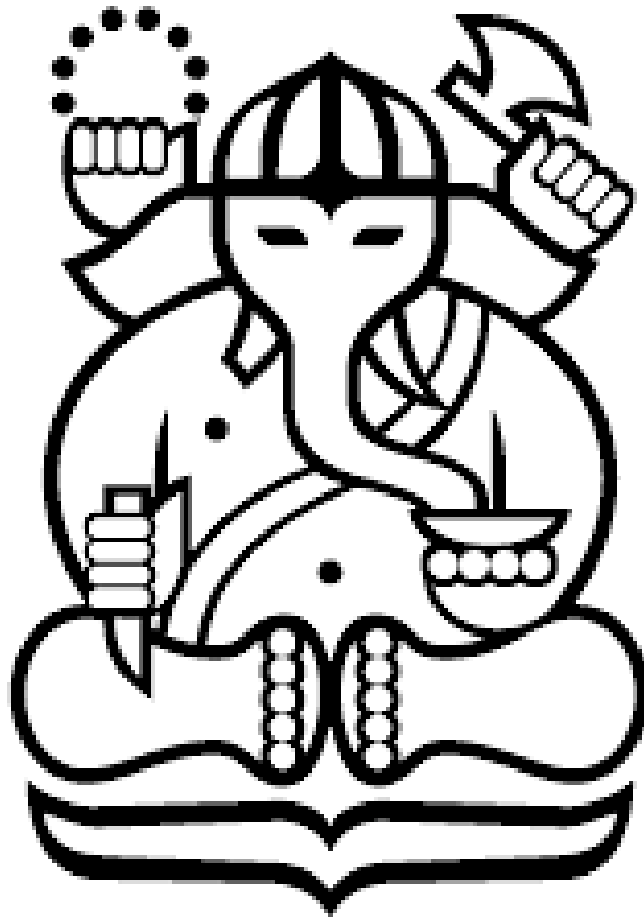


Laporan Tugas Kecil 2

IF2211 Strategi Algoritma

Membangun Kurva Bézier dengan Algoritma Titik Tengah berbasis
Divide and Conquer



Disusun oleh:

Dzaky Satrio Nugroho

13522059

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2024

Daftar Isi

Daftar Isi	2
Bab 1: Analisis dan implementasi dalam algoritma brute force	3
Bab 2: Analisis dan Implementasi Dalam Algoritma Divide and Conquer	4
Bab 3: Source Code Program	5
Bab 4: Uji Coba	6
Bab 5: Analisis Perbandingan Solusi Brute Force Dengan Divide And Conquer	7
Bab 6: Implementasi Bonus	8
Daftar Pustaka	9
Link Repository	9

Bab 1: Analisis dan implementasi dalam algoritma brute force

Persoalan ini dapat diselesaikan dengan pendekatan Brute Force. Jika diketahui n titik kontrol dan t titik. Maka dapat dicari dengan rumus :

$$\begin{aligned} \mathbf{B}(t) &= \sum_{i=0}^n \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i \\ &= (1-t)^n \mathbf{P}_0 + \binom{n}{1} (1-t)^{n-1} t \mathbf{P}_1 + \cdots + \binom{n}{n-1} (1-t) t^{n-1} \mathbf{P}_{n-1} + t^n \mathbf{P}_n, \quad 0 \leq t \leq 1 \end{aligned}$$

Jika terdapat t titik maka t akan diganti dengan 1/(t+1).

Bab 2: Analisis dan Implementasi Dalam Algoritma *Divide and Conquer*

Persoalan ini juga dapat diselesaikan dengan pendekatan Divide and Conquer. Algoritma ini bekerja sesuai dengan namanya, yaitu divide yang berarti memecah masalah menjadi persoalan yang lebih kecil tapi memiliki bentuk yang sama, conquer yang berarti menyelesaikan masalah jika sudah cukup kecil, dan yang terakhir adalah combine yang mana kita menyatukan jawaban dari permasalahan yang sudah kita pecah menjadi satu.

Untuk pembentukan bezier curve dapat dipecah menjadi masalah yang lebih kecil namun serupa, sehingga dapat diselesaikan dengan pendekatan Divide and Conquer. Algoritma dapat menghasilkan titik tengah dari titik kontrol dan titik tengah dari titik tengah tersebut secara rekursif sebanyak iterasi yang diinginkan oleh pengguna.

Bab 3: Source Code Program

3.1. Main Program

```
import numpy as np
import matplotlib.pyplot as plt
from dnc import *
from bf import *

# Fungsi untuk mengecek apakah input adalah float yang valid atau
tidak
def isFloat(strings):
    if(strings[0] == "-"):
        strings = strings[1:]
    if strings.replace(".", "").isnumeric():
        return True
    else:
        return False

# Input banyak titik
inputs = input("Masukkan banyak titik (minimal 3) : ")
while not inputs.isdigit() or int(inputs) < 3 :
    print("Masukan tidak valid!")
    inputs = input("Masukkan banyak titik (minimal 3) : ")

manyPoint = int(inputs)

# Input titik
points = []

for i in range(1,manyPoint+1):
    print(f"Masukkan titik ke-{i} (x y) : ", end="")
    inputs = list(map(str,input().split()))
    while len(inputs) != 2 or not isFloat(inputs[0]) or not
isFloat(inputs[1]) :
        print("Masukan tidak valid!")
        print(f"Masukkan titik ke-{i} (x y) : ", end="")
        inputs = list(map(str,input().split()))
    points.append([float(inputs[0]),float(inputs[1])])

# Input banyak iterasi
inputs = input("Masukkan banyak iterasi (minimal 1) : ")
while not inputs.isdigit() or int(inputs) < 1 :
    print("Masukan tidak valid!")
    inputs = input("Masukkan banyak iterasi (minimal 1) : ")

iterasi = int(inputs)
```

```

# Memilih algoritma yang digunakan
inputs = input("Pilih algoritma yang digunakan (1. Divide and
Conquer 2. Brute Force) : ")
while inputs != "1" and inputs != "2" :
    print("Masukan tidak valid!")
    inputs = input("Pilih algoritma yang digunakan (1. Divide and
Conquer 2. Brute Force) : ")

if inputs == "1" :
    DNCmain(points,iterasi)
else :
    BFMain(points,iterasi)

```

3.2. Divide and Conquer

```

import matplotlib.pyplot as plt
import time

# Fungsi untuk mendapatkan titik tengah dari 2 buah titik
def getMidPoint(p1,p2):
    return [(p1[0]+p2[0])/2,(p1[1]+p2[1])/2]

# Fungsi untuk menghasilkan titik tengah diantara titik kontrol
kemudian
# mencari lagi titik tengahnya secara rekursif
def getPoints(points):
    if len(points) <= 1 :
        return points
    else :
        ret = [points[0]]
        temp = []
        for i in range(len(points) - 1) :
            temp.append(getMidPoint(points[i],points[i+1]))
        ret.extend(getPoints(temp))
        ret.append(points[-1])
        return ret

# Fungsi Divide and Conquer
def divideAndConquer(points, iterate):
    if iterate == 0 :
        return []
    else :
        points = getPoints(points)
        iterate -= 1
        idxMid = len(points)//2
        return divideAndConquer(points[0:idxMid+1],iterate) +

```

```

[points[idxMid]] + divideAndConquer(points[idxMid:],iterate)

def DNCmain(points,iterate) :
    # Mulai waktu perhitungan
    start = time.time()

    # Mulai program Divide and Conquer
    pointAkhir = []
    pointAkhir.append(points[0])
    pointAkhir.extend(divideAndConquer(points,iterate))
    pointAkhir.append(points[-1])

    # Stop waktu perhitungan
    end = time.time()

    # Menghitung lama waktu yang diperlukan
    totalWaktu = end - start

    # Menggambar kurva
    for i in range(len(points)-1) :
        plt.plot([points[i][0], points[i+1][0]], [points[i][1],
points[i+1][1]], marker='o', linestyle='-', color="blue", label =
"Titik Kontrol" if i == 0 else "")
        for i in range(len(pointAkhir)-1) :
            plt.plot([pointAkhir[i][0], pointAkhir[i+1][0]],
[pointAkhir[i][1], pointAkhir[i+1][1]], marker='o', linestyle='-',
color="red", label = "Kurva Bezier" if i == 0 else "")

        x_min, x_max = plt.xlim()
        y_min, y_max = plt.ylim()
        plt.text((x_min+x_max)/2, y_max*1.2, f"Waktu program:
{totalWaktu:.4f} detik", fontsize=14)

    plt.grid(True)
    plt.legend(loc = "best")
    plt.show()

```

3.3. Brute Force

```

import matplotlib.pyplot as plt
import math
import time

# Fungsi Brute Force
def BruteForce(points,iterate):
    ret = []
    t = 1 / (iterate*1.0 + 1)
    for i in range(iterate+2):

```

```

        x = 0.0
        y = 0.0
        for j in range(len(points)):
            x +=
float(math.comb(len(points)-1,j)*((1-t*i)**(len(points)-1-j))*((t*i)
)**j)*points[j][0])
            y +=
float(math.comb(len(points)-1,j)*((1-t*i)**(len(points)-1-j))*((t*i)
)**j)*points[j][1])
            ret.append([x,y])
        return ret

# Fungsi bruteforce
def BFMain(points,iterate):
    # Mulai waktu perhitungan
    start = time.time()

    # Mulai program Brute Force
    pointAkhir = BruteForce(points,iterate)

    # Stop waktu perhitungan
    end = time.time()

    # Menghitung lama waktu yang diperlukan
    totalWaktu = end - start

    # Menggambar kurva
    for i in range(len(points)-1) :
        plt.plot([points[i][0], points[i+1][0]], [points[i][1],
points[i+1][1]], marker='o', linestyle='-', color="blue", label =
"Titik Kontrol" if i == 0 else "")
        for i in range(len(pointAkhir)-1) :
            plt.plot([pointAkhir[i][0], pointAkhir[i+1][0]],
[pointAkhir[i][1], pointAkhir[i+1][1]], marker='o', linestyle='-',
color="red", label = "Kurva Bezier" if i == 0 else "")

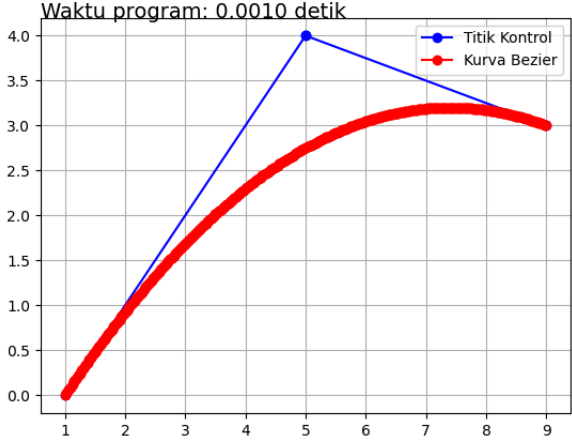
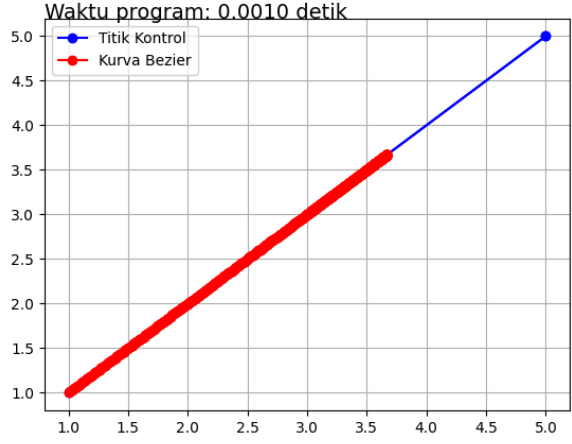
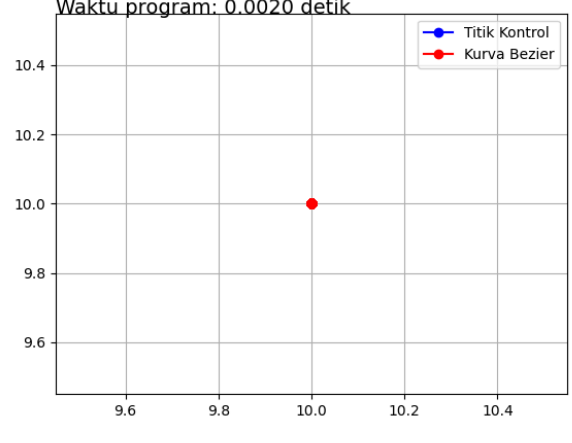
    x_min, x_max = plt.xlim()
    y_min, y_max = plt.ylim()
    plt.text((x_min+x_max)/2, y_max*1.2, f"Waktu program:
{totalWaktu:.4f} detik", fontsize=14)

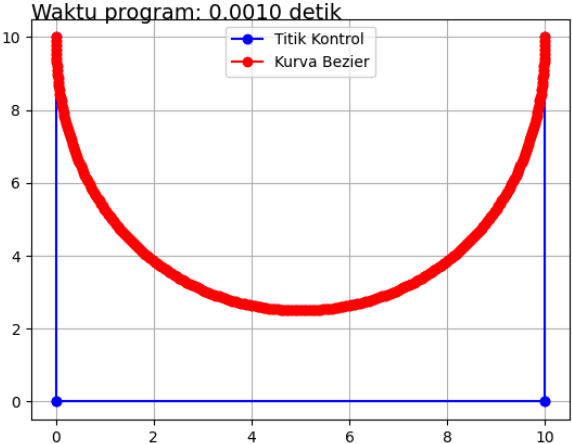
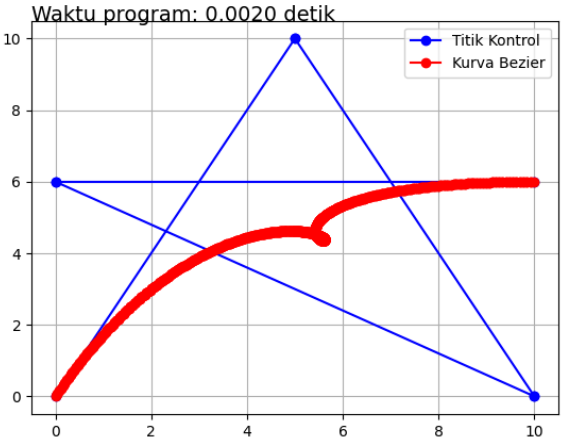
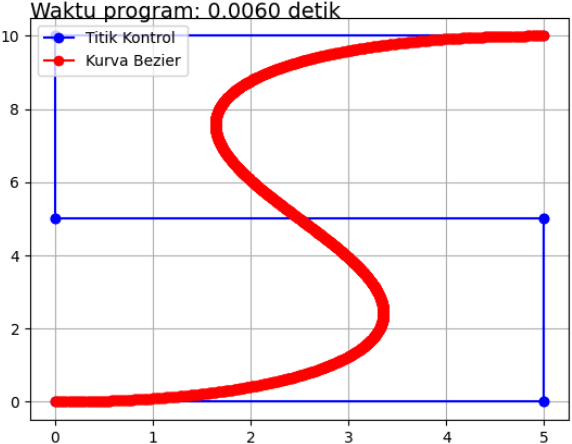
    plt.grid(True)
    plt.legend(loc = "best")
    plt.show()

```


Bab 4: Uji Coba

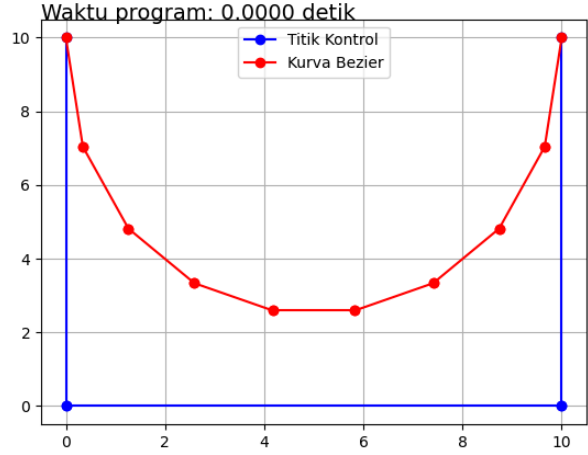

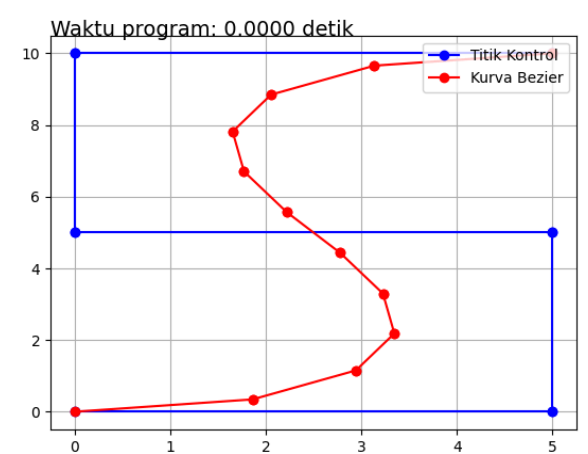
4.1. Divide and Conquer

<p>Waktu program: 0.0010 detik</p>  <p>Legend: Titik Kontrol (blue line with dots), Kurva Bezier (red line with dots)</p>	<p>Masukkan banyak titik (minimal 3) : 3 Masukkan titik ke-1 (x y) : 1 0 Masukkan titik ke-2 (x y) : 5 4 Masukkan titik ke-3 (x y) : 9 3 Masukkan banyak iterasi (minimal 1) : 8 Pilih algoritma yang digunakan (1. Divide and Conquer 2. Brute Force) : 1</p>
<p>Waktu program: 0.0010 detik</p>  <p>Legend: Titik Kontrol (blue line with dots), Kurva Bezier (red line with dots)</p>	<p>Masukkan banyak titik (minimal 3) : 3 Masukkan titik ke-1 (x y) : 1 1 Masukkan titik ke-2 (x y) : 5 5 Masukkan titik ke-3 (x y) : 3 3 Masukkan banyak iterasi (minimal 1) : 9 Pilih algoritma yang digunakan (1. Divide and Conquer 2. Brute Force) : 1</p>
<p>Waktu program: 0.0020 detik</p>  <p>Legend: Titik Kontrol (blue line with dots), Kurva Bezier (red line with dots)</p>	<p>Masukkan banyak titik (minimal 3) : 3 Masukkan titik ke-1 (x y) : 10 10 Masukkan titik ke-2 (x y) : 10 10 Masukkan titik ke-3 (x y) : 10 10 Masukkan banyak iterasi (minimal 1) : 10 Pilih algoritma yang digunakan (1. Divide and Conquer 2. Brute Force) : 1</p>

<p>Waktu program: 0.0010 detik</p> 	<p>Masukkan banyak titik (minimal 3) : 4 Masukkan titik ke-1 (x y) : 0 10 Masukkan titik ke-2 (x y) : 0 0 Masukkan titik ke-3 (x y) : 10 0 Masukkan titik ke-4 (x y) : 10 10 Masukkan banyak iterasi (minimal 1) : 8 Pilih algoritma yang digunakan (1. Divide and Conquer 2. Brute Force) : 1</p>
<p>Waktu program: 0.0020 detik</p> 	<p>Masukkan banyak titik (minimal 3) : 5 Masukkan titik ke-1 (x y) : 0 0 Masukkan titik ke-2 (x y) : 5 10 Masukkan titik ke-3 (x y) : 10 0 Masukkan titik ke-4 (x y) : 0 6 Masukkan titik ke-5 (x y) : 10 6 Masukkan banyak iterasi (minimal 1) : 9 Pilih algoritma yang digunakan (1. Divide and Conquer 2. Brute Force) : 1</p>
<p>Waktu program: 0.0060 detik</p> 	<p>Masukkan banyak titik (minimal 3) : 6 Masukkan titik ke-1 (x y) : 0 0 Masukkan titik ke-2 (x y) : 5 0 Masukkan titik ke-3 (x y) : 5 5 Masukkan titik ke-4 (x y) : 0 5 Masukkan titik ke-5 (x y) : 0 10 Masukkan titik ke-6 (x y) : 5 10 Masukkan banyak iterasi (minimal 1) : 10 Pilih algoritma yang digunakan (1. Divide and Conquer 2. Brute Force) : 1</p>

4.2. Brute Force

<p>Waktu program: 0.0000 detik</p> <p>Titik Kontrol Kurva Bezier</p>	<p>Masukkan banyak titik (minimal 3) : 3 Masukkan titik ke-1 (x y) : 1 0 Masukkan titik ke-2 (x y) : 5 4 Masukkan titik ke-3 (x y) : 9 3 Masukkan banyak iterasi (minimal 1) : 8 Pilih algoritma yang digunakan (1. Divide and Conquer 2. Brute Force) : 2</p>
<p>Waktu program: 0.0000 detik</p> <p>Titik Kontrol Kurva Bezier</p>	<p>Masukkan banyak titik (minimal 3) : 3 Masukkan titik ke-1 (x y) : 1 1 Masukkan titik ke-2 (x y) : 5 5 Masukkan titik ke-3 (x y) : 3 3 Masukkan banyak iterasi (minimal 1) : 9 Pilih algoritma yang digunakan (1. Divide and Conquer 2. Brute Force) : 2</p>
<p>Waktu program: 0.0000 detik</p> <p>Titik Kontrol Kurva Bezier</p>	<p>Masukkan banyak titik (minimal 3) : 3 Masukkan titik ke-1 (x y) : 10 10 Masukkan titik ke-2 (x y) : 10 10 Masukkan titik ke-3 (x y) : 10 10 Masukkan banyak iterasi (minimal 1) : 10 Pilih algoritma yang digunakan (1. Divide and Conquer 2. Brute Force) : 2</p>

<p>Waktu program: 0.0000 detik</p> 	<p>Masukkan banyak titik (minimal 3) : 4 Masukkan titik ke-1 (x y) : 0 10 Masukkan titik ke-2 (x y) : 0 0 Masukkan titik ke-3 (x y) : 10 0 Masukkan titik ke-4 (x y) : 10 10 Masukkan banyak iterasi (minimal 1) : 8 Pilih algoritma yang digunakan (1. Divide and Conquer 2. Brute Force) : 2</p>
<p>Waktu program: 0.0020 detik</p> 	<p>Masukkan banyak titik (minimal 3) : 5 Masukkan titik ke-1 (x y) : 0 0 Masukkan titik ke-2 (x y) : 5 10 Masukkan titik ke-3 (x y) : 10 0 Masukkan titik ke-4 (x y) : 0 6 Masukkan titik ke-5 (x y) : 10 6 Masukkan banyak iterasi (minimal 1) : 9 Pilih algoritma yang digunakan (1. Divide and Conquer 2. Brute Force) : 2</p>
<p>Waktu program: 0.0000 detik</p> 	<p>Masukkan banyak titik (minimal 3) : 6 Masukkan titik ke-1 (x y) : 0 0 Masukkan titik ke-2 (x y) : 5 0 Masukkan titik ke-3 (x y) : 5 5 Masukkan titik ke-4 (x y) : 0 5 Masukkan titik ke-5 (x y) : 0 10 Masukkan titik ke-6 (x y) : 5 10 Masukkan banyak iterasi (minimal 1) : 10 Pilih algoritma yang digunakan (1. Divide and Conquer 2. Brute Force) : 2</p>

Bab 5: Analisis Perbandingan Solusi Brute Force Dengan Divide And Conquer

Dari uji coba pada bab 4 didapatkan bahwa dengan waktu yang sama algoritma Divide and Conquer dapat menghasilkan titik yang lebih banyak daripada algoritma Brute Force. Jadi Solusi Divide and Conquer lebih cepat dalam menghasilkan Bezier Curve dibandingkan dengan solusi Brute Force.

Bab 6: Implementasi Bonus

Bonus yang diimplementasikan adalah generalisasi masalah, sehingga bisa menerima n buah titik kontrol.

Daftar Pustaka

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/Kizaaaa/Tucil2_13522059