LAPORAN TUGAS KECIL 2

IF2211 - STRATEGI ALGORITMA

“Membangun Kurva Bézier dengan Algoritma Titik Tengah berbasis Divide and Conquer”

Dosen:

Ir. Rila Mandala, M.Eng., Ph.D.

Monterico Adrian, S.T., M.T.

Kelompok 74:

Farhan Raditya Aji (13522142)

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER II TAHUN 2023/2024

DAFTAR ISI

BAB I

Algoritma *Divide and Conquer*

Divide and conquer adalah sebuah strategi algoritma yang digunakan dalam pemrograman komputer dan matematika untuk memecahkan masalah yang kompleks menjadi bagian-bagian yang lebih kecil, lebih mudah diselesaikan, dan kemudian menggabungkan solusi-solusi tersebut menjadi solusi untuk masalah aslinya. Strategi ini didasarkan pada prinsip "pecah masalah menjadi bagian yang lebih kecil".

Konsep divide and conquer terdiri dari tiga langkah utama:

1. Divide (Pecah): Masalah awal dibagi menjadi beberapa submasalah yang lebih kecil dan independen. Pemecahan masalahnya dilakukan dengan cara membagi masalah tersebut menjadi dua atau lebih submasalah yang memiliki ukuran yang lebih kecil. Proses pemecahan ini dilakukan berulang kali hingga masalah yang lebih kecil bisa diselesaikan secara langsung.
2. Conquer (Kalahkan): Setelah masalah dibagi menjadi submasalah yang lebih kecil, setiap submasalah dipecahkan secara terpisah. Biasanya, submasalah diselesaikan menggunakan teknik rekursif atau dengan menerapkan strategi divide and conquer secara berulang pada submasalah tersebut hingga memungkinkan untuk mencapai solusi secara langsung.
3. Combine (Gabungkan): Setelah semua submasalah diselesaikan, solusi dari setiap submasalah digabungkan kembali menjadi solusi untuk masalah aslinya. Proses penggabungan ini dilakukan secara sistematis untuk memastikan bahwa solusi yang diberikan memenuhi kriteria masalah asli.

Penerapan strategi *divide and conquer* sering kali menghasilkan algoritma yang efisien untuk memecahkan masalah yang kompleks, seperti pengurutan data, pencarian data, pemecahan masalah geometri, dan banyak lagi. Contoh terkenal dari algoritma yang menggunakan pendekatan divide and conquer termasuk algoritma pengurutan cepat (quick sort) dan algoritma pencarian biner (binary search).

Keunggulan dari pendekatan divide and conquer termasuk kemampuannya untuk memecahkan masalah dengan kompleksitas waktu yang lebih rendah daripada pendekatan brute force (kekerasan), terutama ketika masalah tersebut dapat dibagi secara efisien menjadi submasalah yang lebih kecil. Namun, perlu diingat bahwa tidak semua masalah cocok untuk pendekatan ini, dan dalam beberapa kasus, biaya tambahan dari membagi dan menggabungkan solusi dapat mengurangi efisiensi algoritma. Oleh karena itu, analisis cermat mengenai struktur masalah diperlukan sebelum menerapkan strategi divide and conquer.

BAB II

Analisis Algoritma *Brute Force* sebagai Algoritma Pembanding

Algoritma *brute force* dalam konteks pembuatan kurva Bézier adalah pendekatan langsung yang akan menghitung titik pada kurva secara sekuensial sebanyak iterasi yang dimasukkan user. Berbeda dengan *divide and conquer* yang melalukan dengan membagi dua rekursif pada fungsinya sehingga memecah-mecah menjadi beberapa bagian. Sehingga algoritma ini akan menjadi lebih lama dibandingkan dengan *divide and conquer* karena langsung menyelesaikan dalam satu kesatuan utuh.

Berikut adalah langkah-langkah implementasi algoritma *brute force* pada pembuatan kurva Bézier:

1. Inisialisasi Variabel dan Perhitungan Titik Tengah Awal
    a. Pertama, inisialisasi dengan menyimpan titik-titik tengah antara titik kontrol pada variabel midPoints.

2. Iterasi untuk Membangun Kurva Bezier
    a. Dilakukan iterasi sebanyak iterasi - 1.
    b. Pada setiap iterasi, dimulai dengan menghitung titik tengah antara titik tengah sebelumnya yang disimpan pada variabel midPoints yang akan disimpan pada variabel tempKurvaPoints.
    c. Tambahkan juga titik kontrol pertama pada index pertama dan titik kontrol terakhir pada index terakhir pada tempKurvaPoints.
    d. Setelah itu simpan ke dalam variabel yang akan menyimpan hasil nya pada variabel kurvaPoints.

3. Pembaruan Titik-titik Tengah
    a. Inisialisasi Variabel tempMidPoints dibuat untuk menyimpan titik-titik tengah baru.
    b. Perhitungan titik-titik tengah baru dilakukan iterasi selama masih terdapat titik-titik tengah yang belum diproses.
    c. Pada setiap iterasi, titik tengah baru dihitung dengan menggunakan titik-titik hasil iterasi terakhir dan titik-titik tengah sebelumnya.
    d. Setelah itu titik baru akan disimpan pada variabel tempMidPoints

e. Setelah semua titik tengah baru dihitung dan diproses, list tempMidPoints digunakan untuk memperbarui variabel midPoints hal ini selama proses iterasi berlangsung nilainya tidak terpengaruh

4. Update Titik-Titik Kurva Bezier Setelah Iterasi
    a. Simpan kembali hasil perhitungan titik-titik tengah antar titik yang sudah dihitung pada iterasi (variabel midPoints).
    b. Setelah itu,tambahkan juga titik kontrol pertama pada index pertama dan titik kontrol terakhir.
    c. Terakhir, simpan hasil kurva Bezier yang lengkap ke dalam variabel kurvaPoints.

‘

BAB III

Analisis dan Implementasi Algoritma *Divide and Conquer*

Algoritma *Divide and Conquer* (DNC) dalam pembuatan kurva Bézier adalah pendekatan yang memecah masalah menjadi bagian-bagian yang lebih kecil untuk kemudian menyelesaikannya secara rekursif. Proses dimulai dengan pembagian kurva Bézier menjadi submasalah yang lebih sederhana. Setiap submasalah terdiri dari tiga titik kontrol, yaitu titik awal (P0), titik tengah (P1), dan titik akhir (P2) dari kurva Bézier.

Kemudian, algoritma mencari titik tengah antara dua titik kontrol untuk membagi kurva Bézier menjadi dua bagian. Proses ini dilakukan dengan menggunakan formula sederhana untuk menemukan midpoint dalam ruang dua dimensi. Setelah midpoint ditemukan, algoritma melakukan pemecahan rekursif pada dua submasalah yang dihasilkan dari pembagian sebelumnya. Proses rekursi berlanjut hingga mencapai kondisi dasar, yaitu ketika iterasi mencapai nilai minimum yang telah ditentukan.

Selama proses rekursi, setiap kali algoritma menemukan midpoint baru, titik tersebut ditambahkan ke dalam daftar `kurvaPoints`. Daftar ini berfungsi sebagai wadah untuk mengumpulkan semua midpoint yang ditemukan selama proses pembagian dan rekursi. Setelah semua iterasi selesai dieksekusi, algoritma menggabungkan semua midpoint yang ditemukan untuk membentuk kurva Bézier akhir.

Hasil akhir kurva Bézier adalah gabungan dari semua titik midpoint yang telah dikumpulkan. Dengan demikian, algoritma DNC memungkinkan penanganan efisien masalah pembuatan kurva Bézier dengan memanfaatkan pembagian dan penyelesaian secara rekursif. Pendekatan ini memungkinkan representasi yang akurat dari kurva Bézier dengan tingkat kompleksitas yang dapat diatur sesuai dengan jumlah iterasi yang dipilih.

Berikut adalah langkah-langkah implementasi algoritma *divide and conquer* pada pembuatan kurva Bézier:

1. Pembagian Masalah:
    a. Algoritma DNC memulai dengan membagi masalah besar, yaitu pembuatan kurva Bézier, menjadi submasalah yang lebih kecil.

   b. Setiap submasalah terdiri dari tiga titik kontrol, yaitu titik awal (P0), titik tengah (P1), dan titik akhir (P2) dari kurva Bézier.
   c. Submasalah ini dipecah menjadi dua submasalah lebih kecil dengan menemukan midpoint pada garis yang menghubungkan titik-titik kontrol.
2. Penyelesaian Submasalah:
   a. Setelah pembagian, algoritma mencari midpoint antara titik kontrol untuk membagi kurva Bézier menjadi dua bagian.
   b. Pencarian midpoint ini dilakukan dengan menggunakan formula sederhana untuk menemukan titik tengah antara dua titik dalam ruang dua dimensi.
3. Rekursi:
   a. Setelah pencarian midpoint, algoritma memanggil dirinya sendiri secara rekursif untuk menyelesaikan setiap submasalah yang lebih kecil.
   b. Rekursi dilakukan pada dua submasalah yang dihasilkan dari pembagian sebelumnya.
   c. Proses rekursi berlanjut hingga mencapai kondisi dasar, dimana iterasi berhenti.
4. Pengumpulan Solusi:
   a. Setiap kali algoritma menemukan midpoint baru, titik tersebut ditambahkan ke dalam variabel kurvaPoints.
   b. Variabel ini akan berisi semua midpoint yang ditemukan selama proses pembagian dan rekursi.
5. Gabungan Solusi:
   a. Setelah semua iterasi selesai dieksekusi, algoritma menggabungkan semua midpoint yang ditemukan untuk membentuk kurva Bézier akhir.
   b. Kurva Bézier akhir ini akan terdiri dari serangkaian titik yang merepresentasikan jalur kurva sesuai dengan tingkat kompleksitas iterasi yang dipilih.

BAB IV

Source Code

Source code diimplementasikan menggunakan bahasa *Python* , berikut adalah source code-nya :

1. Brute_Force.py

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

class KurvaBezier_Brute_Force:
    def __init__(self,points):
        self.kurvaPoints = []
        self.midPoints = []
        self.points = points
        self.eksekusi = 0

    def getTime(self):
        return self.eksekusi

    def count_midpoint(self, p0, p1):
        return ((p0[0] + p1[0]) / 2, (p0[1] + p1[1]) / 2)

    def Brute_Force(self,points,iterasi):
        start = time.time()
        self.midPoints = [self.count_midpoint(points[i], points[i+1]) for i in
range(len(points) - 1)]
        for _ in range(iterasi - 1):
            tempKurvaPoints = [self.count_midpoint(self.midPoints[j],
self.midPoints[j+1]) for j in range(len(self.midPoints) - 1)]
            tempKurvaPoints.insert(0, points[0])
            tempKurvaPoints.append(points[-1])
            self.kurvaPoints = tempKurvaPoints[:]
            tempMidPoints = []
            while self.midPoints:
                tempMidPoints.append(self.count_midpoint(tempKurvaPoints[0],
self.midPoints[0]))
                if len(tempMidPoints) % 2 == 0:
                    self.midPoints.pop(0)
                else:
                    tempKurvaPoints.pop(0)
```

```
            self.midPoints = tempMidPoints[:]
        tempKurvaPoints = [self.count_midpoint(self.midPoints[j],
self.midPoints[j+1]) for j in range(len(self.midPoints) - 1)]
        tempKurvaPoints.insert(0, points[0])
        tempKurvaPoints.append(points[-1])
        self.kurvaPoints = tempKurvaPoints[:]
        self.eksekusi = time.time() - start
        self.draw_kurva()

    def draw_kurva(self):
        plt.scatter([p[0] for p in self.points], [p[1] for p in self.points], color='blue',
label='Control Points')
        plt.plot([p[0] for p in self.points], [p[1] for p in self.points], 'b--')
        poin = np.array(self.kurvaPoints)
        plt.plot(poin[:, 0], poin[:, 1], 'r-', label='Bezier Curve')
        plt.grid(True)
```

2. Divide_n_Conquer.py

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

class KurvaBezier_DNC:
    def __init__(self,points):
        self.points = points
        self.kurvaPoints = []
        self.eksekusi = 0

    def count_midpoint(self, p0, p1):
        return ((p0[0] + p1[0]) / 2, (p0[1] + p1[1]) / 2)

    def getTime(self):
        return self.eksekusi

    def startDNC(self, points, iterasi):
        self.kurvaPoints = []
        self.kurvaPoints.append(points[0])
        self.DNC(points, iterasi)
        self.kurvaPoints.append(points[2])

    def DNC(self,points,iterasi):
        start_time = time.time()  # Waktu mulai iterasi
        if iterasi == 0:
            return
        else:
            left_mid = self.count_midpoint(points[0], points[1])
```

```
            right_mid = self.count_midpoint(points[1], points[2])
            hasil_mid = self.count_midpoint(left_mid, right_mid)

            self.DNC([points[0], left_mid, hasil_mid], iterasi-1)
            self.kurvaPoints.append(hasil_mid)
            self.DNC([hasil_mid, right_mid, points[2]], iterasi-1)
        self.eksekusi += time.time() - start_time  # Waktu eksekusi iterasi

    def draw_kurva(self):
        poin = np.array(self.kurvaPoints)
        plt.plot(poin[:, 0], poin[:, 1], 'r-', label='Bezier Curve')

    def animasi(self, frame):
        plt.cla()
        plt.scatter([p[0] for p in self.points], [p[1] for p in self.points], color='blue',
label='Control Points')
        plt.plot([p[0] for p in self.points], [p[1] for p in self.points], 'b--')
        self.startDNC(self.points, frame)
        if frame != 0:
            self.draw_kurva()
        plt.title(f"Kurva Bezier iterasi ke-{frame}")
        plt.xlabel("X")
        plt.ylabel("Y")
        plt.legend()
        plt.grid(True)
        plt.axis('equal')
```

3. main.py

```
from tkinter import *
from Divide_n_Conquer import *
from Brute_Force import *
from matplotlib.animation import FuncAnimation

class App:
    def __init__(self, master):
        self.master = master
        master.title("Kurva Bezier")

        self.label = Label(master, text="Masukkan 3 titik koordinat dalam bentuk tuple
seperti (0, 0)")
        self.label.pack()

        self.label = Label(master, text="Titik 1")
        self.label.pack()
        self.p0_entry = Entry(master)
        self.p0_entry.pack()
```

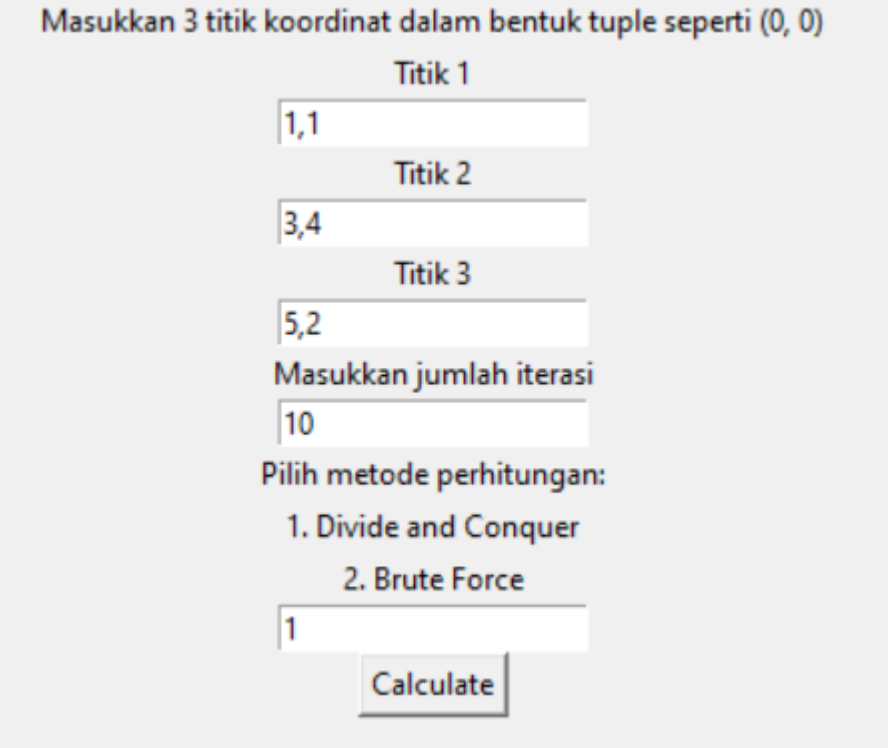
```
        self.label = Label(master, text="Titik 2")
        self.label.pack()
        self.p1_entry = Entry(master)
        self.p1_entry.pack()

        self.label = Label(master, text="Titik 3")
        self.label.pack()
        self.p2_entry = Entry(master)
        self.p2_entry.pack()

        self.label = Label(master, text="Masukkan jumlah iterasi")
        self.label.pack()
        self.iterasi_entry = Entry(master)
        self.iterasi_entry.pack()

        self.label = Label(master, text="Pilih metode perhitungan:")
        self.label.pack()
        self.label = Label(master, text="1. Divide and Conquer")
        self.label.pack()
        self.label = Label(master, text="2. Brute Force")
        self.label.pack()

        self.choice_entry = Entry(master)
        self.choice_entry.pack()

        self.calculate_button = Button(master, text="Calculate",
command=self.calculate)
        self.calculate_button.pack()

    def calculate(self):
        p0 = eval(self.p0_entry.get())
        p1 = eval(self.p1_entry.get())
        p2 = eval(self.p2_entry.get())
        iterasi = int(self.iterasi_entry.get())
        choice = int(self.choice_entry.get())

        if choice == 1 :
            self.DivideNConquer([p0, p1, p2], iterasi)
        else:
            self.Bruteforce([p0, p1, p2], iterasi)

    def Bruteforce(self, points, iterasi):
        print("------------------------Brute Force-----------------------------")

print("==========================================================
===========")
        print("Berikut adalah hasil dari kurva bezier dengan iterasi sebanyak", iterasi)
        kurva_bf = KurvaBezier_Brute_Force(points)
        kurva_bf.Brute_Force(points, iterasi)
```

```
        print("Waktu eksekusi : ", kurva_bf.getTime() , "detik")
        plt.show()

print("============================================================
============")

    def DivideNConquer(self, points, iterasi):
        print("----------------------Divide and Conquer-------------------------")

print("============================================================
============")
        print("Berikut adalah hasil dari kurva bezier dengan iterasi sebanyak", iterasi)
        kurva_dnc = KurvaBezier_DNC(points)
        animasi = FuncAnimation(plt.gcf(), kurva_dnc.animasi,
frames=iterasi+1,repeat=False)
        plt.show()
        print("Waktu eksekusi : ", kurva_dnc.getTime() , "detik")

print("============================================================
============")

root = Tk()
app = App(root)
root.mainloop()
```

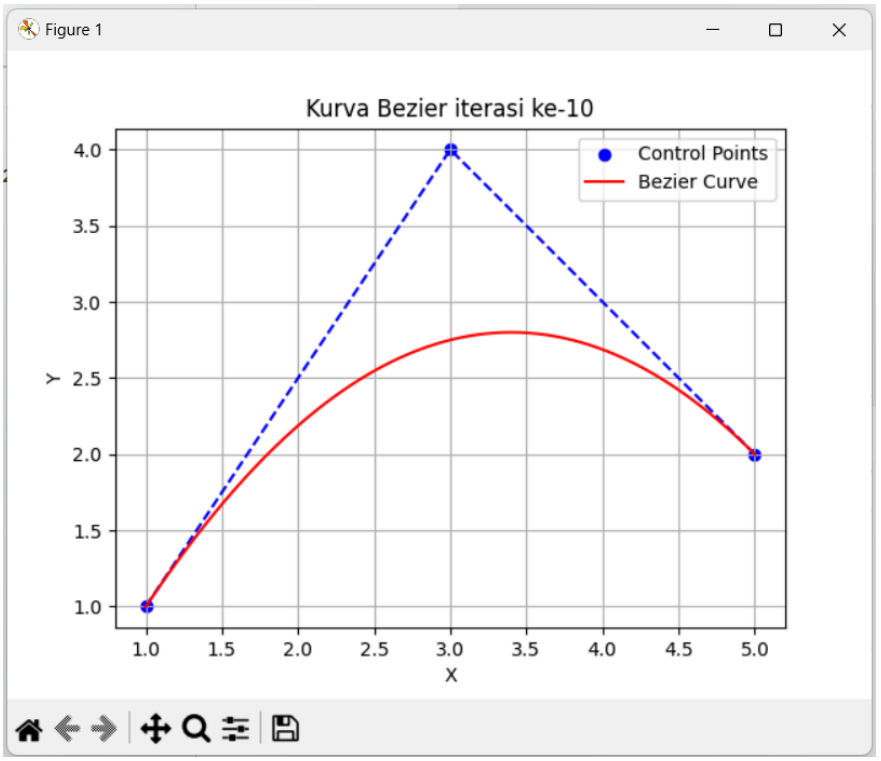
BAB V

Input dan *Output*

1. DNC
    a. Input:

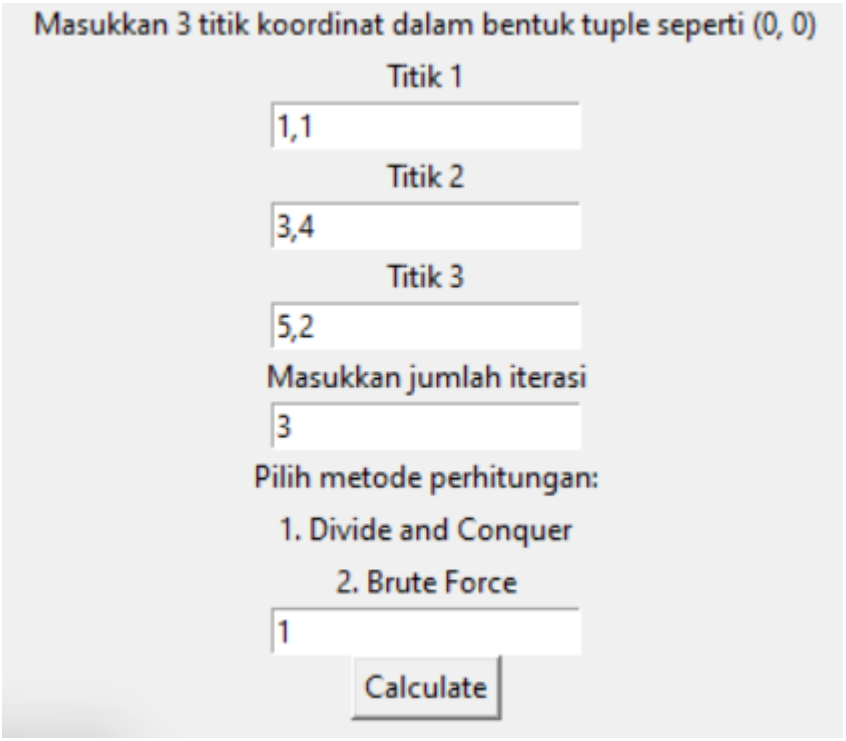


Output:




```
----------------------Divide and Conquer--------------------------
======================================================================
Berikut adalah hasil dari kurva bezier dengan iterasi sebanyak 10
Waktu eksekusi :  0.009999275207519531 detik
======================================================================
```

b. Input:


Masukkan 3 titik koordinat dalam bentuk tuple seperti (0, 0)
Titik 1
1,1
Titik 2
3,4
Titik 3
5,2
Masukkan jumlah iterasi
3
Pilih metode perhitungan:
1. Divide and Conquer
2. Brute Force
1
Calculate

output:

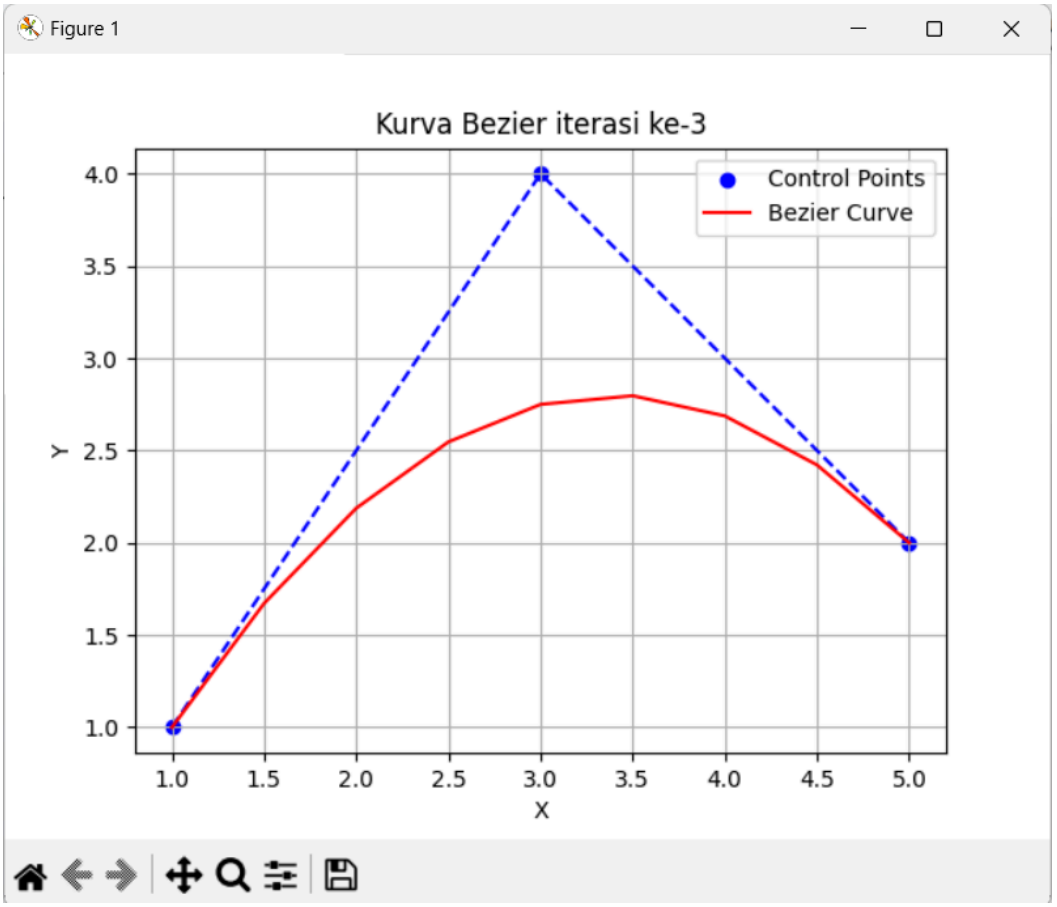
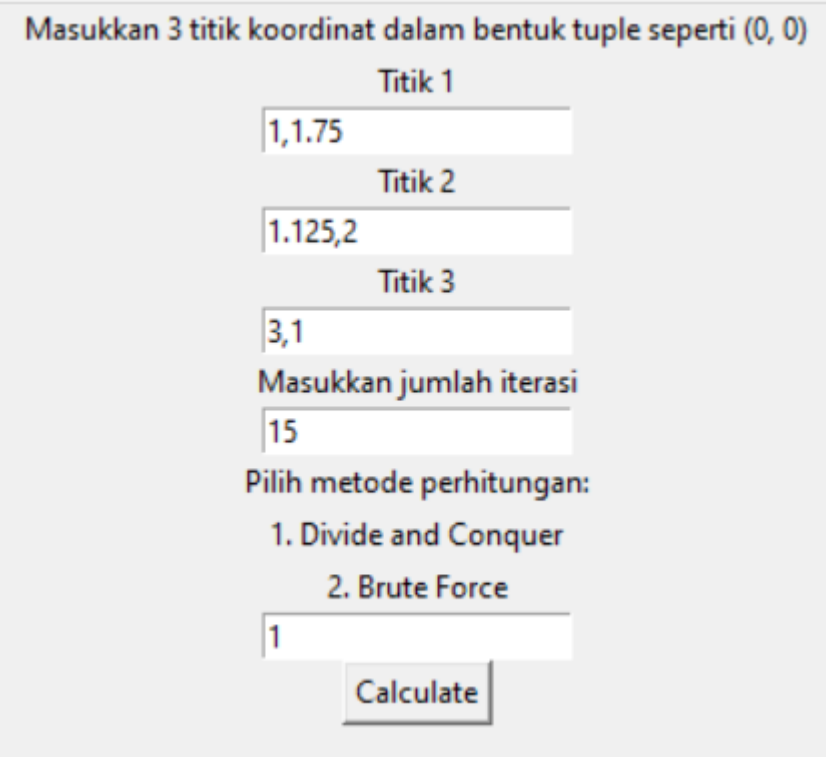


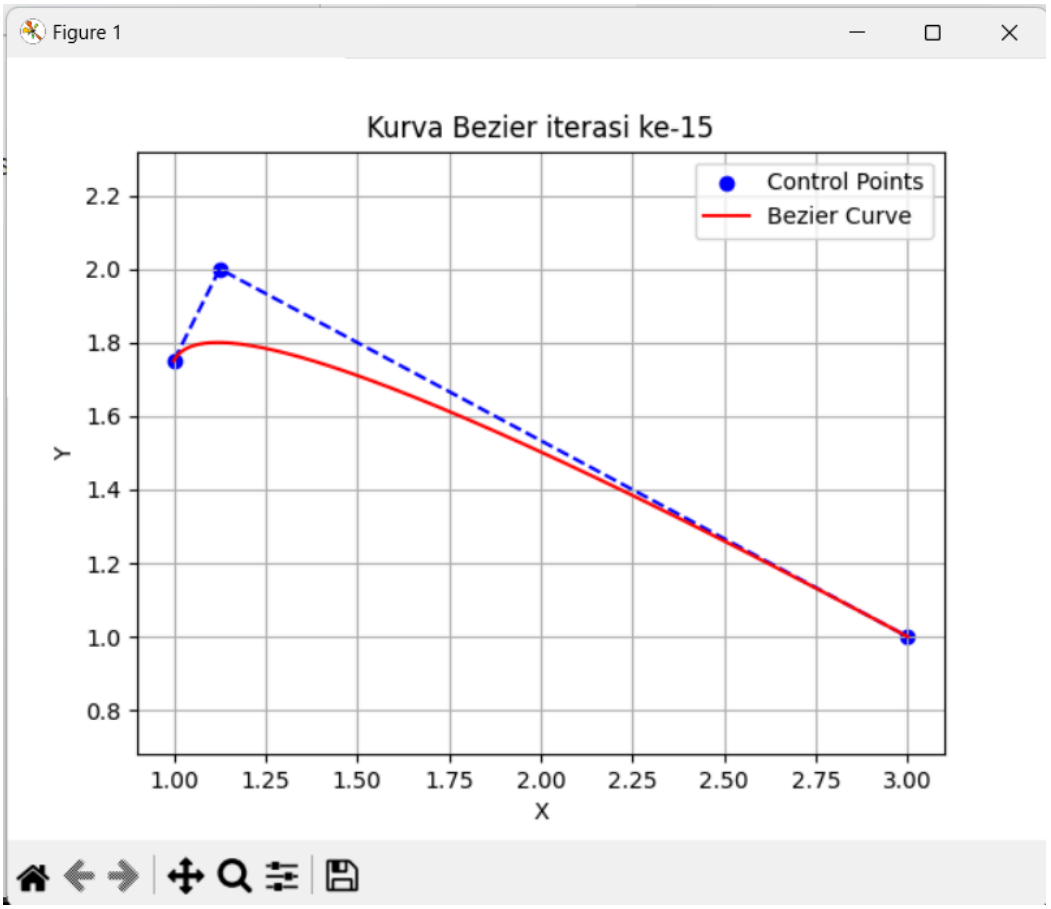
Figure 1
Kurva Bezier iterasi ke-3
Control Points
Bezier Curve
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Y
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
X

```
----------------------Divide and Conquer-------------------------
======================================================================
Berikut adalah hasil dari kurva bezier dengan iterasi sebanyak 3
Waktu eksekusi :  0.0 detik
======================================================================
```

c. Input:

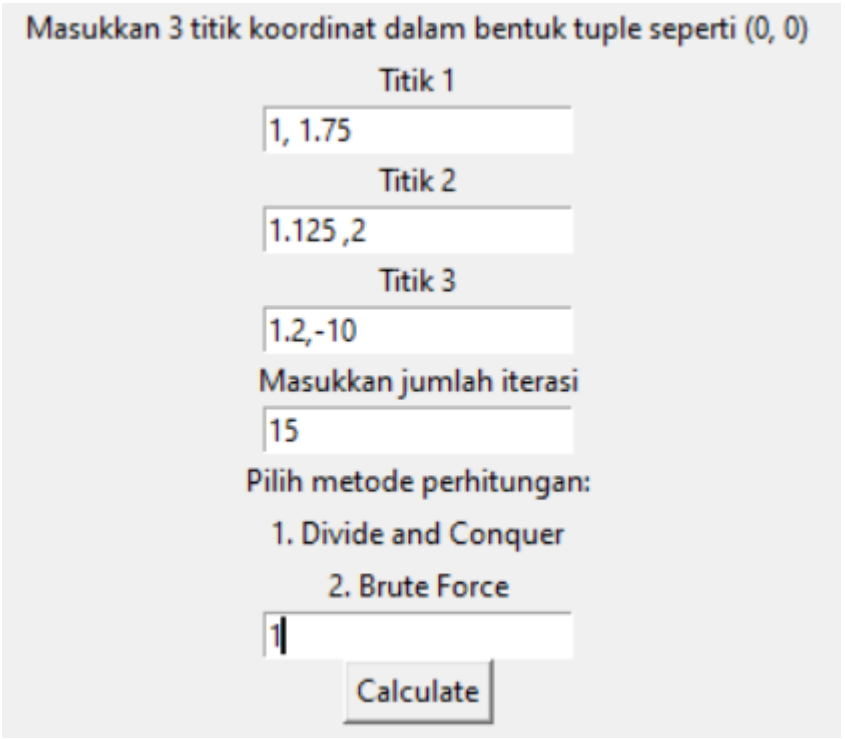


output:

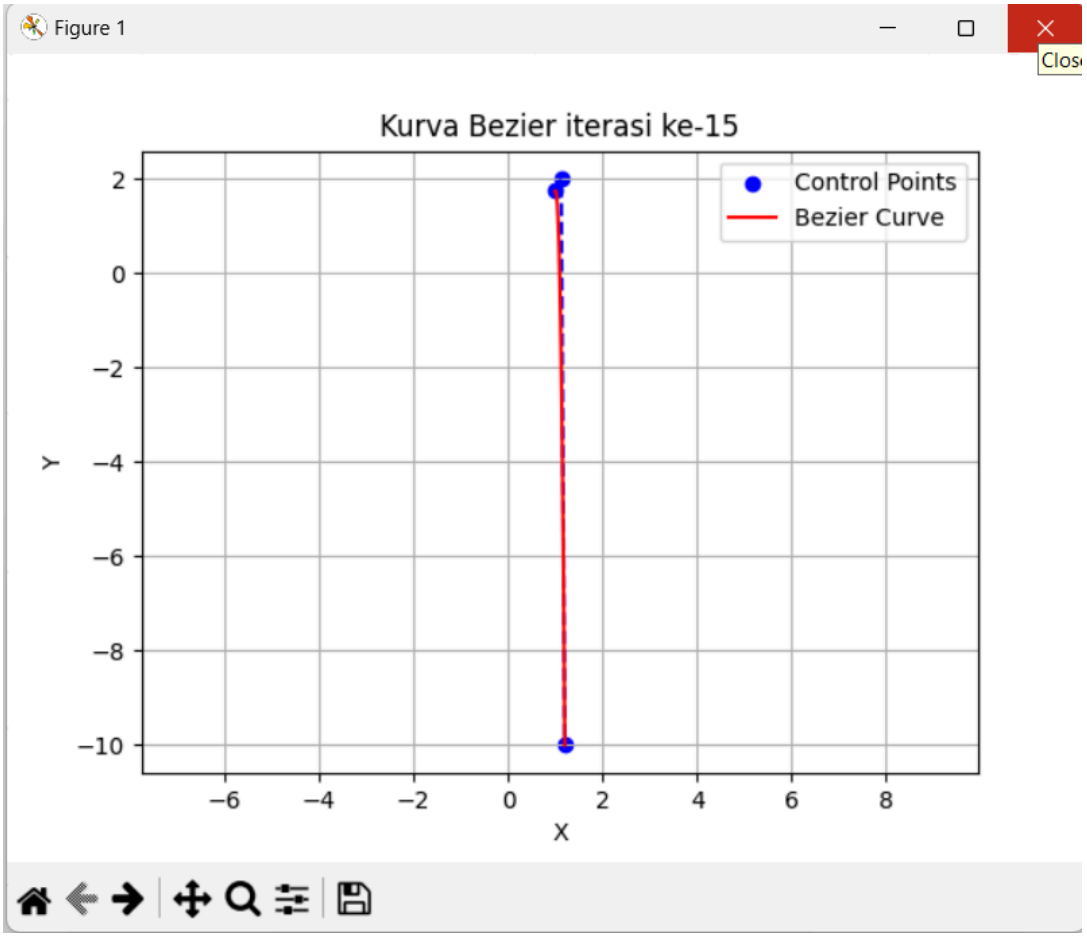



```
----------------------Divide and Conquer--------------------------
======================================================================
Berikut adalah hasil dari kurva bezier dengan iterasi sebanyak 15
Waktu eksekusi :  0.7262659072875977 detik
======================================================================
```

d. Input:

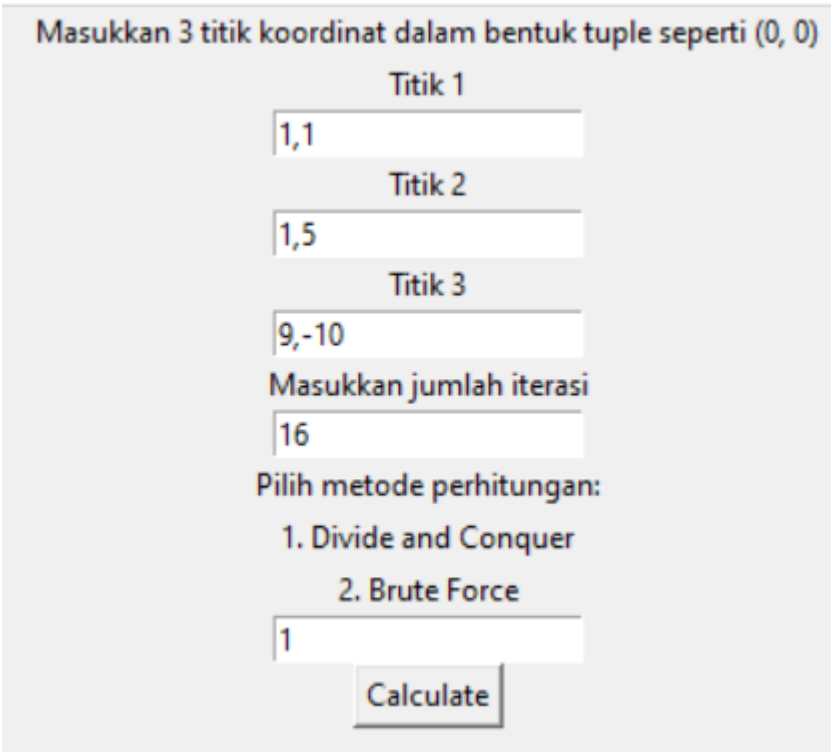

Masukkan 3 titik koordinat dalam bentuk tuple seperti (0, 0)
Titik 1
1, 1.75
Titik 2
1.125 ,2
Titik 3
1.2,-10
Masukkan jumlah iterasi
15
Pilih metode perhitungan:
1. Divide and Conquer
2. Brute Force
1
Calculate

output:




```
----------------------Divide and Conquer--------------------------
======================================================================
Berikut adalah hasil dari kurva bezier dengan iterasi sebanyak 15
Waktu eksekusi :  0.7330248355865479 detik
```

e. Input:


Masukkan 3 titik koordinat dalam bentuk tuple seperti (0, 0)
Titik 1
1,1
Titik 2
1,5
Titik 3
9,-10
Masukkan jumlah iterasi
16
Pilih metode perhitungan:
1. Divide and Conquer
2. Brute Force
1
Calculate

output:

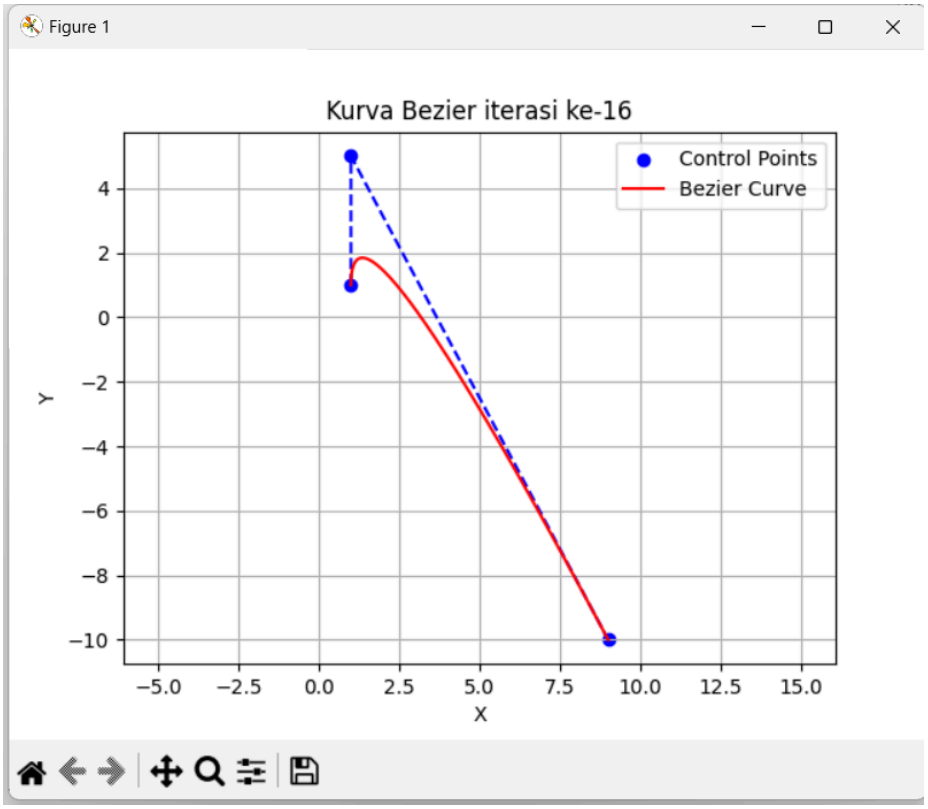
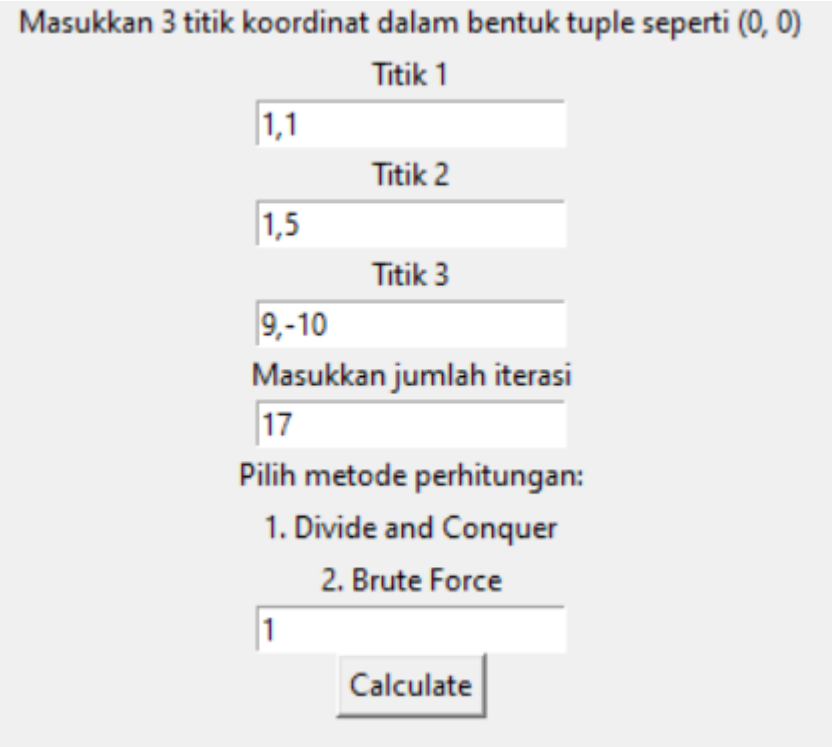


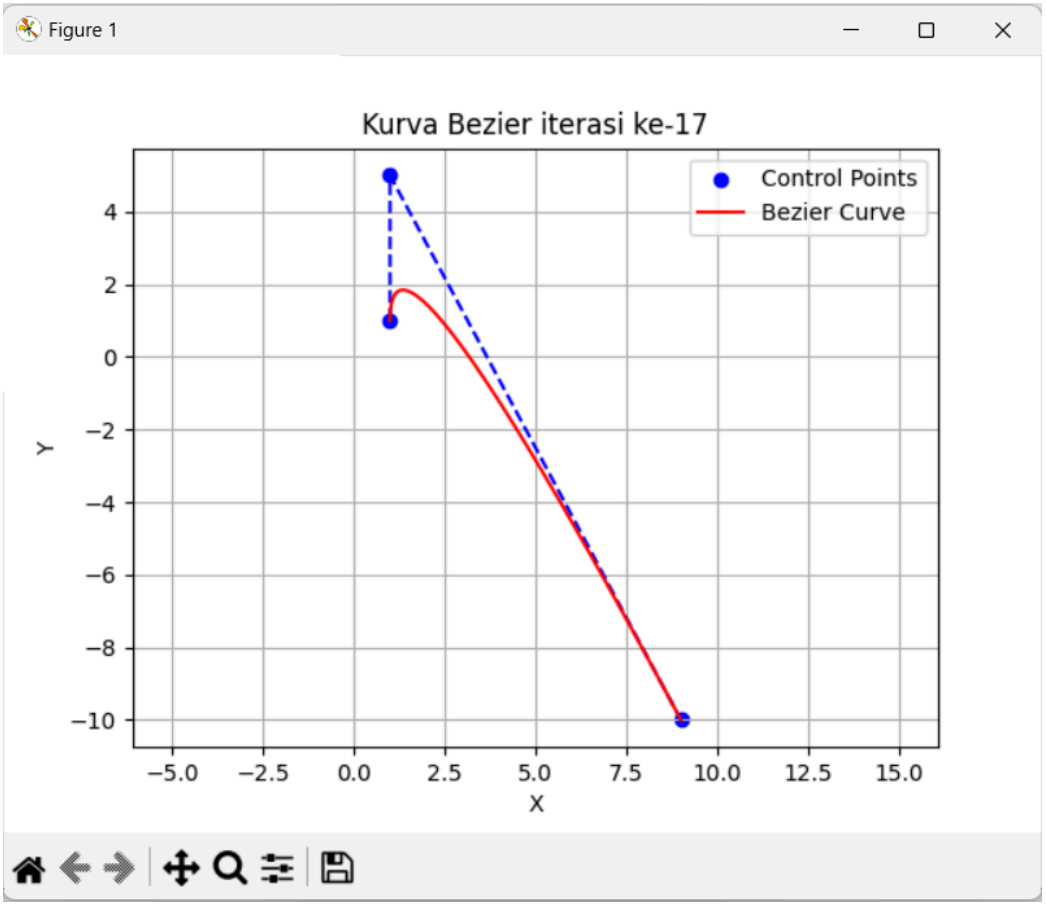
Figure 1
Kurva Bezier iterasi ke-16
Control Points
Bezier Curve
Y
X
4
2
0
−2
−4
−6
−8
−10
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0

```
----------------------Divide and Conquer-------------------------
======================================================================
Berikut adalah hasil dari kurva bezier dengan iterasi sebanyak 16
Waktu eksekusi :  1.560340166091919 detik
======================================================================
```

f. Input:

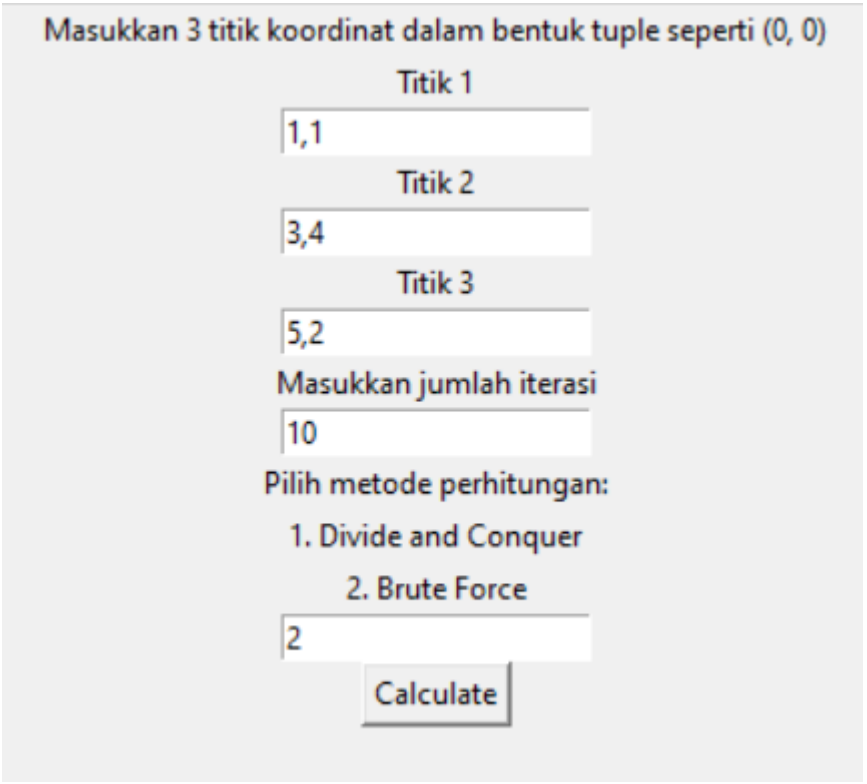


output:

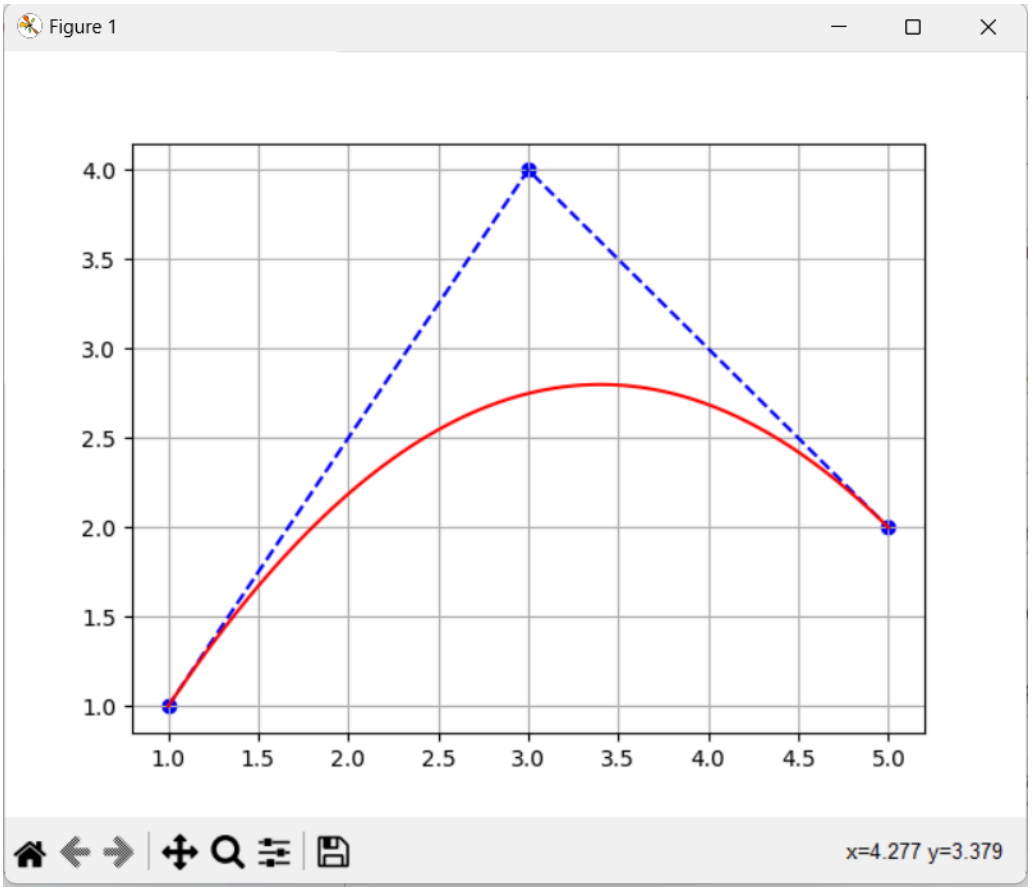



```
----------------------Divide and Conquer-------------------------
======================================================================
Berikut adalah hasil dari kurva bezier dengan iterasi sebanyak 17
Waktu eksekusi :  3.4702775478363037 detik
======================================================================
```

2. *Brute force*
    a. Input:

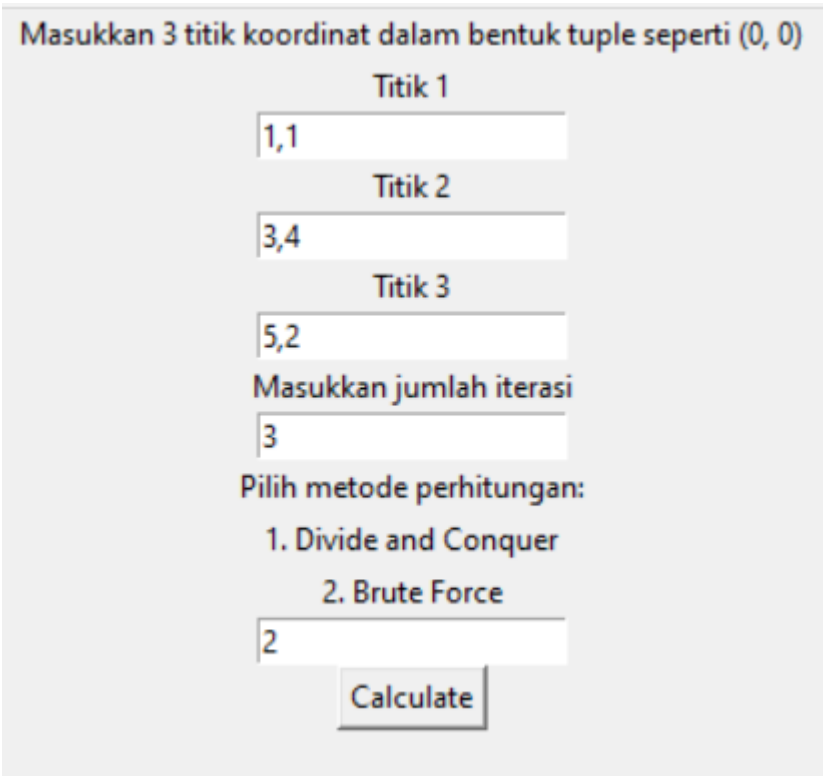

Masukkan 3 titik koordinat dalam bentuk tuple seperti (0, 0)
Titik 1
1,1
Titik 2
3,4
Titik 3
5,2
Masukkan jumlah iterasi
10
Pilih metode perhitungan:
1. Divide and Conquer
2. Brute Force
2
Calculate

output:

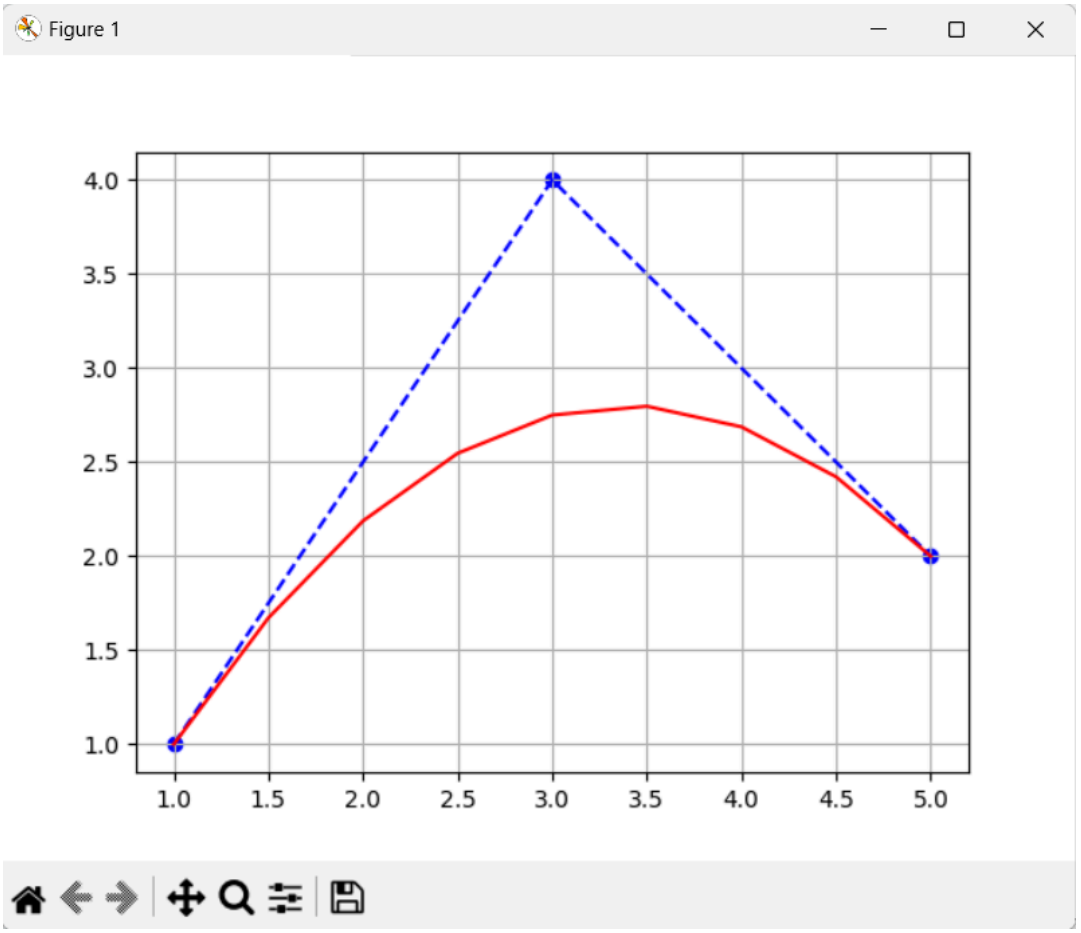



```
-------------------------Brute Force------------------------------
======================================================================
Berikut adalah hasil dari kurva bezier dengan iterasi sebanyak 10
Waktu eksekusi :  0.0020072460174560547 detik
```

b. Input:

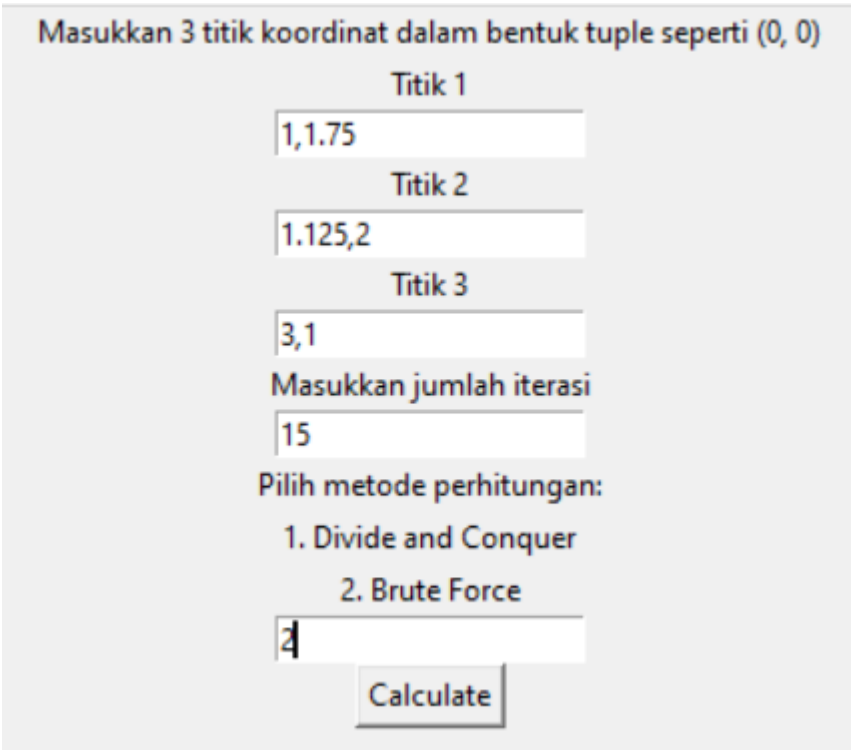

Masukkan 3 titik koordinat dalam bentuk tuple seperti (0, 0)
Titik 1
1,1
Titik 2
3,4
Titik 3
5,2
Masukkan jumlah iterasi
3
Pilih metode perhitungan:
1. Divide and Conquer
2. Brute Force
2
Calculate

output:




```
------------------------Brute Force-------------------------------
========================================================================
Berikut adalah hasil dari kurva bezier dengan iterasi sebanyak 3
Waktu eksekusi :  0.0 detik
```

c. Input:


Masukkan 3 titik koordinat dalam bentuk tuple seperti (0, 0)
Titik 1
1,1.75
Titik 2
1.125,2
Titik 3
3,1
Masukkan jumlah iterasi
15
Pilih metode perhitungan:
1. Divide and Conquer
2. Brute Force
2
Calculate

output:

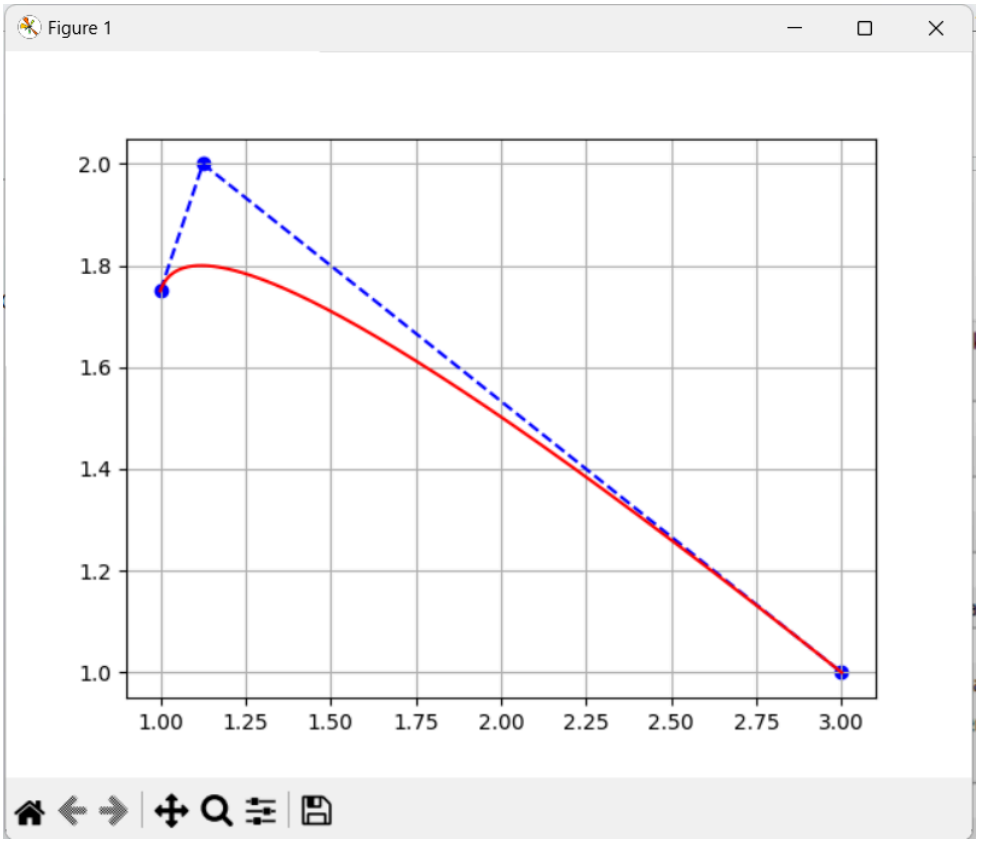
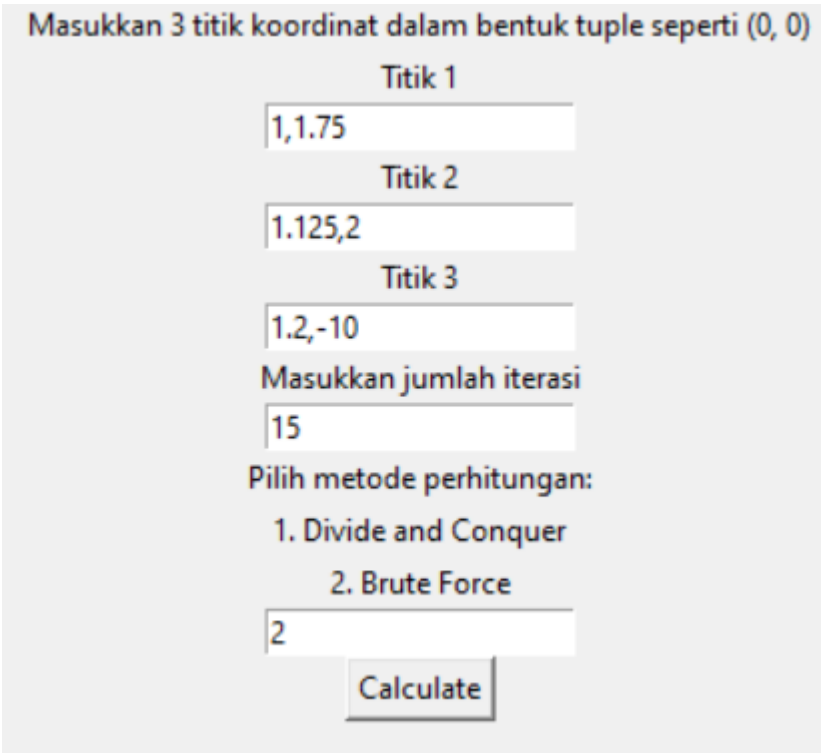


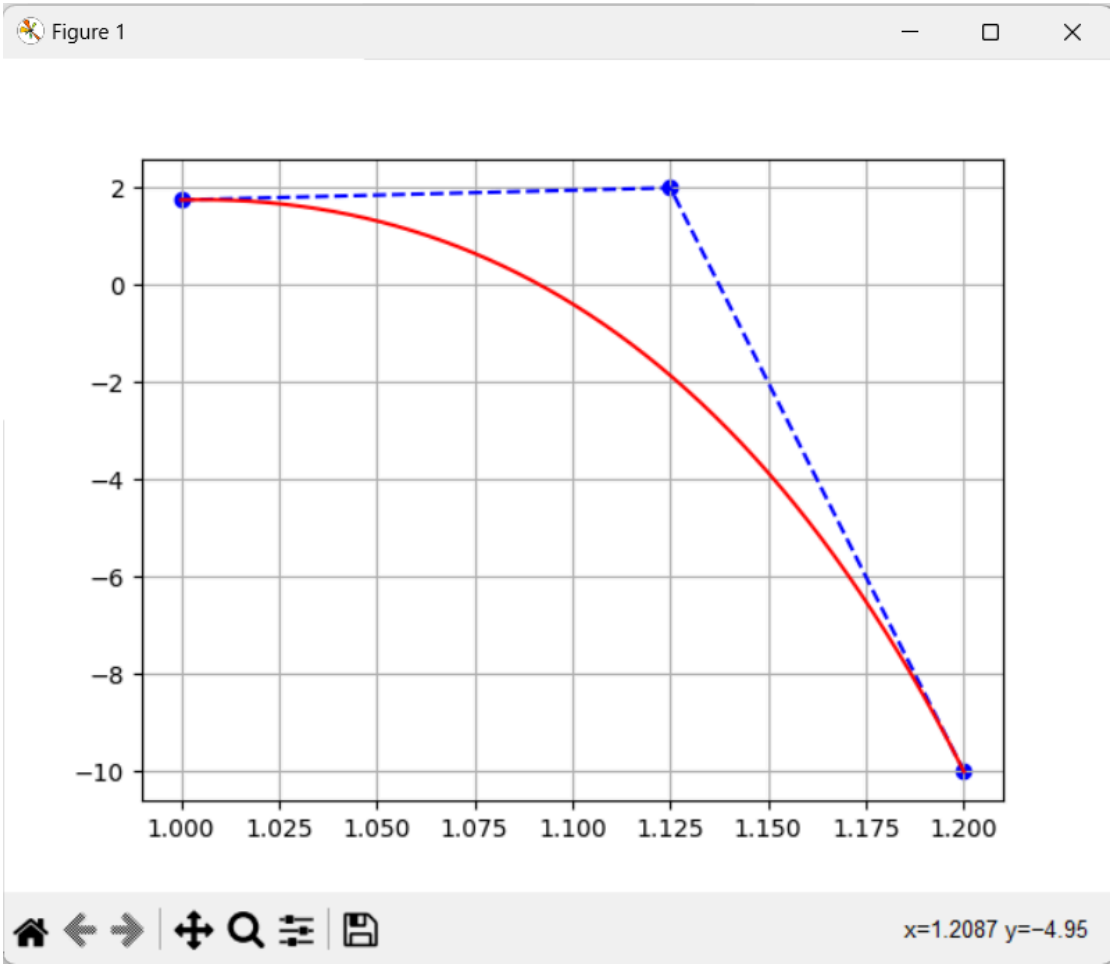
Figure 1
2.0
1.8
1.6
1.4
1.2
1.0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

```
------------------------Brute Force-------------------------------
================================================================
Berikut adalah hasil dari kurva bezier dengan iterasi sebanyak 15
Waktu eksekusi :  0.7467494010925293 detik
```

d. Input:

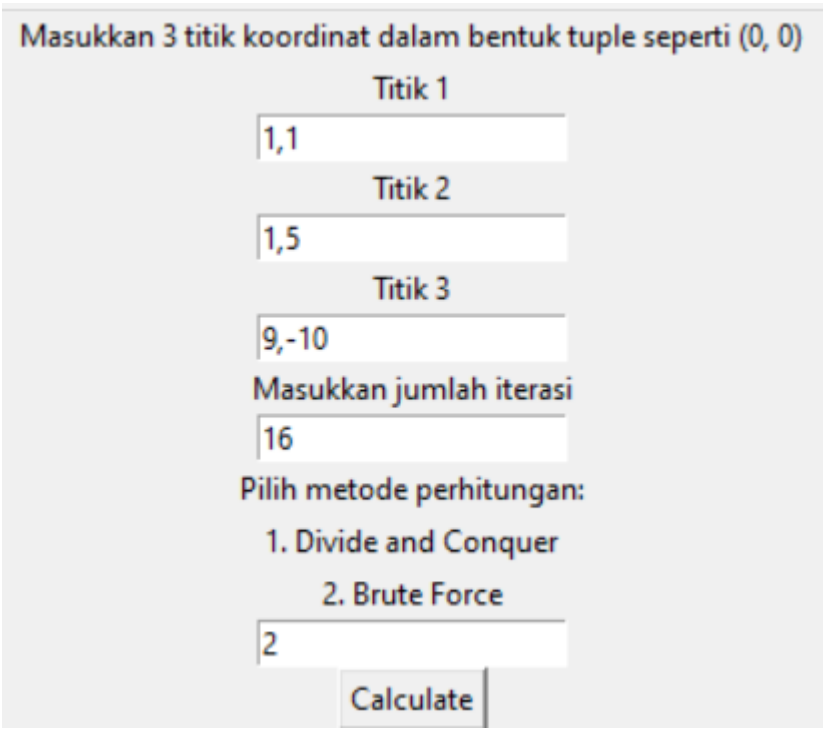


output:

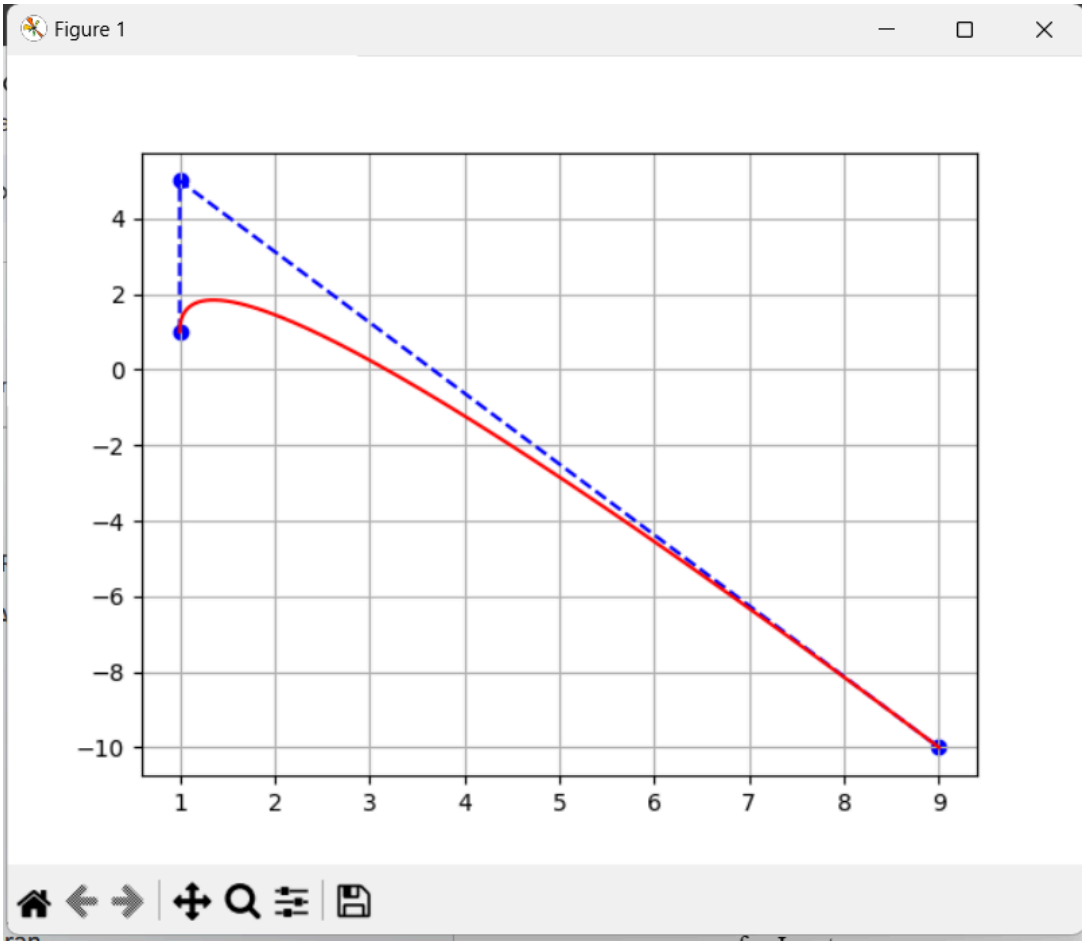



```
-------------------------Brute Force-------------------------------
=======================================================================
Berikut adalah hasil dari kurva bezier dengan iterasi sebanyak 15
Waktu eksekusi :  0.7244539260864258 detik
=======================================================================
```

e. Input:

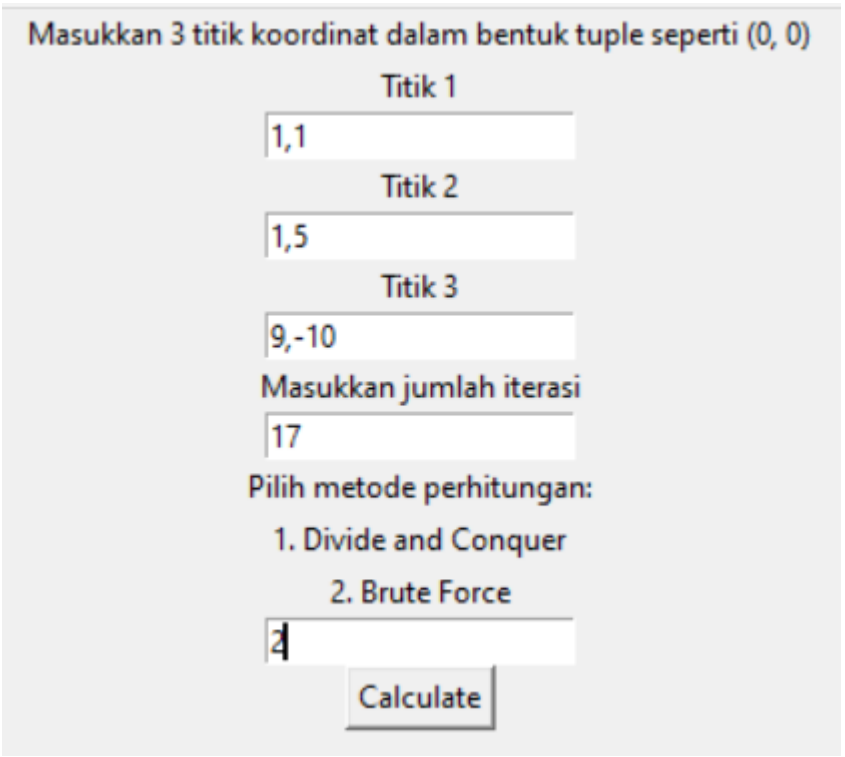

Masukkan 3 titik koordinat dalam bentuk tuple seperti (0, 0)
Titik 1
1,1
Titik 2
1,5
Titik 3
9,-10
Masukkan jumlah iterasi
16
Pilih metode perhitungan:
1. Divide and Conquer
2. Brute Force
2
Calculate

output:




```
-------------------------Brute Force-------------------------------
=======================================================================
Berikut adalah hasil dari kurva bezier dengan iterasi sebanyak 16
Waktu eksekusi :  2.8805553913116455 detik
=======================================================================
```

f. Input


Masukkan 3 titik koordinat dalam bentuk tuple seperti (0, 0)
Titik 1
1,1
Titik 2
1,5
Titik 3
9,-10
Masukkan jumlah iterasi
17
Pilih metode perhitungan:
1. Divide and Conquer
2. Brute Force
2
Calculate

output:

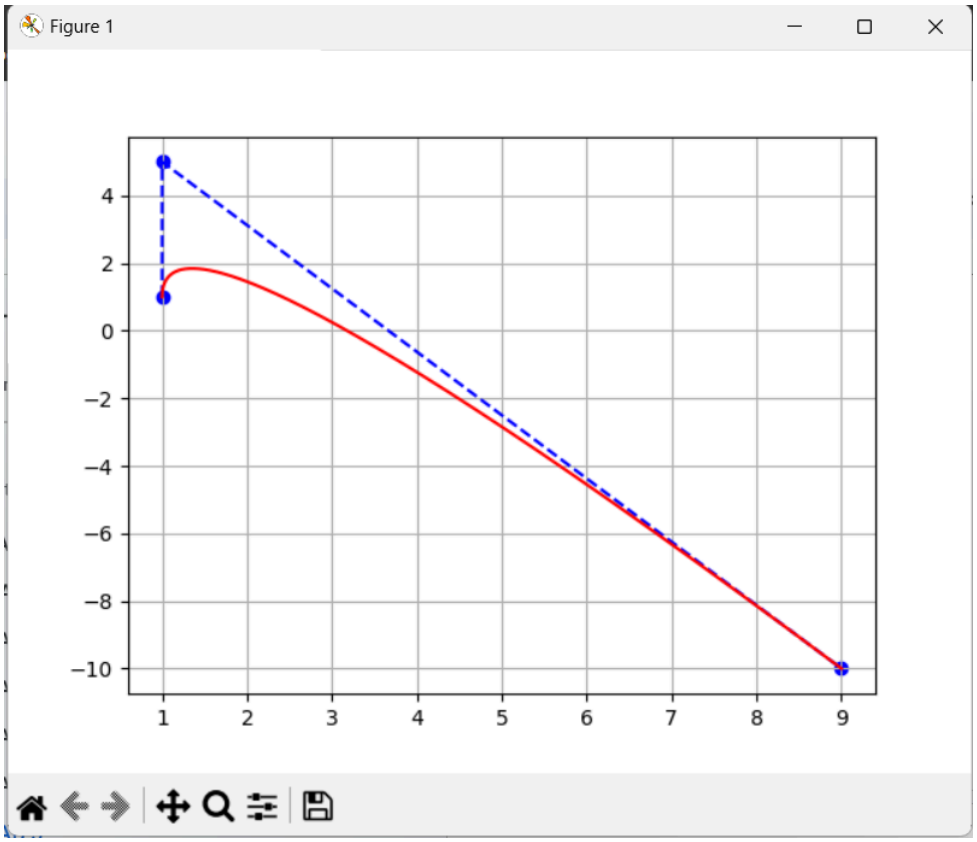


Figure 1
4
2
0
−2
−4
−6
−8
−10
1
2
3
4
5
6
7
8
9

```
-------------------------Brute Force-------------------------------
=================================================================
Berikut adalah hasil dari kurva bezier dengan iterasi sebanyak 17
Waktu eksekusi :  11.625321626663208 detik
```

BAB VI

Hasil Analisis Perbandingan Brute force dengan DNC

Perbandingan antara algoritma *brute force* dan algoritma *divide and conquer* (DNC) tetap tergantung berbagai kondisi. Secara keseluruhan jika dilakukan iterasi dalam jumlah yang cukup besar (berkisar diatas 15 iterasi) algoritma *divide and conquer* akan jauh lebih cepat dibandingkan *brute force*. Berikut penjelasannya.

1. Brute Force:

   Metode brute force menghasilkan kurva Bezier dengan cara menghitung titik tengah secara berulang dari segmen garis yang terdefinisi oleh titik kontrol dan dilakukan secara utuh dan sekuensial. Kompleksitas waktu dari algoritma brute force ini sekitar O(2^N), di mana N adalah jumlah iterasi. Ini karena setiap iterasi menggandakan jumlah titik tengah yang dihitung. Jika jumlah iterasi kecil, brute force mungkin lebih cepat karena overhead dari rekursi dan pengelompokan titik tidak signifikan.

2. Divide and Conquer:

   Metode DNC membagi kurva menjadi dua bagian pada setiap iterasi dan menghitung titik tengah dari dua titik kontrol yang baru. Kompleksitas waktu dari algoritma DNC adalah O(2^N), tetapi faktor konstantanya lebih kecil daripada brute force karena setiap iterasi hanya membagi masalah menjadi dua bagian, tidak menggandakannya. Jadi, meskipun kompleksitas algoritma DNC juga eksponensial, waktu eksekusi sebenarnya cenderung lebih cepat karena overhead rekursi yang lebih rendah dan jumlah titik tengah yang lebih sedikit yang perlu dihitung.

Jadi dalam kasus banyak iterasi, DNC lebih unggul karena meskipun kedua algoritma memiliki kompleksitas yang sama secara teoritis, faktor konstanta yang lebih rendah pada DNC membuatnya lebih efisien dalam praktiknya. Selain itu, dalam kasus ini, algoritma brute force menggunakan loop dan pop untuk mengelola titik tengah, yang dapat menghabiskan waktu tambahan, sementara DNC menggunakan rekursi yang cenderung lebih efisien dalam menangani struktur pemanggilan. Itu juga dapat berkontribusi pada perbedaan waktu eksekusi yang diamati.

BAB VI

Implementasi Bonus

Pada tugas ini saya mengerjakan bonus yang menganimasikan visualisasi dari Kurva Beziernya. Implementasi ini saya menggunakan library matplotlib pada python dan menggunakan sebuah fungsi FuncAnimation. Maka setiap kali iterasi DNC dilakukan akan terlihat bagaimana *step by step* dari Kurva Bezier ini akan terbentuk

Pranala Repository Github

Link Github :

https://github.com/sibobbbbbb/Tucil2_13522142.git

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	✔	