sebagai titik akhir, dan P1 sebagai titik kontrol di antara keduanya. Berikut adalah rumus untuk membuat suatu kurva Bézier kuadratik berdasarkan banyaknya jumlah titik pada kurva tersebut. Semakin banyak titik yang ada pada kurva, maka kurva akan semakin mulus.

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

Algoritma Divide and Conquer adalah metode memecahkan masalah (problem) dengan membagi-bagi masalah tersebut menjadi masalah-masalah yang lebih kecil (subproblem). Kemudian setiap masalah yang lebih kecil akan diselesaikan dan setiap solusi untuk masalah kecil tersebut akan digabungkan untuk menyelesaikan masalah utama. Secara umum Divide and Conquer memiliki tiga tahap. Tahap pertama adalah membagi masalah utama menjadi masalah-masalah yang lebih kecil namun masih memiliki kemiripan dengan masalah utama. Tahap kedua adalah menyelesaikan setiap masalah kecil tersebut. Dalam tahap ini masalah bisa langsung diselesaikan jika memang berukuran sangat kecil. Namun jika subproblem masih terlalu besar untuk diselesaikan secara langsung, maka dapat menggunakan rekursi. Tahap ketiga adalah menggabungkan setiap solusi dari *subproblem* menjadi satu solusi utama.

Pada Tugas Kecil Pemrograman ini, penulis diberi tugas untuk membuat program yang dapat menghasilkan kurva Bézier kuadratik dengan memanfaatkan algoritma Divide and Conquer. Kurva akan dibuat secara iteratif menggunakan algoritma Titik Tengah.

BAB II ISI

2.1. Algoritma Brute Force

2.1.1. Implementasi dalam Python

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

def quadratic_bezier_bf(p0, p1, p2, t):
    qx = (1 - t) ** 2 * p0[0] + 2 * (1 - t) * t * p1[0] + t ** 2 * p2[0]
    qy = (1 - t) ** 2 * p0[1] + 2 * (1 - t) * t * p1[1] + t ** 2 * p2[1]
    return qx, qy

def main():
    # input koordinat p0, p1, p2
    p0x, p0y = map(float, input("Masukkan koordinat p0x dan p0y.
Terpisahkan oleh spasi: ").split())
    p1x, p1y = map(float, input("Masukkan koordinat p1x dan p1y.
Terpisahkan oleh spasi: ").split())
    p2x, p2y = map(float, input("Masukkan koordinat p2x dan p2y.
Terpisahkan oleh spasi: ").split())
    jumlah_iterasi = int(input("Masukkan banyaknya iterasi: "))

    start_time = time.time()

    jumlah_titik = 2**jumlah_iterasi + 1

    p0 = (p0x, p0y)
    p1 = (p1x, p1y)
    p2 = (p2x, p2y)

    values = np.linspace(0, 1, jumlah_titik)
    BezierCurve = np.array([quadratic_bezier_bf(p0, p1, p2, v) for v in
values])

    plt.figure(figsize=(12, 8))
    BezierCurve = np.array(BezierCurve)
    plt.plot(BezierCurve[:, 0], BezierCurve[:, 1], label="Bezier Curve
```

```

with BF", color = "blue")
    plt.scatter([p0[0], p1[0], p2[0]], [p0[1], p1[1], p2[1]], color =
"red", label = "control points")
    plt.title("Quadratic Bezier Curve BF ({}
iterations)".format(jumlah_iterasi))
    plt.grid(True)
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.legend()
    plt.savefig('Bezier with BF.png')

    end_time = time.time()
    program_time = (end_time - start_time)

    print("waktu jalan program: {:.3f} detik".format(program_time))

if __name__ == "__main__":
    main()

```

Kode di atas adalah implementasi program untuk membuat kurva Bézier kuadratik dengan algoritma Brute Force. Program akan menerima koordinat P0, P1, dan P2. Kemudian akan menerima jumlah iterasi yang akan dilakukan. Kemudian akan dicari banyaknya titik yang akan terbentuk dengan rumus

$$\text{jumlah titik} = 2^x + 1$$

Dengan x adalah jumlah iterasi yang akan dilakukan. Lalu jumlah titik akan diolah dengan np.linspace dan untuk setiap titik akan dikenakan fungsi quadratic_bezier_bf sehingga didapatkan koordinat titik-titik yang diinginkan. Selanjutnya program melakukan plotting untuk titik-titik tersebut sehingga terbentuk kurva Bézier. Kompleksitas algoritma ini adalah

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

karena dilakukan pengulangan (looping) dua kali terhadap jumlah titik.

2.2. Algoritma Divide and Conquer

2.2.1. Implementasi dalam Python

```

import matplotlib.pyplot as plt
import numpy as np

```

```

import time

def quadratic_bezier_dnc(p0, p1, p2, iterasi):
    midLeft = ((p0[0] + p1[0]) / 2, (p0[1] + p1[1]) / 2)
    midRight = ((p1[0] + p2[0]) / 2, (p1[1] + p2[1]) / 2)
    midCenter = ((midLeft[0] + midRight[0]) / 2, (midLeft[1] +
midRight[1]) / 2)
    if(iterasi == 1):
        return[p0, midCenter, p2]
    else:
        leftDNC = quadratic_bezier_dnc(p0, midLeft, midCenter, iterasi -
1)
        rightDNC = quadratic_bezier_dnc(midCenter, midRight, p2, iterasi
- 1)

        return leftDNC + rightDNC[1:]

def main():
    # input koordinat p0, p1, p2
    p0x, p0y = map(float, input("Masukkan koordinat p0x dan p0y.
Terpisahkan oleh spasi: ").split())
    p1x, p1y = map(float, input("Masukkan koordinat p1x dan p1y.
Terpisahkan oleh spasi: ").split())
    p2x, p2y = map(float, input("Masukkan koordinat p2x dan p2y.
Terpisahkan oleh spasi: ").split())
    jumlah_iterasi = int(input("Masukkan banyaknya iterasi: "))

    start_time = time.time()

    p0 = (p0x, p0y)
    p1 = (p1x, p1y)
    p2 = (p2x, p2y)

    BezierCurve = quadratic_bezier_dnc(p0, p1, p2, jumlah_iterasi)

    plt.figure(figsize=(12, 8))
    BezierCurve = np.array(BezierCurve)
    plt.plot(BezierCurve[:, 0], BezierCurve[:, 1], label="Bezier Curve
with DnC", color = "blue")

```

```

plt.scatter([p0[0], p1[0], p2[0]], [p0[1], p1[1], p2[1]], color =
"red", label = "control points")
plt.title("Quadratic Bezier Curve DnC ({0}
iterations)".format(jumlah_iterasi))
plt.grid(True)
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.savefig('Bezier with DnC.png')

end_time = time.time()
program_time = (end_time - start_time)

print("waktu jalan program: {:.3f} detik".format(program_time))

if __name__ == "__main__":
    main()

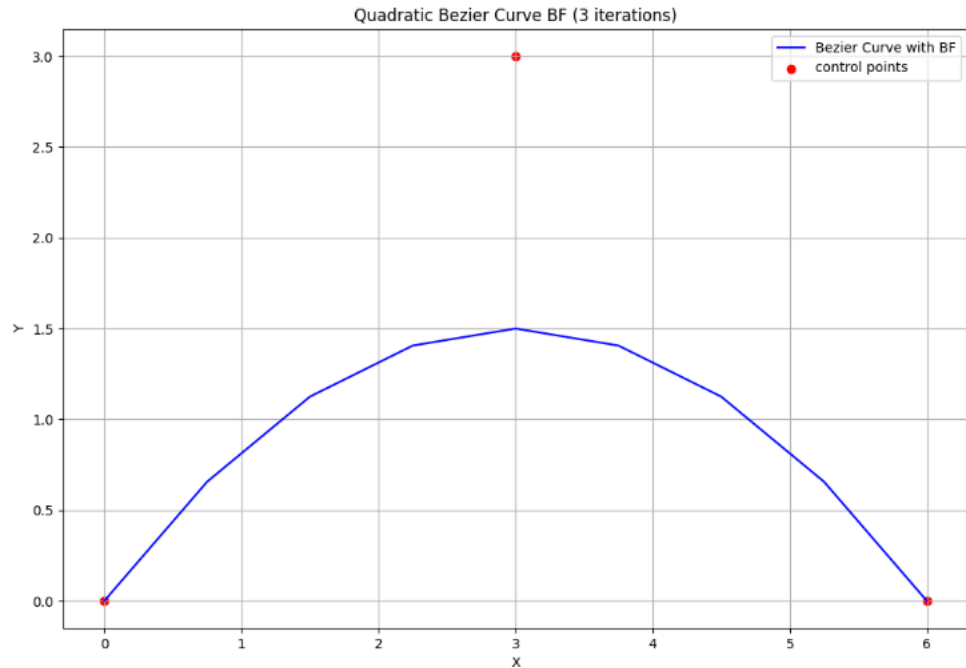
```

Kode di atas adalah implementasi program untuk membuat kurva Bézier kuadratik dengan algoritma Divide and Conquer. Program akan menerima input berupa koordinat P0, P1, P2, dan banyaknya iterasi yang akan dilakukan. Kemudian program akan menghasilkan koordinat titik-titik pada kurva Bézier secara rekursif dengan fungsi `quadratic_bezier_dnc`. Bagian Divide and Conquer terletak pada fungsi ini dimana jika jumlah iterasi lebih dari satu, maka program akan membagi titik-titik menjadi dua bagian yaitu bagian yang di sebelah kiri P1 (antara P0 dan P1) dan bagian yang di sebelah kanan P1 (antara P1 dan P2). Untuk setiap bagian ini program akan menjalankan fungsi secara rekursif lagi hingga didapatkan semua koordinat titik. Lalu program akan menggabungkan array titik di sebelah kiri dengan array titik yang ada di sebelah kanan. Setelah itu program akan melakukan plotting untuk membuat kurva Bézier. Kompleksitas algoritma ini adalah

$$O(n * \log(n))$$

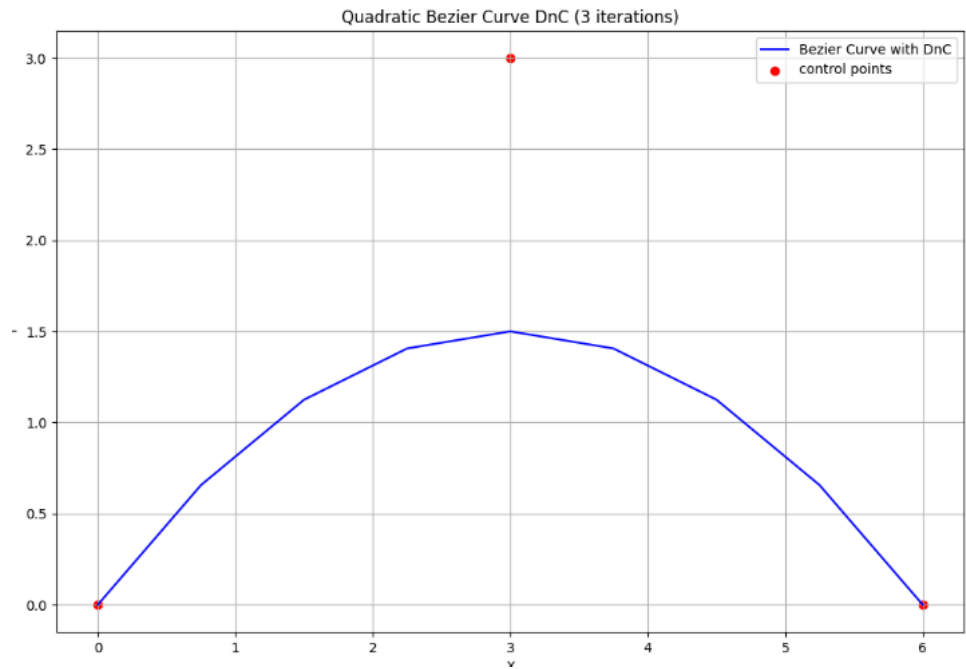
2.3. Pengujian

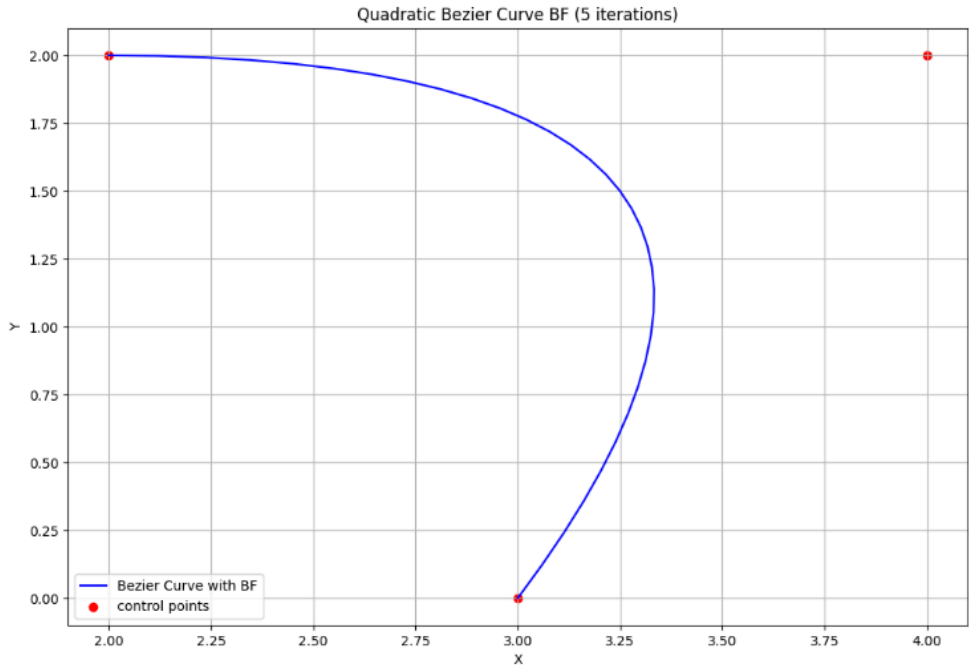
1	P0: 0,0 P1: 3,3 P2: 6,0 Iterasi: 3	Brute force:
---	---	--------------

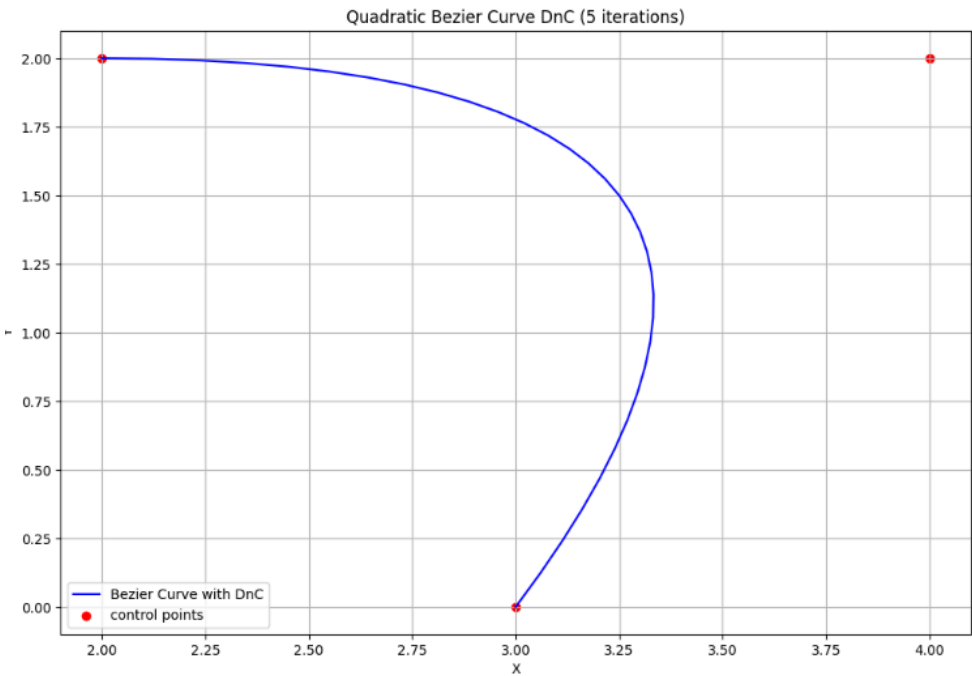
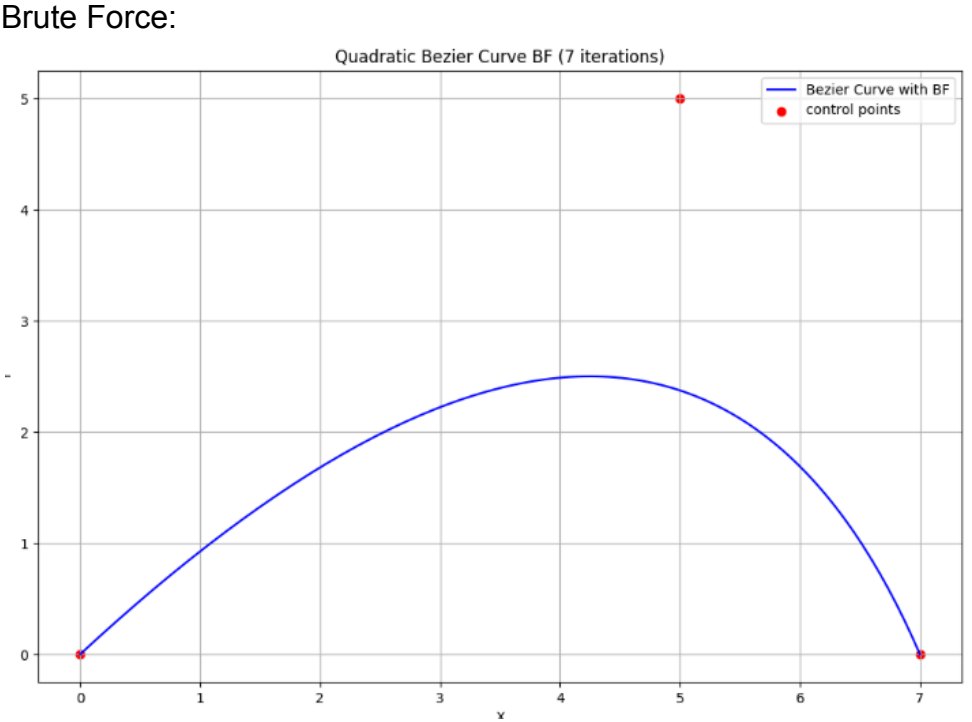


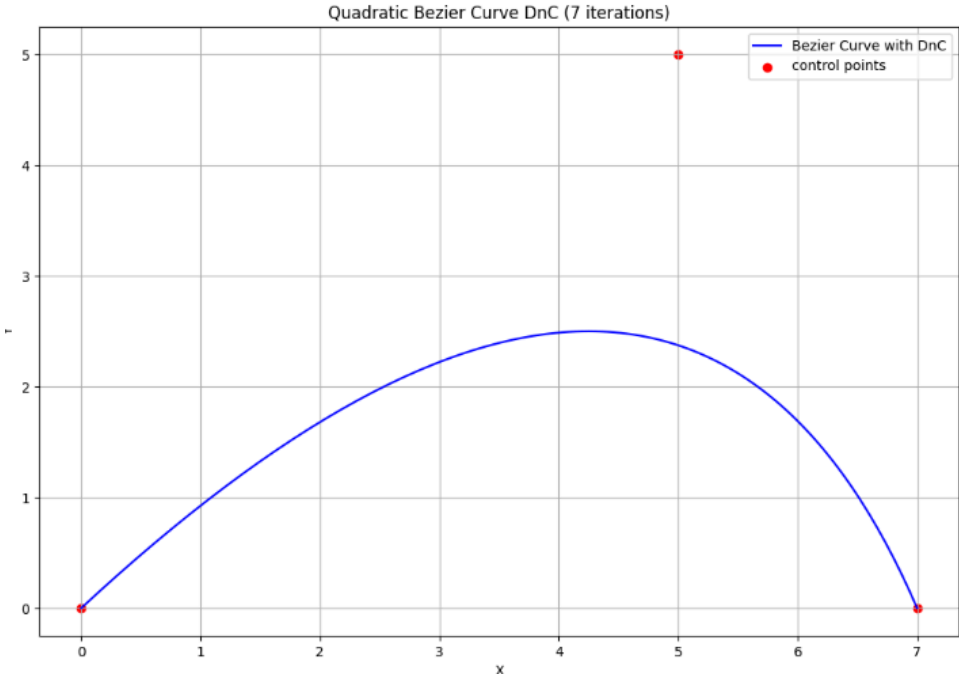
Masukkan koordinat p_0x dan p_0y . Terpisahkan oleh spasi: 0 0
 Masukkan koordinat p_1x dan p_1y . Terpisahkan oleh spasi: 3 3
 Masukkan koordinat p_2x dan p_2y . Terpisahkan oleh spasi: 6 0
 Masukkan banyaknya iterasi: 3
 waktu jalan program: 0.171 detik

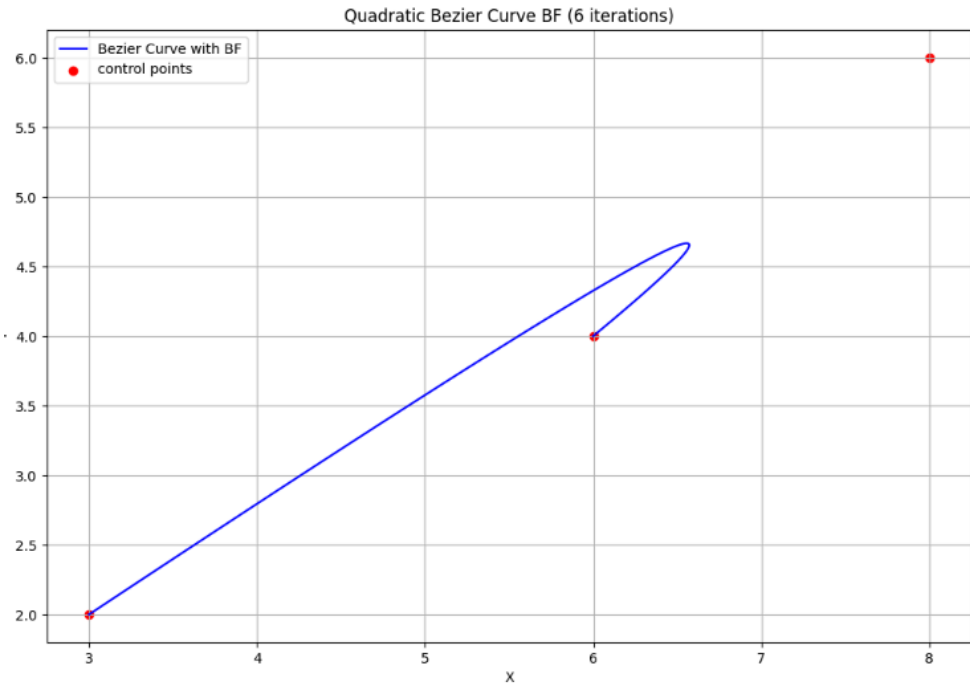
DnC:



		<p>Masukkan koordinat p_0x dan p_0y. Terpisahkan oleh spasi: 0 0</p> <p>Masukkan koordinat p_1x dan p_1y. Terpisahkan oleh spasi: 3 3</p> <p>Masukkan koordinat p_2x dan p_2y. Terpisahkan oleh spasi: 6 0</p> <p>Masukkan banyaknya iterasi: 3</p> <p>waktu jalan program: 0.184 detik</p>
2	P0: 2,2 P1: 4,2 P2: 3,0 Iterasi: 5	<p>Brute Force:</p>  <p>Masukkan koordinat p_0x dan p_0y. Terpisahkan oleh spasi: 2 2</p> <p>Masukkan koordinat p_1x dan p_1y. Terpisahkan oleh spasi: 4 2</p> <p>Masukkan koordinat p_2x dan p_2y. Terpisahkan oleh spasi: 3 0</p> <p>Masukkan banyaknya iterasi: 5</p> <p>waktu jalan program: 0.213 detik</p> <p>DnC:</p>

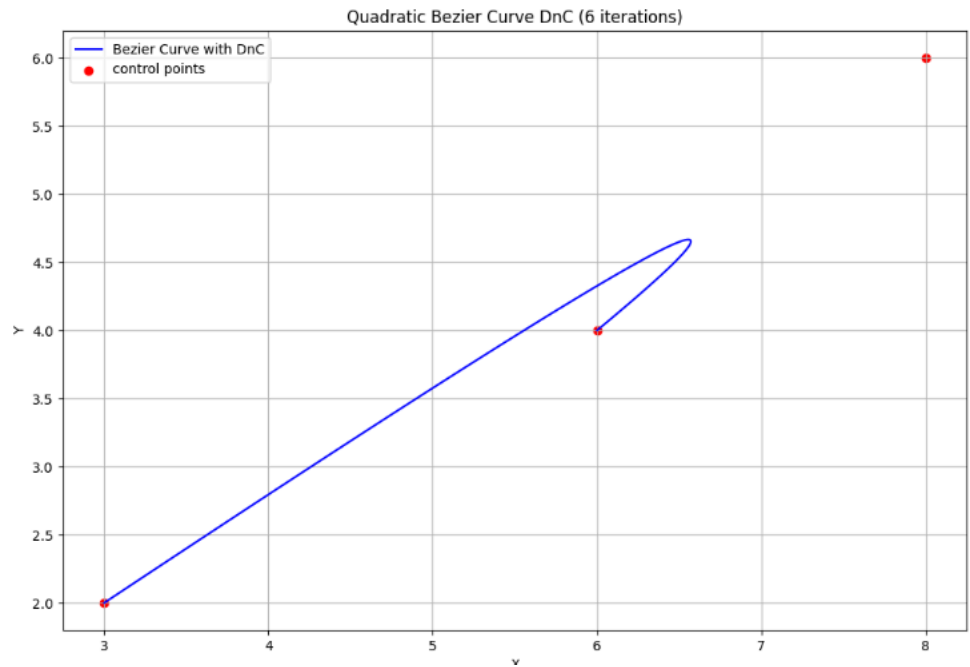
		<div data-bbox="446 205 1412 877"><p>Quadratic Bezier Curve DnC (5 iterations)</p><p>Masukkan koordinat p0x dan p0y. Terpisahkan oleh spasi: 2 2 Masukkan koordinat p1x dan p1y. Terpisahkan oleh spasi: 4 2 Masukkan koordinat p2x dan p2y. Terpisahkan oleh spasi: 3 0 Masukkan banyaknya iterasi: 5 waktu jalan program: 0.192 detik</p></div>
3	<div data-bbox="248 1136 373 1276"><p>P0: 0,0 P1: 5,5 P2: 7,0 Iterasi: 7</p></div>	<div data-bbox="446 1136 1412 1854"><p>Brute Force:</p><p>Quadratic Bezier Curve BF (7 iterations)</p></div>

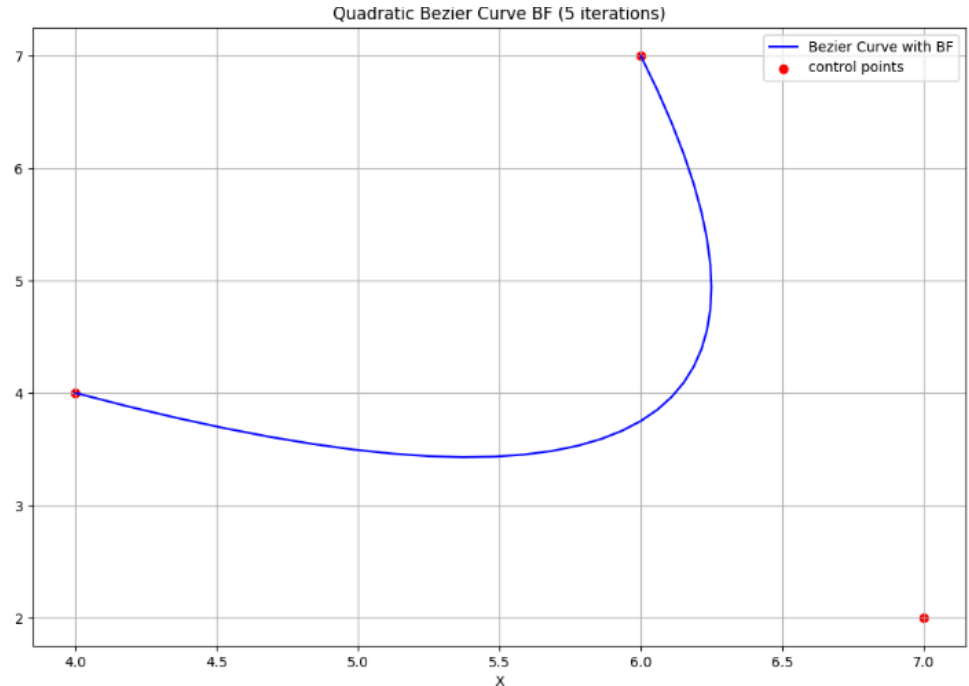
		<div>Masukkan koordinat p0x dan p0y. Terpisahkan oleh spasi: 0 0 Masukkan koordinat p1x dan p1y. Terpisahkan oleh spasi: 5 5 Masukkan koordinat p2x dan p2y. Terpisahkan oleh spasi: 7 0 Masukkan banyaknya iterasi: 7 waktu jalan program: 0.193 detik</div> <div>DnC:</div> <div></div> <div>Masukkan koordinat p0x dan p0y. Terpisahkan oleh spasi: 0 0 Masukkan koordinat p1x dan p1y. Terpisahkan oleh spasi: 5 5 Masukkan koordinat p2x dan p2y. Terpisahkan oleh spasi: 7 0 Masukkan banyaknya iterasi: 7 waktu jalan program: 0.178 detik</div>
4	P0: 6,4 P1: 8,6 P2: 3,2 Iterasi: 6	Brute Force:

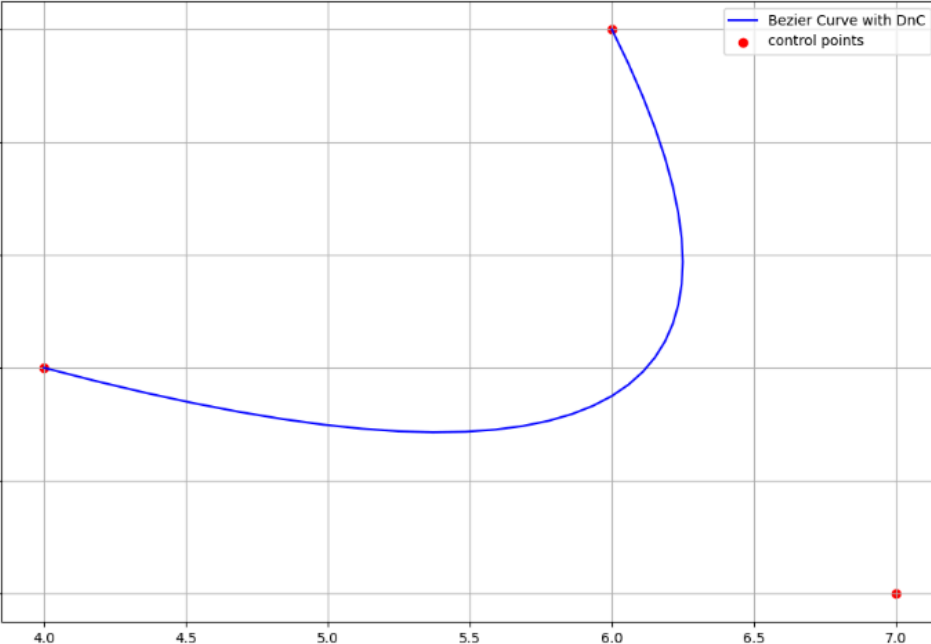
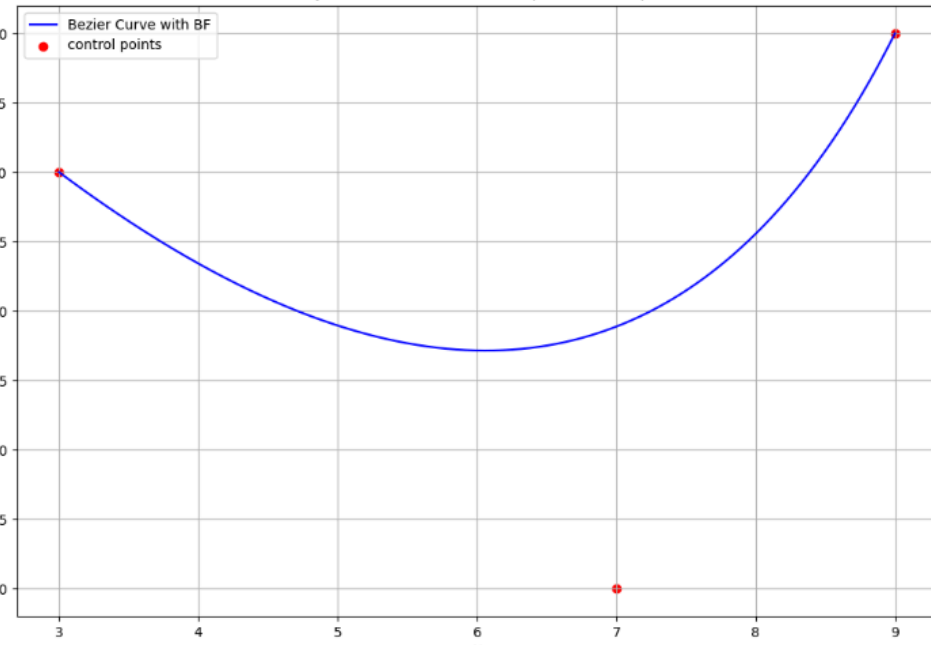


Masukkan koordinat p_0x dan p_0y . Terpisahkan oleh spasi: 6 4
 Masukkan koordinat p_1x dan p_1y . Terpisahkan oleh spasi: 8 6
 Masukkan koordinat p_2x dan p_2y . Terpisahkan oleh spasi: 3 2
 Masukkan banyaknya iterasi: 6
 waktu jalan program: 0.202 detik

DnC:

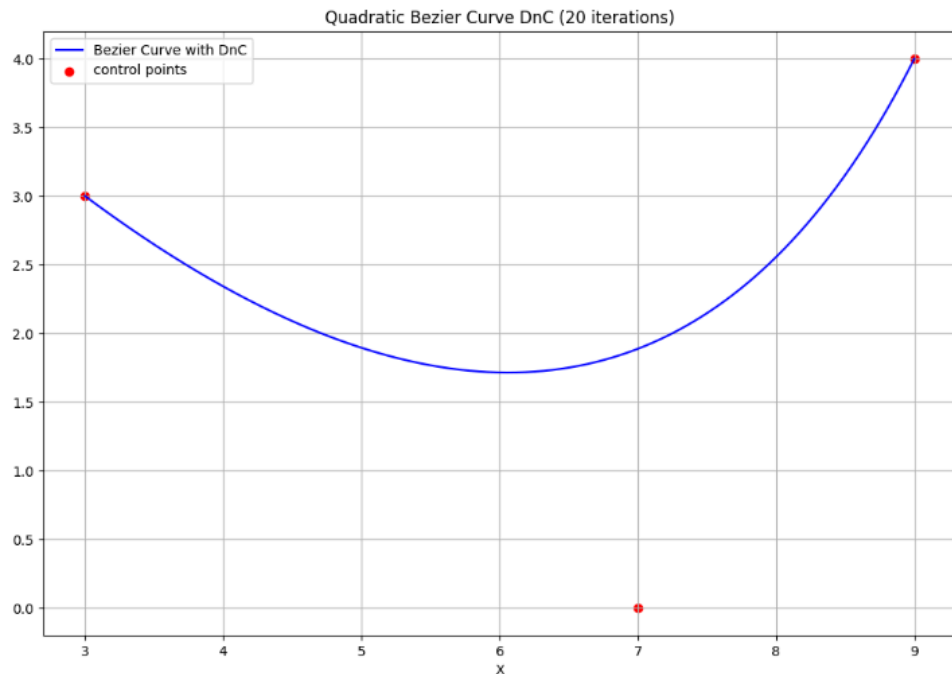


		<p>Masukkan koordinat p_0x dan p_0y. Terpisahkan oleh spasi: 6 4</p> <p>Masukkan koordinat p_1x dan p_1y. Terpisahkan oleh spasi: 8 6</p> <p>Masukkan koordinat p_2x dan p_2y. Terpisahkan oleh spasi: 3 2</p> <p>Masukkan banyaknya iterasi: 6</p> <p>waktu jalan program: 0.177 detik</p>
5	P0: 4,4 P1: 7,2 P2: 6,7 Iterasi: 5	<p>Brute Force:</p>  <p>Masukkan koordinat p_0x dan p_0y. Terpisahkan oleh spasi: 4 4</p> <p>Masukkan koordinat p_1x dan p_1y. Terpisahkan oleh spasi: 7 2</p> <p>Masukkan koordinat p_2x dan p_2y. Terpisahkan oleh spasi: 6 7</p> <p>Masukkan banyaknya iterasi: 5</p> <p>waktu jalan program: 0.347 detik</p> <p>DnC:</p>

		<div data-bbox="446 205 1412 892"><p>Quadratic Bezier Curve DnC (5 iterations)</p><p>Masukkan koordinat p0x dan p0y. Terpisahkan oleh spasi: 4 4 Masukkan koordinat p1x dan p1y. Terpisahkan oleh spasi: 7 2 Masukkan koordinat p2x dan p2y. Terpisahkan oleh spasi: 6 7 Masukkan banyaknya iterasi: 5 waktu jalan program: 0.194 detik</p></div>
6	<div data-bbox="251 1134 397 1281"><p>P0: 3,3 P1: 7,0 P2: 9,4 Iterasi: 20</p></div>	<div data-bbox="446 1134 1412 1858"><p>Brute Force:</p><p>Quadratic Bezier Curve BF (20 iterations)</p></div>

Masukkan koordinat p_0x dan p_0y . Terpisahkan oleh spasi: 3 3
Masukkan koordinat p_1x dan p_1y . Terpisahkan oleh spasi: 7 0
Masukkan koordinat p_2x dan p_2y . Terpisahkan oleh spasi: 9 4
Masukkan banyaknya iterasi: 20
waktu jalan program: 1.771 detik

DnC:



Masukkan koordinat p_0x dan p_0y . Terpisahkan oleh spasi: 3 3
Masukkan koordinat p_1x dan p_1y . Terpisahkan oleh spasi: 7 0
Masukkan koordinat p_2x dan p_2y . Terpisahkan oleh spasi: 9 4
Masukkan banyaknya iterasi: 20
waktu jalan program: 1.195 detik

2.4. Analisis Perbandingan

Berdasarkan hasil dari pengujian yang telah dilakukan, terlihat bahwa implementasi menggunakan algoritma Brute Force maupun Divide and Conquer dapat membuat kurva Bézier dengan waktu yang tidak jauh berbeda. Namun jika diperhatikan lebih lanjut, terlihat bahwa algoritma Divide and Conquer dapat membuat kurva Bézier dengan waktu yang lebih cepat untuk iterasi yang lebih banyak. Sedangkan algoritma Brute Force dapat membuat kurva Bézier dengan waktu yang lebih cepat untuk iterasi yang lebih sedikit.

Dari hasil pengujian yang diperoleh, algoritma Brute Force lebih cocok untuk membuat kurva Bézier dengan iterasi yang sedikit. Sedangkan algoritma

Divide and Conquer lebih cocok untuk membuat kurva Bézier dengan iterasi yang lebih banyak. Hasil ini sesuai dengan kompleksitas waktu masing-masing algoritma. Divide and Conquer memiliki kompleksitas waktu:

$$O(n * \log(n))$$

Sedangkan algoritma Brute Force memiliki kompleksitas waktu:

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

Maka untuk jumlah iterasi yang semakin banyak, algoritma Divide and Conquer akan menghasilkan solusi dengan waktu yang lebih cepat dibandingkan algoritma Brute Force.

BAB III PENUTUP

Dari tugas pemrograman ini dapat disimpulkan bahwa penulis telah berhasil membuat program yang dapat membuat kurva Bézier kuadratik dengan pendekatan secara Divide and Conquer dan secara Brute Force. Kurva Bézier dapat diimplementasikan dengan berbagai cara, beberapa diantaranya adalah Divide and Conquer dan Brute Force. Setiap cara memiliki kelebihan dan kekurangan masing-masing. Dari hasil uji coba yang telah dilakukan, pembuatan kurva Bézier dengan algoritma Divide and Conquer lebih optimal untuk kasus dengan jumlah iterasi yang banyak.

LAMPIRAN

Github : <https://github.com/Syfoe/Tucil-2-STIMA-13522133>

Checklist

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.		✓

DAFTAR PUSTAKA

- Rinaldi, M. (n.d). Algoritma Divide and Conquer (2024) Bagian 1.
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-\(2024\)-Bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian1.pdf)
- Rinaldi, M. (n.d). Algoritma Divide and Conquer (2024) Bagian 2.
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-\(2024\)-Bagian2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian2.pdf)
- Rinaldi, M. (n.d). Algoritma Divide and Conquer (2024) Bagian 3.
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-\(2024\)-Bagian3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian3.pdf)
- Rinaldi, M. (n.d). Algoritma Divide and Conquer (2024) Bagian 4.
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-\(2024\)-Bagian4.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian4.pdf)
- <https://mmrndev.medium.com/understanding-bézier-curves-f6eaa0fa6c7d>
- <https://www.geeksforgeeks.org/introduction-to-divide-and-conquer-algorithm-data-structure-and-algorithm-tutorials/>