# LAPORAN TUGAS KECIL 2

# IF2211 STRATEGI ALGORITMA

# MEMBANGUN KURVA BEZIER DENGAN ALGORITMA TITIK TENGAH BERBASIS DIVIDE AND CONQUER


Disusun oleh:

Julian Caleb S. - 13522099

Christopher Brian - 13522106

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

# DAFTAR ISI

## DESKRIPSI PERSOALAN

Kurva Bezier adalah sebuah kurva halus yang dibuat dengan menggabungkan titik-titik kontrol yang menentukan bentuk dan arah sebuah kurva. Kurva bezier banyak diaplikasikan di dunia nyata, seperti pen tool, pembuatan animasi, dan pembuatan produk dengan presisi. Kurva Bezier didefinisikan dengan beberapa titik kontrol P0 hingga Pn (dengan n adalah order, n = 1,2,3,...) dengan P0 dan Pn menjadi ujung kurva dan sisanya menjadi titik kontrol antara yang umumnya tidak terletak pada kurva.

Kurva Bezier dengan 2 titik kontrol disebut sebagai Kurva Bezier Linier, 3 titik kontrol disebut sebagai Kurva Bezier Kuadratik, dan 4 titik kontrol sebagai Kurva Bezier Cubic. Kurva Bezier Kuadratik dapat dibuat dengan menghubungkan P0 - P1 dan P1 - P2, kemudian mencari titik tengah kedua garis, Q0 dan Q1 dengan rumus parametrik sebagai berikut

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

$$Q_1 = B(t) = (1-t)P_1 + tP_2, \qquad t \in [0,1]$$

dengan t sebagai jarak antara P dan Q. Gabungkan Q0 dan Q1, menggunakan rumus yang sama

$$R_0 = B(t) = (1-t)Q_0 + tQ_1, \qquad t \in [0,1]$$

dan hasil dari rumus tersebut adalah Kurva Bezier Kuadratik. Rumus tersebut juga dapat disubstitusikan menjadi

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

Tugas kecil 2 Stima ini meminta mahasiswa untuk mengimplementasikan pembuatan Kurva Bezier Kuadratik, namun dengan pendekatan Brute Force dan Divide and Conquer. Sebagai bonus, mahasiswa diminta untuk membuat animasi setiap proses pembuatan titik dan garis serta pengimplementasian untuk Kurva Bezier dengan n > 3.

## ANALISIS DAN IMPLEMENTASI BRUTE FORCE

Untuk n buah titik kontrol, koordinat dari titik kurva Bezier adalah sebagai berikut.

$$C(n-1,\,0)(1-t)^{n}(t)^{0}P_{0} + C(n-1,\,1)(1-t)^{n-1}(t)^{1}P_{1} + \ \ldots\ + C(n-1,\,n-1)(1-t)^{0}(t)^{n-1}P_{n-1}$$

Untuk menggambarkan kurva Bezier dengan pendekatan *brute force*, digunakan dua pengulangan, pengulangan pertama untuk setiap iterasi titik kurva dan pengulangan kedua untuk menghitung nilai setiap suku dalam persamaan kurva Bezier. Pengulangan kedua terdapat di dalam pengulangan pertama.

Pada pengulangan pertama, dilakukan iterasi untuk setiap titik pada kurva Bezier yang akan dibuat. Pada setiap iterasi, nilai koordinat x dan y untuk titik akan di-*reset* menjadi 0 (akan lebih lanjut dijelaskan alasannya di penjelasan pengulangan kedua). Iterasi dilakukan sebanyak jumlah titik dalam kurva (n titik antara + 2 buah titik kontrol, yaitu titik kontrol pertama dan terakhir).

Pada pengulangan kedua, dilakukan iterasi untuk setiap suku dalam persamaan kurva Bezier. Pada setiap iterasi, dilakukan penambahan nilai suku ke nilai koordinat x dan y setiap titik dalam kurva Bezier yang akan dibuat. Suku ke-i dari persamaan kurva Bezier bernilai sebagai berikut, di mana n adalah jumlah titik kontrol dan k adalah nomor titik kontrol.

$$C(n-1,\,k)(1-t)^{n}(t)^{0}P_{k}$$

Pada setiap iterasi, akan ditambahkan nilai suku ke-i untuk menghitung nilai koordinat x dan y dari setiap titik pada kurva Bezier. Untuk algoritma akan diimplementasikan menggunakan bahasa Python.

## ANALISIS DAN IMPLEMENTASI DIVIDE AND CONQUER

Divide and Conquer adalah sebuah algoritma yang membagi persoala menjadi beberapa subpersoalan kemudian menyelesaikan masing-masing subpersoalan dan menggabungkan semuanya kembali. Algoritma Divide and Conquer dibagi menjadi 3 tahap, yaitu Divide, pembagian persoalan menjadi beberapa subpersoalan, Conquer, penyelesaian masing-masing subpersoalan, dan Combine, yang menggabungkan solusi-solusi membentuk solusi persoalan semula. Divide and Conquer pada umumnya menggunakan skema rekursif, jadi pada kasus Kurva Bezier, akan diterapkan juga penggunaan rekursif.

Sama seperti Brute Force, akan digunakan bahasa pemrograman Python. Misal fungsi utama DnC kurvaBezier disebuat DrawBezierCurve yang menerima array of points (titik) berupa (x,y) dan iteration (jumlah iterasi). Mari membahas Conquernya terlebih dahulu yaitu dengan menentukan titik tengah, di mana jika pada rumus titik yang dibuat merupakan parametrik terhadap t, pada DnC akan ditetapkan titik tengah tepat diantara 2 buah titik control point, atau dengan kata lain, t = 0.5. Hal ini berlaku untuk setiap control point awal, yang kemudian antara titik-titik tengah tersebut akan dicari lagi titik-titik tengah, yang diulangi hingga ditemukan 1 titik tengah terakhir. Sebagai contoh, untuk 3 titik p0, p1, dan p2 (yang akan menjadi Kurva Bezier Kuadratik), untuk BezierCurve([p0, p1, p2], 1), akan dicari titik tengah antara p0 dengan p1, misal q0, dan p1 dengan p2, misal q1, dan dilanjutkan dengan titik tengah antara q0 dengan q1, misal q2. Hapus titik tengah yang lama yaitu p1, didapatkan [p0, q0, q2, q1, p2]. Kemudian, jika kita bedah dan hanya sambungkan titik-titik ujung kurva (p0 dan p2), dengan titik tengah yang paling baru, akan didapatkan [p0, q2, p2], yang sudah merupakan cikal bakal dari kurva bezier, atau bisa juga disebut DnC iterasi pertama.

Dari sini, dapat dilakukan bagian Dividenya, yaitu dengan mengolah sebelah kiri kurva dengan sebelah kanan kurva. Kembali pada kasus 3 titik sebagai contoh, kita dapat melakukan divide dengan memanggil kembali fungsi BezierCurve, dengan sebelah kiri kurva melalui BezierCurve([p0, q0, q2], iteration-1) dilanjutkan sebelah kanan kurva

melalui BezierCurve([q2, q1, p2], iteration-1), untuk berapapun iteration. Fungsi BezierCurve akan terus dijalankan sebanyak iterasi yang akan menjadi input oleh pengguna, sehingga basisnya adalah ketika sudah iterasi terakhir atau sama dengan 0.

Terakhir, untuk combine, akan dilakukan return gabungan antara BezierCurve kiri, titik tengah, dan BezierCurve kanan sebagai array. Kemudian, menggunakan matplotlib.pyplot, akan dilukis garis antara masing-masing titik i menuju i+1 untuk banyak titik-1 (Array hasil terurut dari kiri ke kanan sesuai dengan keinginan pengguna pada input).

Algoritma yang sama dapat juga digunakan untuk n titik, n >= 3, dengan Conquernya melakukan rekursif sebanyak n-1 kali, mengambil semua titik-titik ujung dan menghapus titik-titik tengah yang sudah tidak terpakai, didapatkan titik sebanyak 2n-1, yang dapat dibagi 2 (titik paling tengah dipakai di keduanya sebagai ujung kanan untuk kurva kiri dan ujung kiri untuk kurva kanan) untuk melakukan Divide. Divide dan Combine sendiri berjalan sama.

# SOURCE CODE PROGRAM BRUTE FORCE

```python
import matplotlib.pyplot as plt
import math
from typing import List, Tuple
import time

# Fungsi untuk menggambarkan garis diantara dua titik
# Input titik pertama dan titik kedua
def draw_line(p0 : (float, float), p1 : (float, float), color: str):
    arr = []
    arr.append(p0)
    arr.append(p1)
    x = [point[0] for point in arr]
    y = [point[1] for point in arr]
    plt.plot(x, y, marker = 'o', linestyle = '-', color = color)

# Fungsi rekursif untuk membuat kurva Bezier
# Input berupa list titik kontrol dan jumlah titik antara titik kontrol pertama dan titik kontrol terakhir)
def draw_bezier_curve_bf(control_points: List[Tuple[float, float]], iteration: int):
    start_time = time.time()
    arr_of_curve_points = []
    num_control_points = len(control_points)
    for p in range(num_control_points - 1):
        draw_line(control_points[p], control_points[p + 1], "blue")
    if (iteration > 0):
        # t merupakan interval titik antara
        # misal terdapat 4 titik antara, intervalnya adalah 1 dibagi 4 + 1, yaitu 0.2
        t = float(1 / (iteration + 1))
        # iterasi sebanyak jumlah titik antara ditambah dua titik kontrol, yaitu titik kontrol pertama dan titik kontrol terakhir
        for i in range (iteration + 2):
            # Reset nilai koordinat x dan y
            x = 0.0
            y = 0.0
            for j in range(num_control_points):
                # Penambahan nilai setiap suku persamaan kurva Bezier ke dalam nilai koordinat x dan y setiap titik dalam kurva
                x += float(math.comb(num_control_points - 1, j) * ((1 - t * i) ** (num_control_points - 1 - j)) * ((t * i) ** j) * control_points[j][0])
                y += float(math.comb(num_control_points - 1, j) * ((1 - t * i) ** (num_control_points - 1 - j)) * ((t * i) ** j) * control_points[j][1])
            arr_of_curve_points.append((x, y))
        # Gambarkan garis antar titik dalam kurva Bezier
        for k in range(iteration + 1):
            draw_line(arr_of_curve_points[k], arr_of_curve_points[k + 1], "red")
        end_time = time.time()
        elapsed_time = end_time - start_time
        plt.gca().set_aspect('equal', adjustable='box')
        x_min, x_max = plt.xlim()
        y_min, y_max = plt.ylim()
        center_x = (x_min + x_max) / 2
        center_y = y_max
        plt.text(center_x, center_y, f"Elapsed time: {elapsed_time:.4f} seconds", fontsize=12, ha='center')
        plt.grid(True)
        plt.show()
```

**SOURCE CODE PROGRAM DIVIDE AND CONQUER**

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

# Fungsi untuk menampilkan garis antar 2 titik
def DrawLine(p0, p1) :
    arr = []
    arr.append(p0)
    arr.append(p1)
    x = [point[0] for point in arr]
    y = [point[1] for point in arr]
    plt.plot(x, y, marker='o', linestyle='-', color="black")

# Fungsi untuk membuat titik tengah antar 2 titik
def MakeNewPoint(p0, p1) :
    x = 0.5*p0[0] + 0.5*p1[0]
    y = 0.5*p0[1] + 0.5*p1[1]
    return (x,y)
```

```python
# -- CONQUER --
# Fungsi untuk secara rekursif mencari titik tengah (1) antar 2 titik
# control point, kemudian mencari titik tengah (2) antar hasil
# masing-masing titik tengah (1). Hal ini diulangi secara rekursif hingga
# berakhir dengan 1 titik tengah terakhir.
# Kemudian, akan disatukan dari titik paling kiri masing-masing array input
# rekursi, titik paling tengah, lanjut dengan titik paling kanan masing-masing
# array input rekursi. Untuk n titik, hasil akhirnya adalah (2n-1) titik
def PopulatePoints(controlPoints) :
    if (len(controlPoints) <= 1) :
        return controlPoints
    else :
        temp1 = []
        temp1.append(controlPoints[0])
        temp2 = []
        for i in range (len(controlPoints) - 1) :
            temp2.append(MakeNewPoint(controlPoints[i], controlPoints[i+1]))
        result = PopulatePoints(temp2)
        temp1.extend(result)
        temp1.append(controlPoints[-1])
        return temp1
```

```python
# Algoritma utama secara divide and conquer
# Control points input akan dipopulate sesuai dengan fungsi PopulatePoints,
# di mana untuk n titik, dihasilkan 2n - 1 titik hasil.
# -- DIVIDE --
# Titik hasil kemudian akan diproses secara dipisah dan rekursif, dengan meng-
# gunakan titik tengah, yaitu pada index midArrIdx = (2n-1)//2, dan dibagi
# pemanggilan fungsi dari idx 0 hingga midArrIdx (DrawBezierCurve kiri)
# dan idx midArrIdx hingga last index (DrawBezierCurve kanan).
# Divide dilakukan hingga iteration, yang setiap pemanggilannya -1,
# mencapai nilai 0.
# -- COMBINE --
# Jika mencapai basis, fungsi akan mengembalikan empty array. Di luar itu,
# akan dilakukan proses combine yaitu antara DrawBezierCurve kiri, titik tengah
# dan DrawBezierCurve kanan.
def DrawBezierCurve(controlPoints, iteration) :
    print("Iterasi mundur ke:", iteration)
    if (iteration > 0) :
        print("Sebelum populate: ", controlPoints)
        newControlPoints = PopulatePoints(controlPoints)
        print("Hasil populate:", newControlPoints)
        iteration -= 1
        midArrIdx = (len(newControlPoints)//2) # + 1 - 1 karena index
        print("Left branch:", newControlPoints[0:midArrIdx+1])
        print("Right branch:", newControlPoints[midArrIdx:])
        return DrawBezierCurve(newControlPoints[0:midArrIdx+1], iteration) + [newControlPoints[midArrIdx]] + DrawBezierCurve(newControlPoints[midArrIdx:], iteration)
    else :
        return []
```

```python
# Initialization
def BezierMain(controlPoints, iteration) :
    # Print titik awal
    print("Control points awal:", controlPoints)

    # Memulai perhitungan waktu
    start_time = time.time()
    print(start_time)

    # Divide and Conquer
    finalArrayPoints = []
    finalArrayPoints.append(controlPoints[0])
    finalArrayPoints.extend(DrawBezierCurve(controlPoints, iteration))
    finalArrayPoints.append(controlPoints[-1])

    # Menampilkan kurva bezier
    for i in range (len(controlPoints) - 1) :
        DrawLine(controlPoints[i], controlPoints[i + 1])
    for i in range (len(finalArrayPoints) - 1) :
        DrawLine(finalArrayPoints[i], finalArrayPoints[i + 1])

    # Menghentikan perhitungan waktu
    end_time = time.time()
    print(end_time)
    elapsed_time = end_time - start_time
    plt.gca().set_aspect('equal', adjustable='box')
    x_min, x_max = plt.xlim()
    y_min, y_max = plt.ylim()
    center_x = (x_min + x_max) / 2
    center_y = y_max
    plt.text(center_x, center_y, f"Elapsed time: {elapsed_time:.4f} seconds", fontsize=12, ha='center')

    plt.grid(True)
    plt.show()
```

```python
# Mau visualisasi?
print("\nWanna see the visualization?")
answer = int(input("No (0) / Yes (1): "))

if (answer == 1) :
    for i in range (iteration) :
        # Hapus array
        finalArrayPoints.clear()

        # Divide and Conquer
        finalArrayPoints.append(controlPoints[0])
        finalArrayPoints.extend(DrawBezierCurve(controlPoints, iteration))
        finalArrayPoints.append(controlPoints[-1])

        # Menampilkan kurva bezier
        for j in range (len(controlPoints) - 1) :
            DrawLine(controlPoints[j], controlPoints[j + 1])
        for j in range (len(finalArrayPoints) - 1) :
            DrawLine(finalArrayPoints[j], finalArrayPoints[j + 1])
        plt.grid(True)
        plt.show()
```

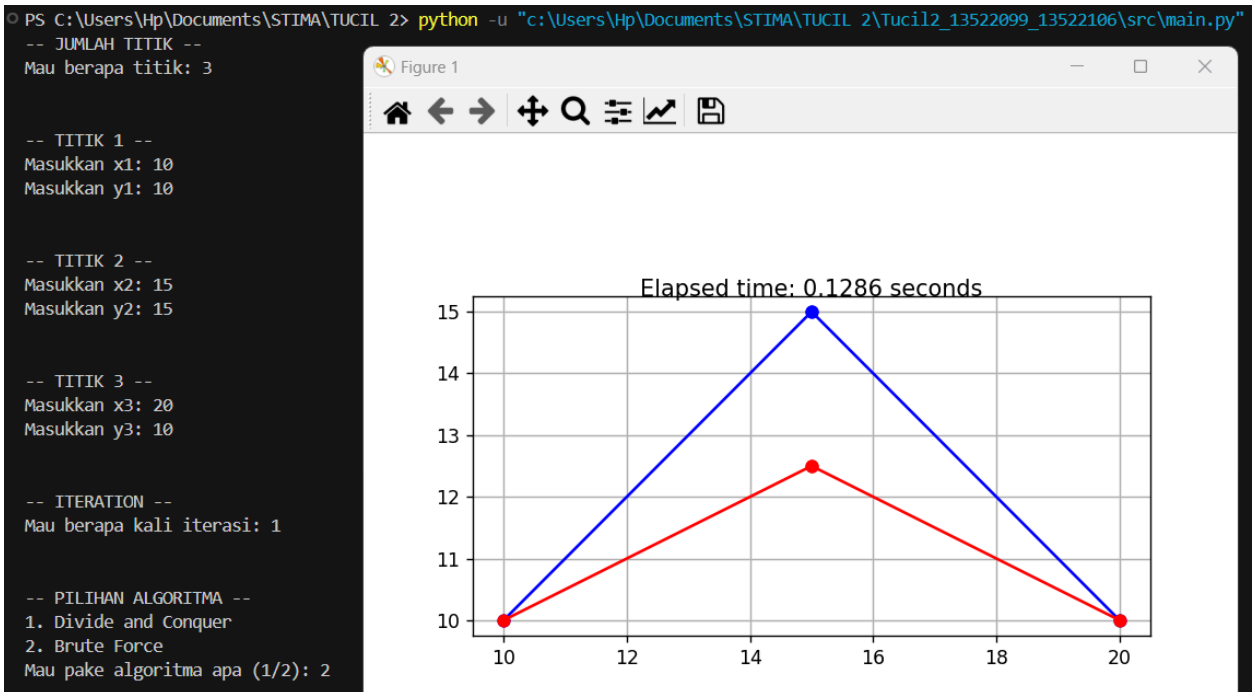
## SOURCE CODE MAIN

```python
import matplotlib.pyplot as plt
from dncbonus import BezierMain
from bruteforce import draw_bezier_curve_bf

# Meminta input jumlah titik
print("-- JUMLAH TITIK --")
n = int(input("Mau berapa titik: "))
print("\n")

# Meminta input titik
controlPoints = []
for i in range (n) :
    print(f"-- TITIK {i+1} --")
    x = float(input(f"Masukkan x{i+1}: "))
    y = float(input(f"Masukkan y{i+1}: "))
    controlPoints.append((x,y))
    print("\n")

# Meminta input banyak iterasi
print("-- ITERATION --")
iteration = int(input("Mau berapa kali iterasi: "))
print("\n")

# Memilih algoritma
print("-- PILIHAN ALGORITMA --")
print("1. Divide and Conquer\n2. Brute Force")
choice = int(input("Mau pake algoritma apa (1/2): "))
while (choice != 1) and (choice != 2):
    print("Pilihan algoritma tidak valid")
    choice = int(input("Mau pake algoritma apa (1/2): "))
print("\n")

# Eksekusi
if choice == 1:
    BezierMain(controlPoints, iteration)
else:
    draw_bezier_curve_bf(controlPoints, iteration)
```

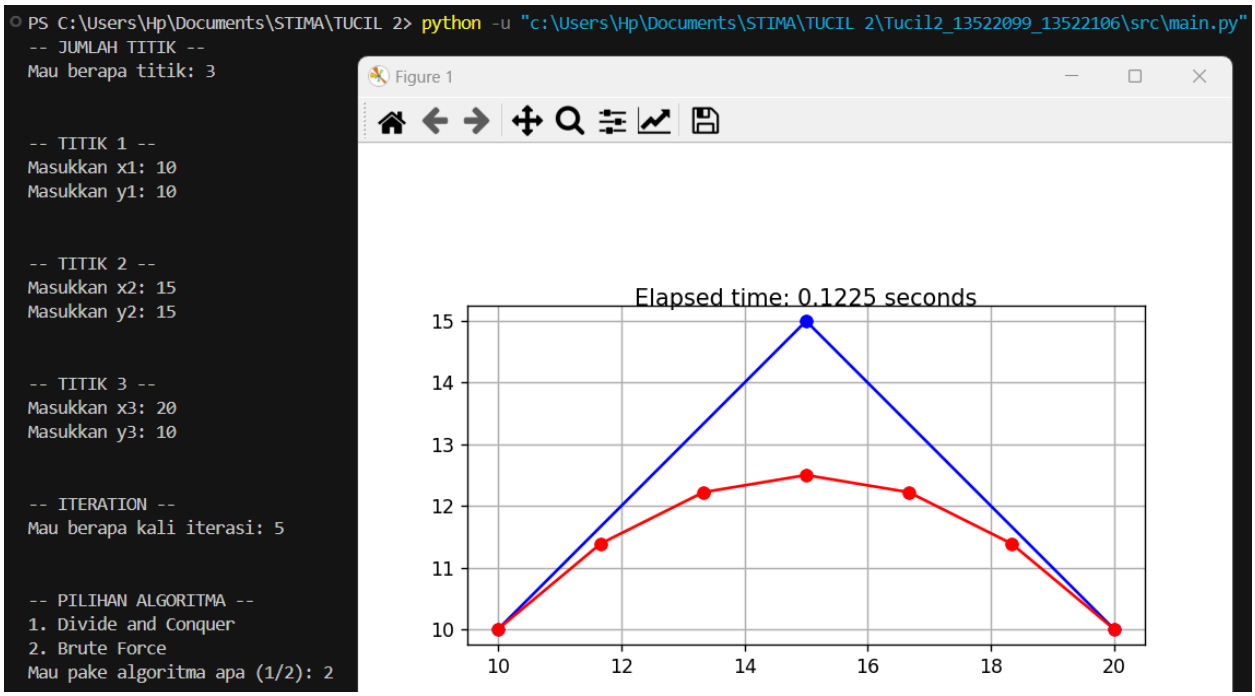
## TESTING

Brute Force

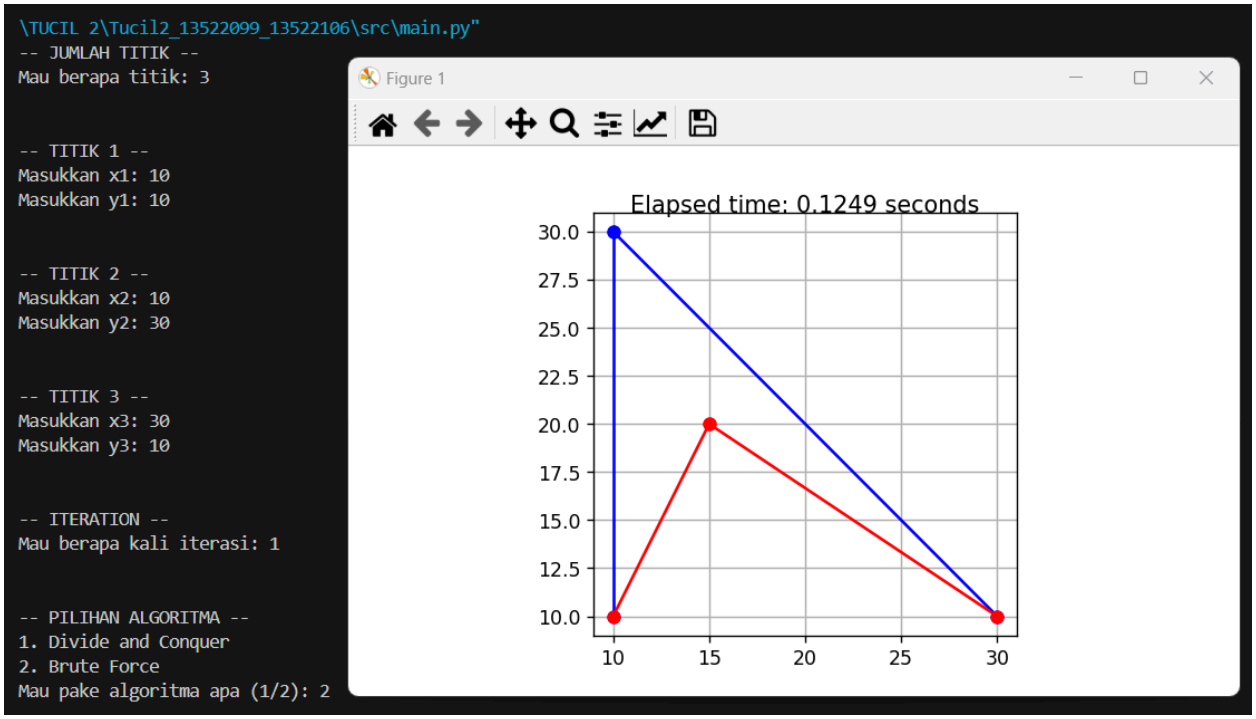
3 titik, (10, 10), (15, 15), (20, 10), 1 iterasi




3 titik, (10, 10), (15, 15), (20, 10), 5 iterasi



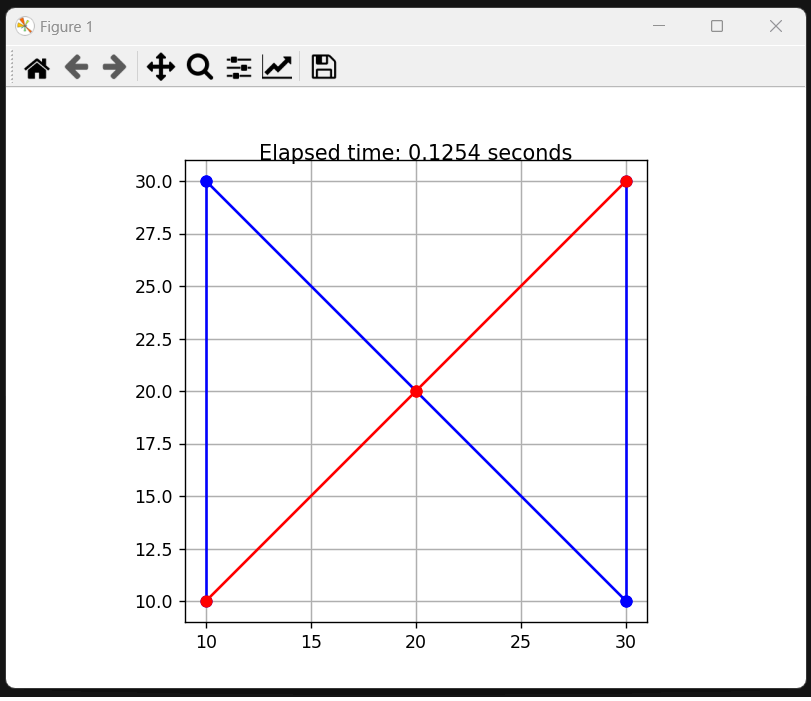
3 titik, (10, 10), (10, 30), (30, 10), 1 iterasi




3 titik, (10, 10), (10, 30), (30, 10), 5 iterasi

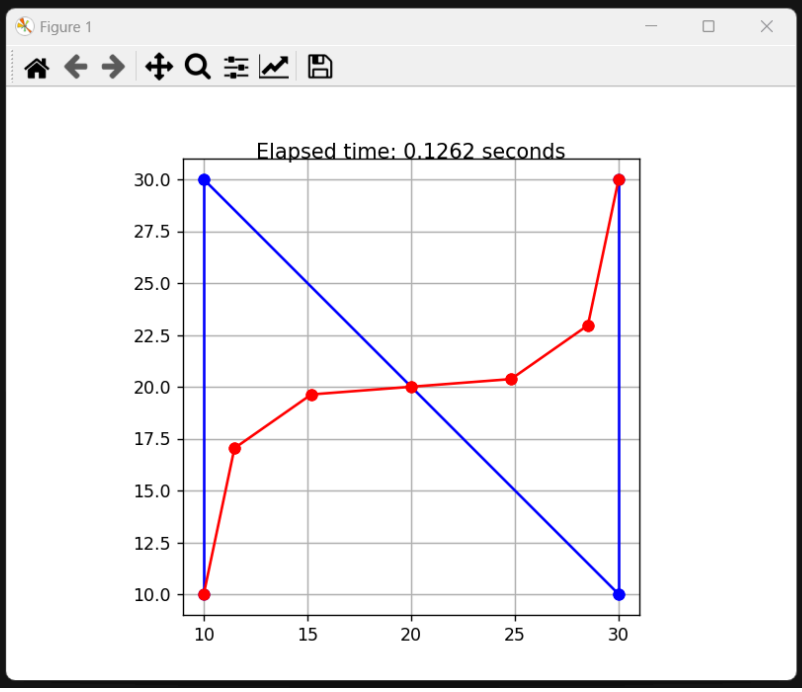
4 titik, (10, 10), (10, 30), (30, 10), (30, 30), 1 iterasi

```
\TUCIL 2\Tucil2_13522099_13522106\src\main.py"
-- JUMLAH TITIK --
Mau berapa titik: 4


-- TITIK 1 --
Masukkan x1: 10
Masukkan y1: 10


-- TITIK 2 --
Masukkan x2: 10
Masukkan y2: 30


-- TITIK 3 --
Masukkan x3: 30
Masukkan y3: 10


-- TITIK 4 --
Masukkan x4: 30
Masukkan y4: 30


-- ITERATION --
Mau berapa kali iterasi: 1


-- PILIHAN ALGORITMA --
1. Divide and Conquer
2. Brute Force
Mau pake algoritma apa (1/2): 2
```



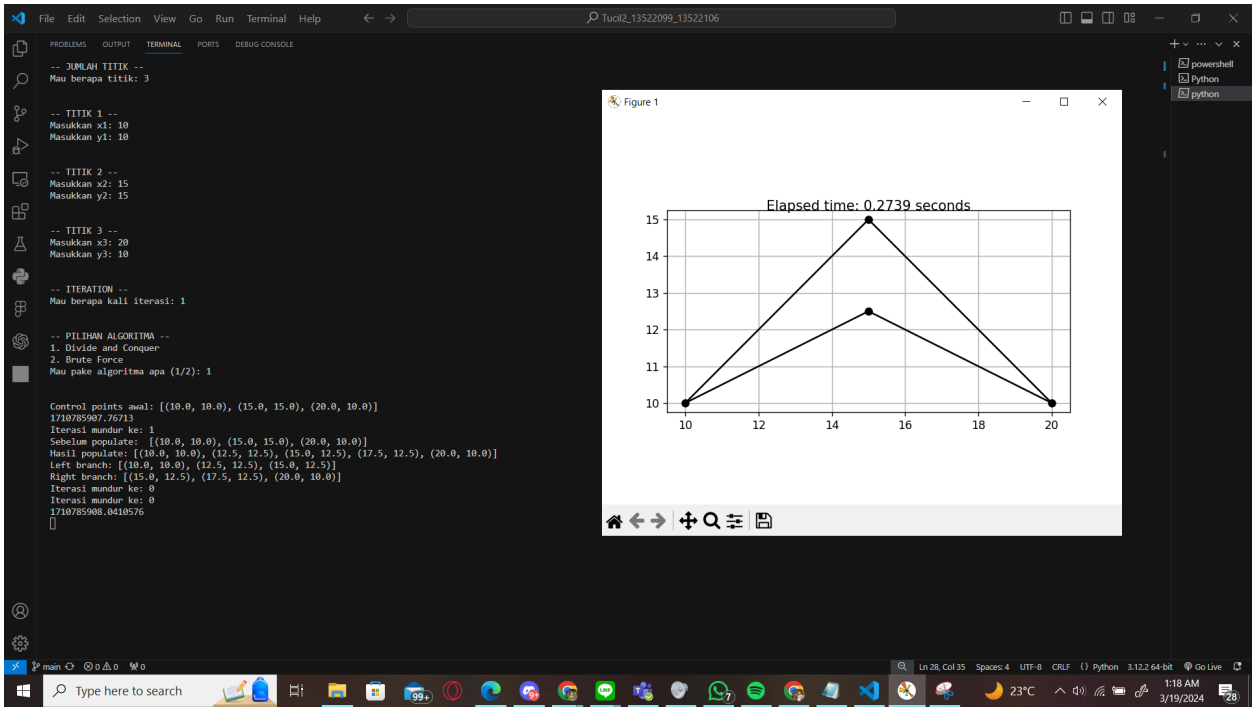

4 titik, (10, 10), (10, 30), (30, 10), (30, 30), 5 iterasi

```
PS C:\Users\Hp\Documents\STIMA\TUCIL 2> python -u "c:\Users\Hp\Documents\STIMA\TUCIL 2\Tucil2_13522099_13522106\src\main.py"
  -- JUMLAH TITIK --
  Mau berapa titik: 4


  -- TITIK 1 --
  Masukkan x1: 10
  Masukkan y1: 10


  -- TITIK 2 --
  Masukkan x2: 10
  Masukkan y2: 30


  -- TITIK 3 --
  Masukkan x3: 30
  Masukkan y3: 10


  -- TITIK 4 --
  Masukkan x4: 30
  Masukkan y4: 30


  -- ITERATION --
  Mau berapa kali iterasi: 5


  -- PILIHAN ALGORITMA --
  1. Divide and Conquer
  2. Brute Force
  Mau pake algoritma apa (1/2): 2
```

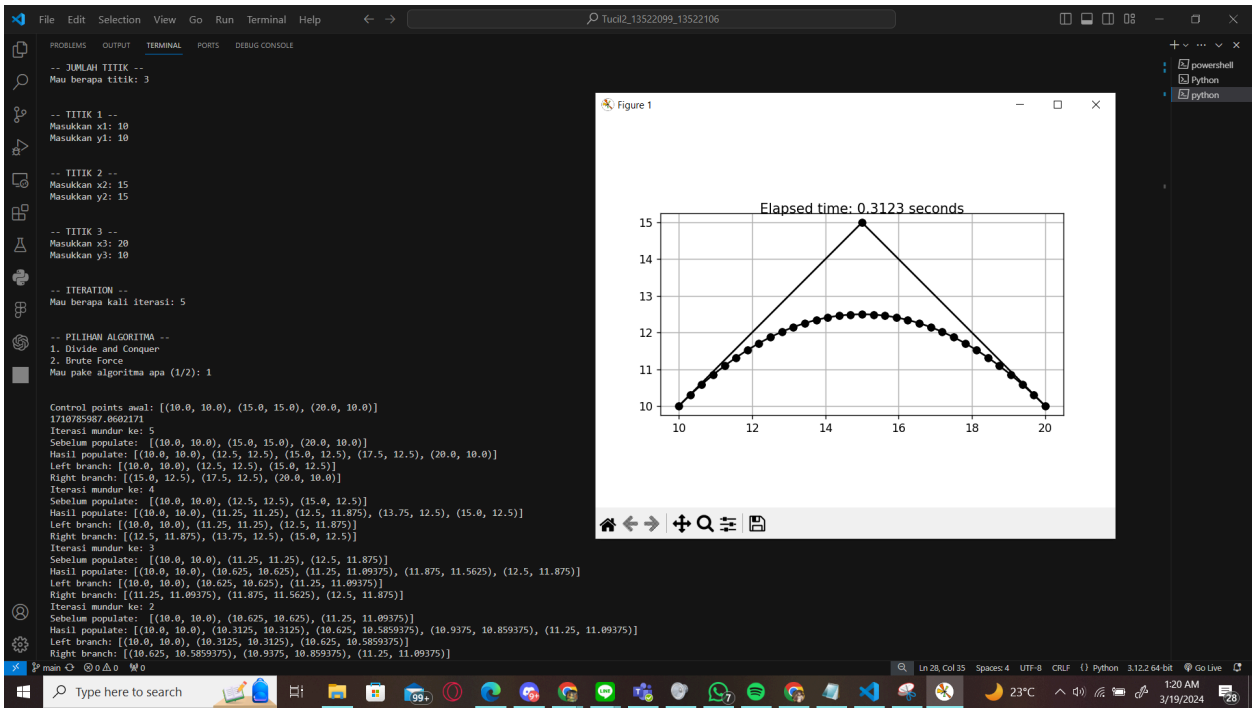


Divide and Conquer

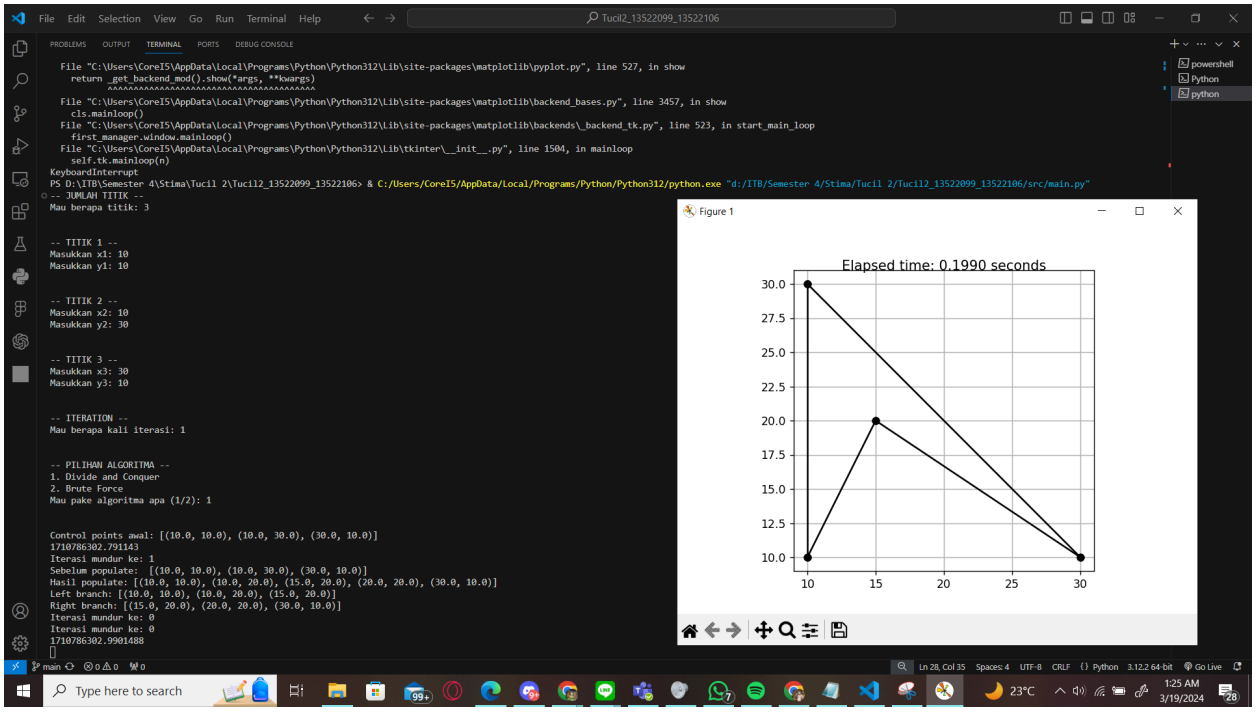
3 titik, (10, 10), (15, 15), (20, 10), 1 iterasi



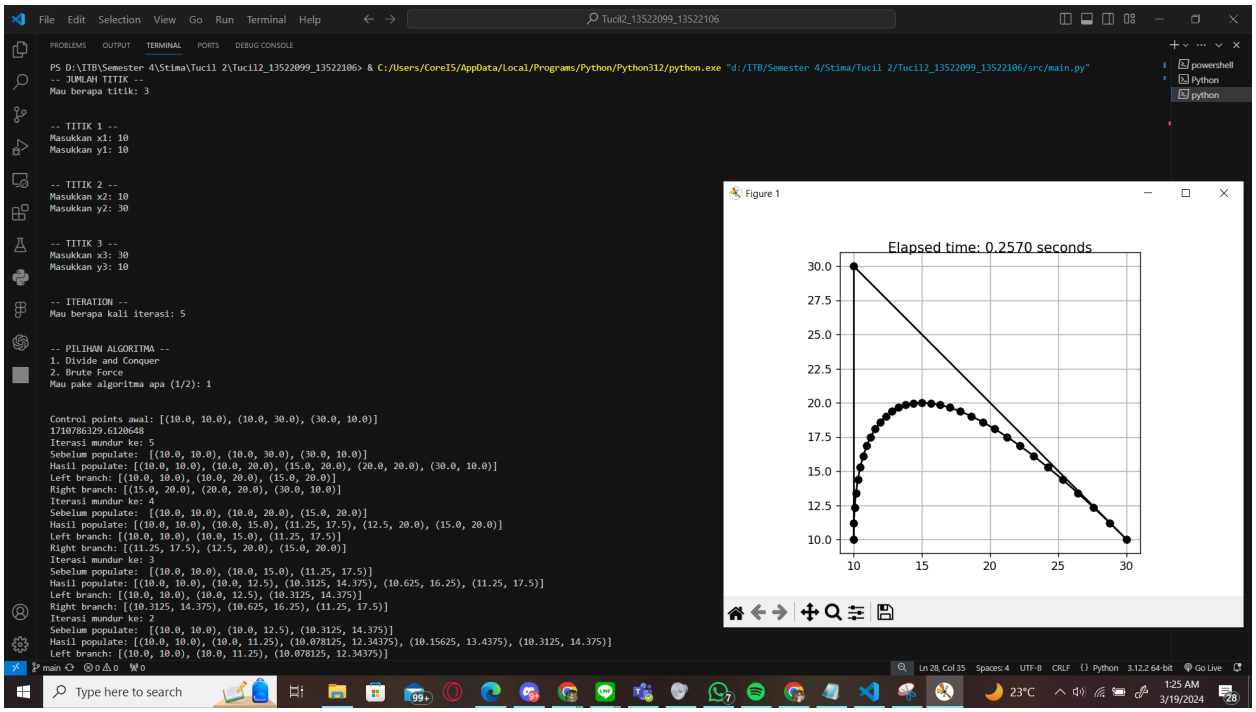

3 titik, (10, 10), (15, 15), (20, 10), 5 iterasi



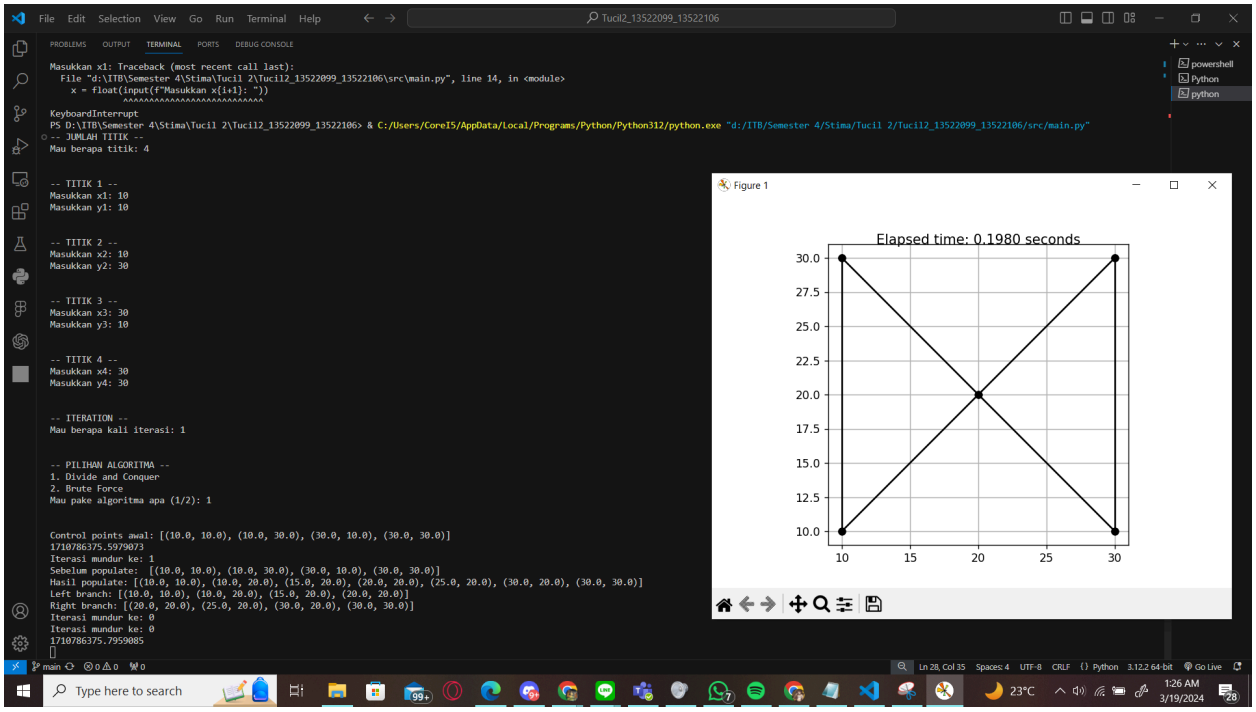
3 titik, (10, 10), (10, 30), (30, 10), 1 iterasi



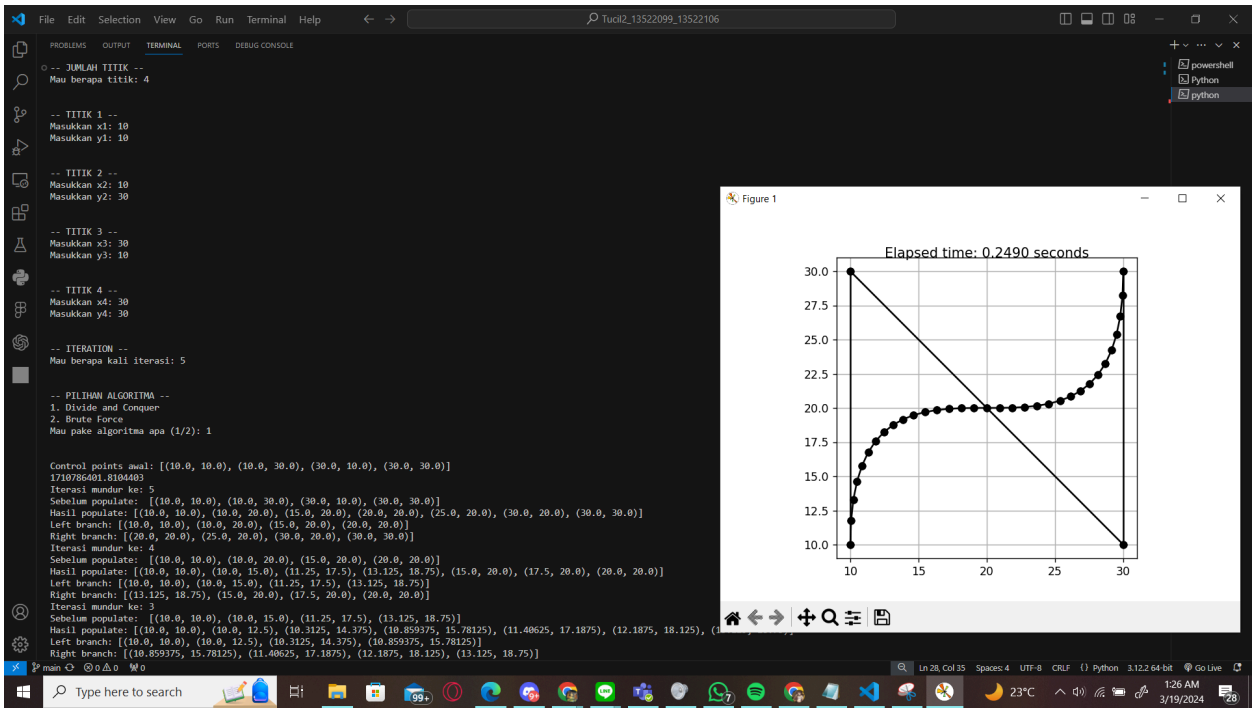

3 titik, (10, 10), (10, 30), (30, 10), 5 iterasi



4 titik, (10, 10), (10, 30), (30, 10), (30, 30), 1 iterasi




4 titik, (10, 10), (10, 30), (30, 10), (30, 30), 5 iterasi



## ANALISIS PERBANDINGAN

Untuk Algoritma Brute Force, akan dilakukan perhitungan kompleksitas waktu dalam notasi T(n) dan O(n) sebagai berikut.

- Fungsi draw_line hanya membuat garis antara 2 titik, sehingga T(1) dan O(1)
- Fungsi draw_bezier_curve_bf terdiri atas dua pengulangan utama, di mana pengulangan kedua terdapat di dalam pengulangan pertama. Dengan n sebagai jumlah titik antara dan m sebagai jumlah titik kontrol, maka dapat kita rumuskan kompleksitas waktu untuk masing-masing pengulangan dan fungsi keseluruhan. Pengulangan pertama melakukan iterasi sebanyak jumlah titik antara ditambah dua, sehingga kompleksitas waktunya adalah T(n + 2) dan O(n). Sementara itu, pengulangan kedua melakukan iterasi sebanyak jumlah titik kontrol, sehingga kompleksitas waktunya adalah T(n) dan O(n). Secara keseluruhan, fungsi memiliki kompleksitas waktu T(nm + 2m) dan O(nm).

Untuk Divide and Conquer, akan dilakukan perhitungan Time Complexity dan Big O Notation dari tiap fungsi sebagai berikut:

- Fungsi DrawLine hanya membuat garis antara 2 titik, sehingga T(1) dan O(1)
- Fungsi MakeNewPoint juga hanya mencari titik tengah, sehingga kompleksitasnya hanya T(6) dari 4 perkalian dan 2 penjumlahan, dan O(1) setelah disederhanakan.
- Fungsi PopulatePoints merupakan fungsi rekursif (T(n-1)) yang terdapat looping untuk setiap titik pada array dikurangi 1 (T(n-1)), dan proses lainnya (c) sehingga total = T(n-1) + T(n-1) + c = 2(T(n-1)) + c, sehingga Big O Notationnya O(2^n). Sebenarnya bisa disederhanakan menjadi O(1) saja jika fungsi digunakan untuk beberapa titik tertentu, misal 3 titik, hanya memanggil MakeNewPoint sebanyak 3 kali daripada fungsi rekursif.
- Fungsi DrawBezierCurve merupakan gabungan dari beragam operasi, di mana untuk n titik, memanggil fungsi Populate Points yaitu 2(T(n-1)) + c, dan dibagi menjadi 2 melalui divide yaitu 2T(((2n-1)+1)/2) = 2T(n) dengan 2n-1 adalah peningkatan jumlah titik diakibatkan PopulatePoints dan +1 karena titik tengah dipakai di kedua Divide, dan pengurangan iteration serta penggabungan array yang

kita misalkan sebagai d, sehingga total = 2(T(n-1)) + c + 2T(n) + d dan kompleksitasnya menjadi O(2^n).

- Fungsi BezierMain sebenarnya hanyalah memanggil DrawBezierCurve dengan beberapa tambahan operasi awal, sehingga kompleksitas akhirnya sama, yaitu O(2^n).

Berdasarkan perhitungan Time Complexity (T(n)) dan Big O Notation (O(n)) di atas, karena terdapat perbedaan pada elemen yang diiterasi, di mana pendekatan *Brute Force* melakukan iterasi berdasarkan jumlah titik antara, sedangkan pendekatan *Divide and Conquer* melakukan iterasi berdasarkan berapa kali membuat garis kontrol baru, maka kompleksitas waktu dari kedua pendekatan tidak dapat dibandingkan pada elemen/tarik ukur yang sama. Hasil yang kami dapat ialah pendekatan *Brute Force* memiliki kompleksitas waktu T(nm + 2m) dan O(nm) di mana n adalah jumlah titik kontrol dan m adalah jumlah titik antara, sedangkan pendekatan *Divide and Conquer* memiiki kompleksitas waktu 2(T(n - 1)) + c + 2T(n) + d dan $O(2^n)$ di mana n adalah iterasi, c adalah proses lainnya, dan d adalah pengurangan iteration serta penggabungan array.

## LAMPIRAN

| Poin | Ya | Tidak |
|---|---|---|
| 1. Program berhasil dijalankan | V | |
| 2. Program dapat melakukan visualisasi kurva Bezier | V | |
| 3. Solusi yang diberikan program optimal | V | |
| 4. **[Bonus]** Program dapat membuat kurva untuk *n* titik kontrol. | V | |
| 5. **[Bonus]** Program dapat melakukan visualisasi proses pembuatan kurva | V | |

nb. Bonus 5 dilakukan pada DnC dengan melakukan looping untuk setiap iterasi dari 1 sampai input

**PRANALA**

https://github.com/Julian-Caleb/Tucil2_13522099_13522106/