

Laporan Tugas Kecil 2
IF2211 Strategi Algoritma

**Membangun Kurva Bézier dengan Algoritma Titik
Tengah berbasis Divide and Conquer**



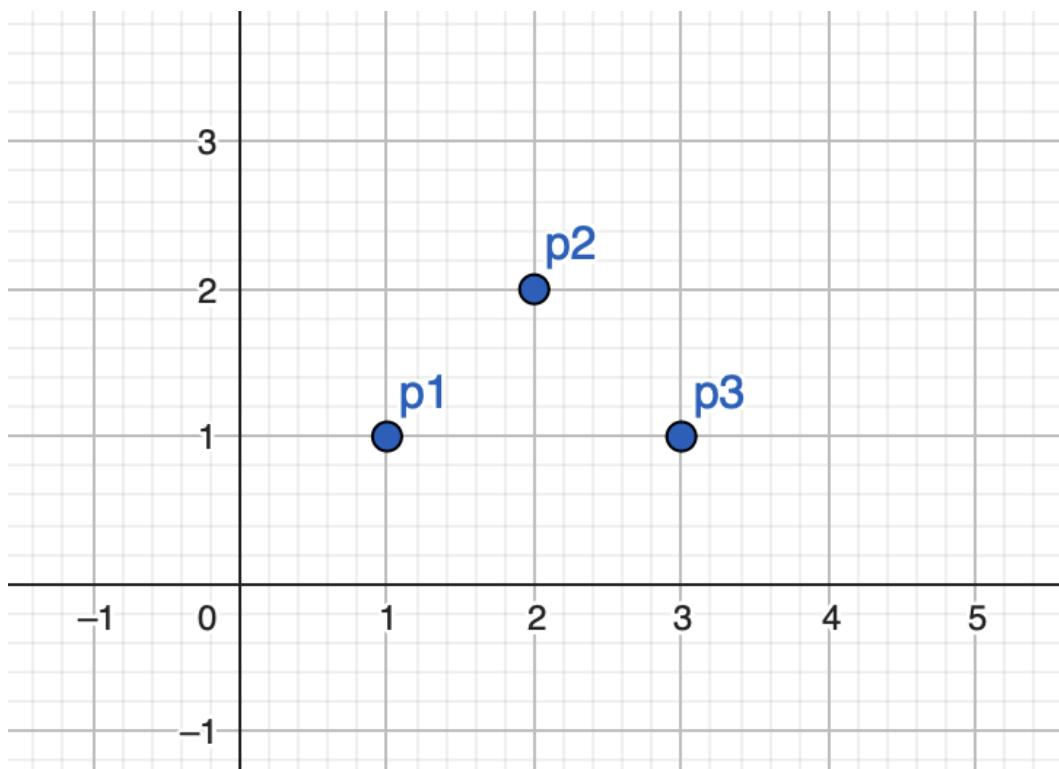
Disusun Oleh
Fabian Radenta Bangun 13522105

Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Semester IV - 2024

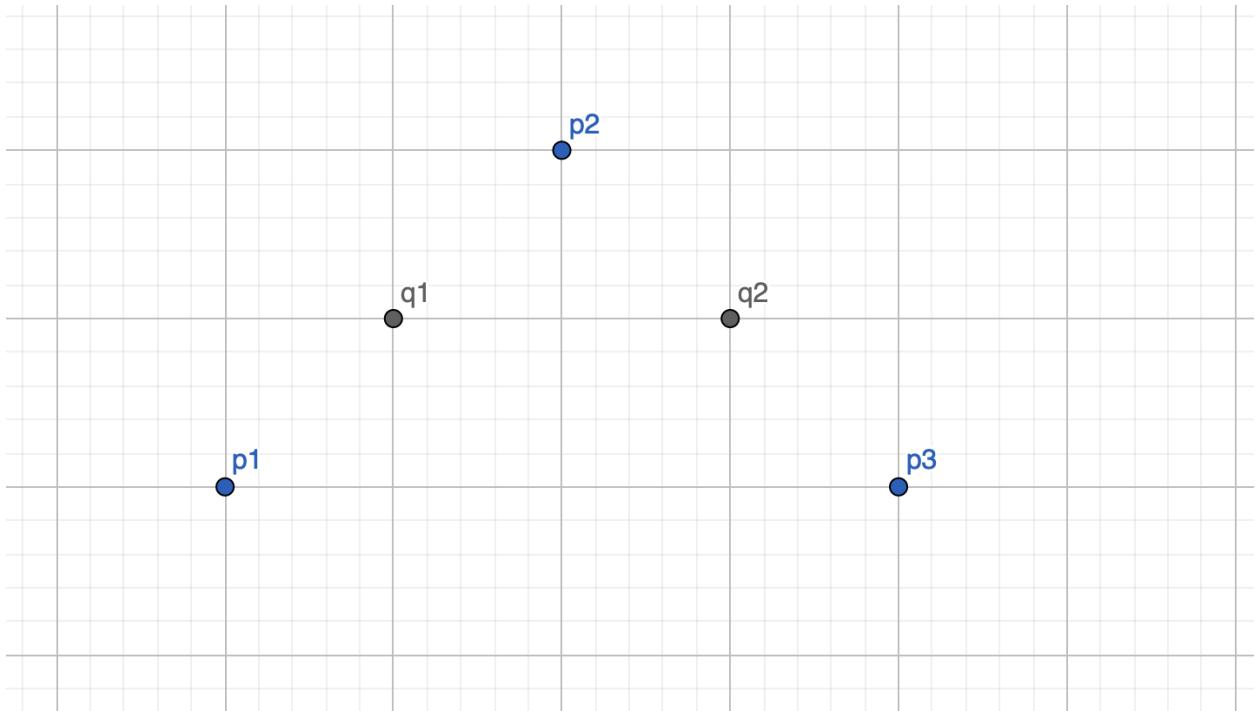
A. Algoritma *Divide and Conquer* dalam Membangun Kurva Bézier dengan Algoritma Titik Tengah

Ide umum dari pembangunan Kurva Bézier ini adalah dengan mencari titik-titik yang dilewati kurva. Titik yang didapatkan bergantung pada jumlah iterasi yang dilakukan. Jumlah sisi yang didapatkan adalah senilai 2^n dengan n adalah jumlah iterasi. Kemudian dari titik-titik yang telah didapatkan, gambarkan garis yang menghubungkan titik-titik tersebut sehingga membentuk suatu kurva. Perlu diketahui bahwa semakin banyak jumlah iterasi yang dilakukan, akan semakin banyak pula sisi yang diperoleh. Dan semakin banyak sisi yang diperoleh, semakin akurat juga kurva yang dihasilkan.

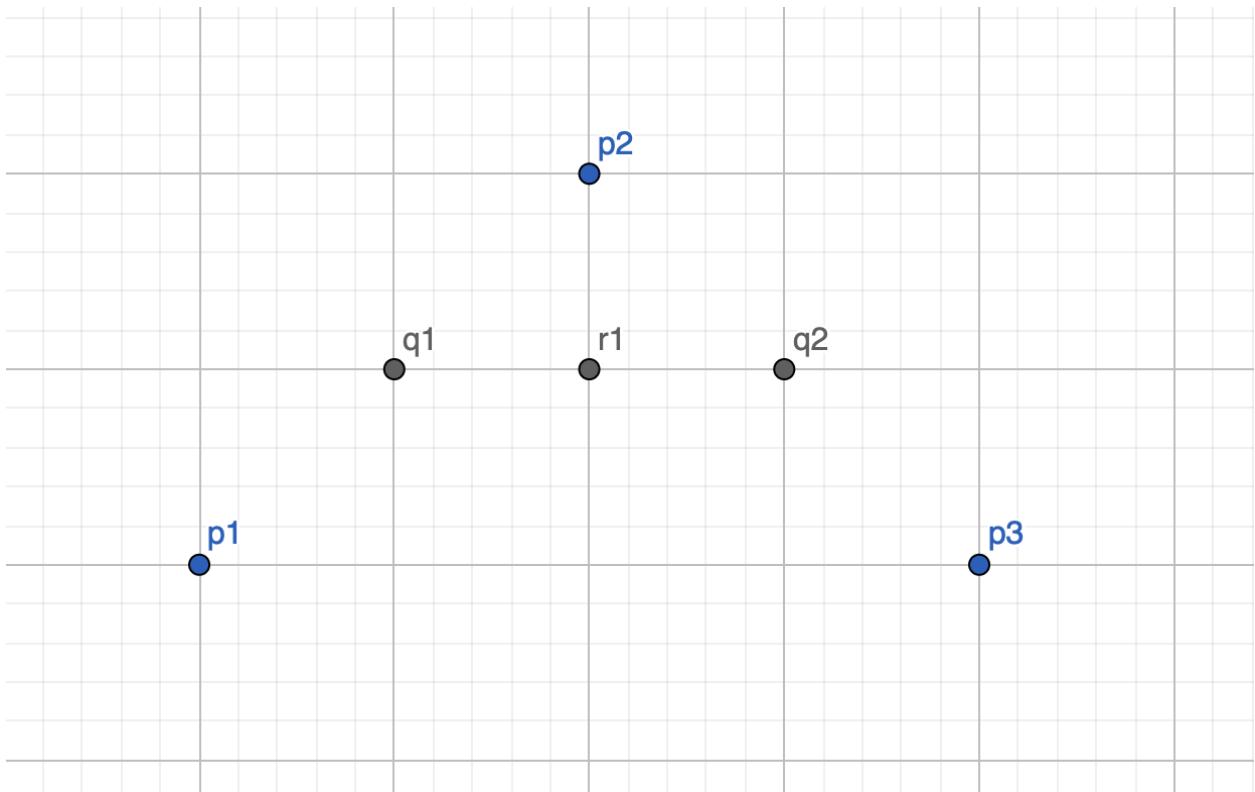
Untuk kasus titik kontrol pertama, kedua dan ketiga ada pada titik (1,1), (2,2), dan (3,1) secara berurutan diilustrasikan dalam gambar sebagai berikut.



Kemudian kita deklarasikan titik q₁ yang merupakan titik tengah dari p₁ dan p₂ dan q₂ yang merupakan titik tengah dari p₂ dan p₃.



Dari gambar tersebut kita deklarasikan lagi titik r1 yang merupakan titik tengah q1 dan q2.



Jika dihubungkan garis p1-r1-p3 maka didapatkan kurva dari titik p1,p2,p3 dengan jumlah iterasinya adalah satu. Untuk jumlah iterasi sama dengan dua maka kita temukan lagi titik yang berada di tengah titik p1q1 yang merupakan titik tengah dari p1 dan q1 dan titik q1r1 yang merupakan titik tengah dari q1 dan r1. Kemudian temukan titik tengah p1,q1,r1 yang merupakan titik tengah dari p1q1 dan q1r1. Begitu pula kita lakukan di sebelah kanan kurva. Kemudian kita tarik garis yang menggabungkan titik-titik tersebut (jumlah titik ada 5). Begitu juga dengan jumlah iterasi yang lebih besar

Implementasi algoritma *divide and conquer* dalam program ini adalah sebagai berikut :

1. Rekursi akan berhenti pada *base case* nya, yaitu jika nilai jumlah iterasi adalah satu. Karena jika nilai jumlah iterasinya sama dengan satu maka titik yang dilewati kurva hanya tiga. Yang pertama adalah titik kontrol pertama. Kemudian yang kedua adalah titik tengah yang didapat dari fungsi mid_of_mid(p1,p2,p3). Yang terakhir adalah titik kontrol ketiga.
2. Untuk jumlah iterasi lebih dari satu maka program akan membagi eksekusi menjadi dua sub-*array*, kita nyatakan dengan kiri dan kanan. Untuk masing-masing kiri dan kanan dipanggil lagi fungsi divide and conquer, hanya saja parameter fungsi tersebut berbeda. Sub-*array* kiri akan menangani kurva yang terbentuk disebelah kiri titik kontrol kedua dan sub-*array* kanan akan menangani kurva yang terbentuk disebelah kanan titik kontrol kedua. Masing-masing sub-*array* akan terus memanggil fungsi divide and conquer lagi sampai jumlah iterasinya bernilai satu. Pada setiap pemanggilan jumlah iterasi akan dikurangi satu.

B. Algoritma *Brute Force* dalam Membangun Kurva Bézier

Untuk algoritma *brute force*, ide umumnya sama seperti yang telah dijelaskan pada bagian A. Hanya saja metode dalam mencari titik-titik yang dilewati oleh kurva berbeda. Implementasi algoritma *brute force* dalam program ini adalah sebagai berikut :

1. Dideklarasikan fungsi persamaan umum dalam mencari nilai titik yang dilewati kurva pada nilai t tertentu dengan, yaitu :

$$\mathbf{B}(t) = (1-t)^2 \mathbf{P}_0 + 2(1-t)t \mathbf{P}_1 + t^2 \mathbf{P}_2 , t \in [0, 1].$$

2. Setelah mencoba beberapa kasus dengan algoritma *divide and conquer* yang sebelumnya telah dibuat, didapatkan pola bahwa banyaknya sisi yang didapatkan adalah sebesar 2^n dengan n adalah jumlah iterasi.
3. Dari penemuan pola tersebut, kita bisa melakukan pengulangan untuk mendapatkan nilai t . Untuk iterasi *integer* i sebanyak 2^n kali maka nilai t nya adalah $i/2n$.
4. Setiap titik yang didapat dari hasil iterasi dimasukkan kedalam array dan dikembalikan menjadi kembalian fungsi.

C. Source Code Program

1. File Algoritma Brute Force (bruteforce.py)

Pada file ini dilakukan perhitungan untuk memperoleh titik-titik yang dilewati kurva dengan pendekatan algoritma Brute Force.

- a. Import Kelas Point dari file point.py dan library matplotlib

```
from point import Point
```

- b. Fungsi yang mengembalikan titik yang dilewati kurva pada nilai t tertentu

```
def bezier_curve_function(point1, point2, point3, t) :
    temp_x = ((1-t)**2)*point1.x + 2*(1-t)*t*point2.x + (t**2)*point3.x
    temp_y = ((1-t)**2)*point1.y + 2*(1-t)*t*point2.y + (t**2)*point3.y
    return Point(temp_x,temp_y)
```

- c. Fungsi yang mengembalikan sebuah *array of point* dari titik-titik yang diperoleh dengan algoritma *brute force*

```
def main_function(point1, point2, point3, n_iterate) :
    hasil = []
    for i in range (2*n_iterate+1) :
        hasil.append(bezier_curve_function(point1, point2, point3, i/(2*n_iterate)))
    return hasil
```

2. File Algoritma Divide and Conquer (dnc.py)

Pada file ini dilakukan perhitungan untuk memperoleh titik-titik yang dilewati kurva dengan pendekatan algoritma Divide and Conquer.

- a. Import Kelas Point dari file point.py dan library matplotlib

```
from point import Point
```

b. Fungsi yang mengembalikan nilai tengah dari 2 titik

```
def mid_point(p1,p2) :  
    mid_x = (p1.x + p2.x)/2  
    mid_y = (p1.y + p2.y)/2  
    return Point(mid_x,mid_y)
```

c. Fungsi yang mengembalikan nilai tengah dari nilai tengah 2 titik

```
def mid_of_mid(p1,p2,p3) :  
    left_mid = mid_point(p1,p2)  
    right_mid = mid_point(p2,p3)  
    return mid_point(left_mid,right_mid)
```

d. Fungsi yang mengembalikan sebuah *array of point* dari titik-titik yang diperoleh dengan algoritma *divide and conquer*

```
def divide_and_conquer(point1, point2, point3, n_iterate) :  
    # base case  
    if (n_iterate==1) :  
        return [point1,mid_of_mid(point1,point2,point3)]  
  
    #recurrent  
    else :  
        left = divide_and_conquer(point1,mid_point(point1,point2),mid_of_mid(point1,point2,point3), n_iterate-1)  
        right = divide_and_conquer(mid_of_mid(point1,point2,point3), mid_point(point2,point3),point3, n_iterate-1)  
        return left + right
```

3. File Kelas Point (point.py)

Pada file ini dideklarasikan Kelas Point yang memiliki atribut x dan y yang menyatakan absis dan oordinat titik.

a. Deklarasi Kelas Point

```
class Point :  
    def __init__(self, x, y) :  
        self.x = x  
        self.y = y
```

4. File Prosedur dan Fungsi lain (utils.py)

Pada file ini terdapat beberapa subprogram (prosedur dan fungsi) akan dipanggil di program utama. File ini dibuat untuk meminimalisasi banyaknya baris program pada file main.py

a. Import Kelas Point dari file point.py

```
from point import Point
```

b. Prosedur untuk menampilkan pesan pembuka program

```

def welcome() :
    print("""
Selamat datang di Program Visualisasi Kurva Bézier

Untuk menggunakan program ini, silakan persiapkan masukan berikut :
- Jumlah iterasi yang digunakan dalam perhitungan
- Tiga buah titik kontrol (x,y) yang membentuk kurva

Ayo kita mulai!
""")

```

c. Fungsi *processing input* untuk merubah input yang bertipe string menjadi point

```

def process_input_point(input_str_1, input_str_2, input_str_3) :
    # pisahkan absis dan oordinat
    x_str_1, y_str_1 = input_str_1.split(',')
    x_str_2, y_str_2 = input_str_2.split(',')
    x_str_3, y_str_3 = input_str_3.split(',')

    # casting tipe data
    x1 = float(x_str_1)
    x2 = float(x_str_2)
    x3 = float(x_str_3)
    y1 = float(y_str_1)
    y2 = float(y_str_2)
    y3 = float(y_str_3)

    # declare point
    point1 = Point(x1,y1)
    point2 = Point(x2,y2)
    point3 = Point(x3,y3)
    return point1, point2, point3

```

d. Prosedur untuk menampilkan menu pilihan

```

def method_option() :
    print("""
Pilihan metode :
1. Divide and Conquer
2. Brute-Force
""")

```

- e. Prosedur untuk menampilkan pesan terima kasih

```
def thanks() :  
    print('\n'+23*'=')  
    print(5*' '+'Terima kasih"+6*' ')  
    print(23*'=')  
    print(7*' '+'° , +♡+ . ° '+7*' ')
```

5. File Program Utama (main.py)

- a. Import file eksternal, library matplotlib, dan library time

```
import bruteforce
import dnc
import utils
import matplotlib.pyplot as plot
import time
```

- b. Menampilkan pesan pembuka dan try untuk *error handling*

```
utils.welcome()  
try :
```

- c. Terima *input* dari *user*

```
# terima input jumlah iterasi dan titik kontrol dari user
iterasi = int(input("Jumlah iterasi : "))
input_point_1 = input("Titik 1 : ")
input_point_2 = input("Titik 2 : ")
input_point_3 = input("Titik 3 : ")
```

- d. Proses input dari string menjadi point

```
# processing input dari string menjadi point  
p1, p2, p3 = utils.process_input_point(input_point_1, input_point_2, input_point_3)
```

- e Memeriksa apakah ada kesalahan dalam input

```
# memeriksa apakah ada kesalahan dalam input
if p1 is None or p2 is None or p3 is None:
    raise ValueError("Titik yang dimasukkan tidak valid.")
```

- f. Terima input metode algoritma

```
# terima input metode algoritma  
utils.method_option()  
tipe = int(input("Masukkan pilihan anda : "))
```

- g. Periksa apakah input metode algoritma sudah valid

```

# periksa apakah input sudah benar
while tipe not in [1,2] :
    tipe = int(input("Tolong masukkan input yang valid\nMasukkan pilihan anda : "))

```

h. Program utama

```

# program utama
start_time = time.time()
hasil = []
if (tipe==1) :
    hasil = dnc.divide_and_conquer(p1, p2, p3, iterasi)
    hasil.append(p3)
else :
    hasil = bruteforce.main_function(p1, p2, p3, iterasi)

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

# menampilkan execution time
print("Waktu eksekusi program :", execution_time, "seconds")

```

i. Menggambar kurva

```

# menggambar kurva
x_points = [point.x for point in hasil]
y_points = [point.y for point in hasil]
plot.figure(figsize=(8, 6))
plot.plot([p1.x, p2.x], [p1.y, p2.y], color='red')
plot.plot([p2.x, p3.x], [p2.y, p3.y], color='red')
plot.scatter(x_points, y_points, color='black', s=25)
plot.scatter(p1.x, p1.y, color='red')
plot.scatter(p2.x, p2.y, color='red')
plot.scatter(p3.x, p3.y, color='red')
plot.plot(x_points, y_points, color='blue')
plot.xlabel('X Coordinate')
plot.ylabel('Y Coordinate')
plot.title('Bezier Curve')
plot.grid(True)
plot.gca().set_aspect("equal", adjustable="box")
plot.show()

```

j. Error handling

```
except ValueError as e :  
    print("Program berhenti. Error:",e)
```

- k. Menampilkan pesan terima kasih

```
utils.thanks()
```

D. Program Testing

1. Algoritma *Divide and Conquer*

- a. Percobaan 1

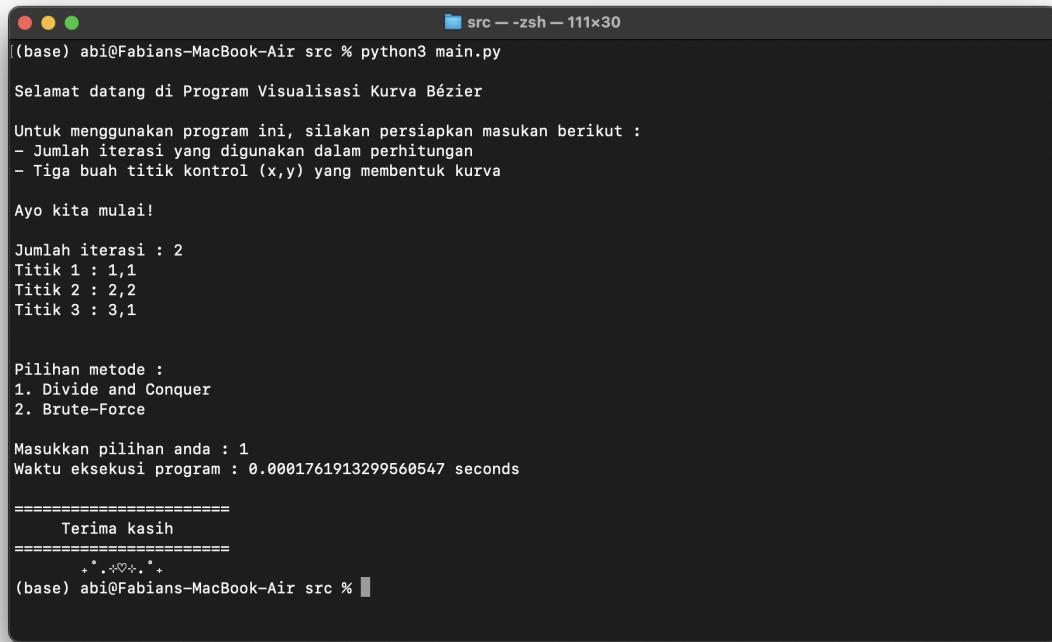
Data

Jumlah iterasi : 2

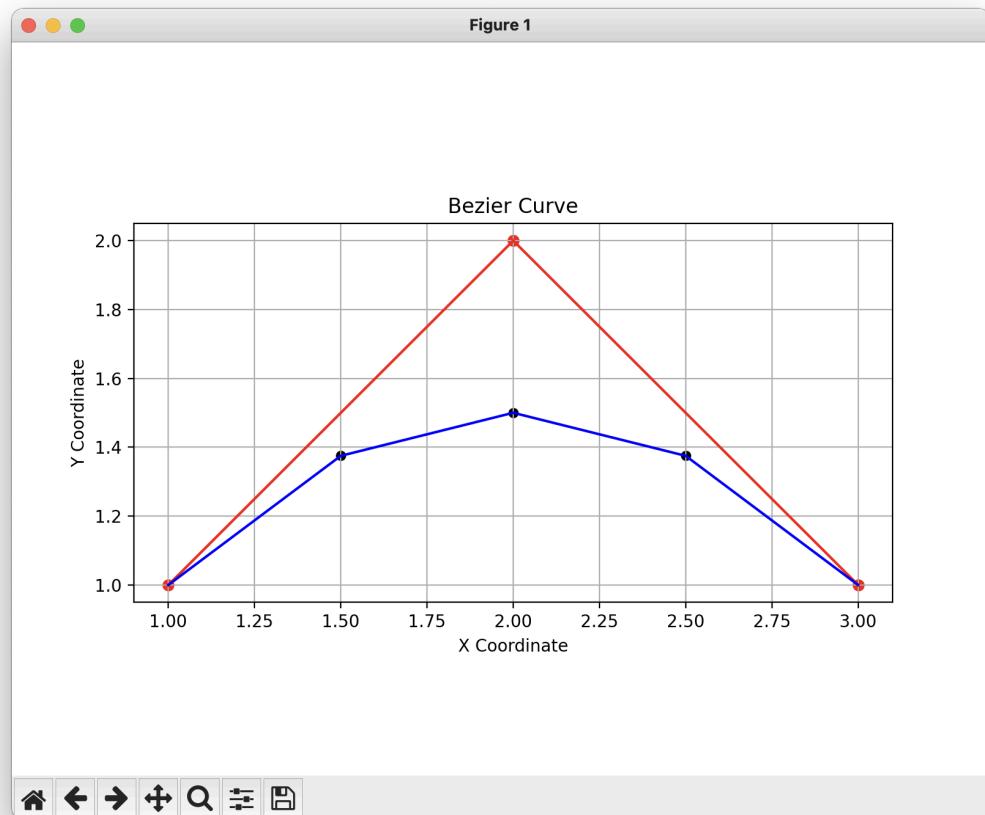
Titik Kontrol 1 : (1,1)

Titik Kontrol 1 : (2,2)

Titik Kontrol 1 : (3,1)



```
src -- zsh -- 111x30  
[(base) abi@Fabians-MacBook-Air src % python3 main.py  
Selamat datang di Program Visualisasi Kurva Bézier  
Untuk menggunakan program ini, silakan persiapkan masukan berikut :  
- Jumlah iterasi yang digunakan dalam perhitungan  
- Tiga buah titik kontrol (x,y) yang membentuk kurva  
Ayo kita mulai!  
Jumlah iterasi : 2  
Titik 1 : 1,1  
Titik 2 : 2,2  
Titik 3 : 3,1  
  
Pilihan metode :  
1. Divide and Conquer  
2. Brute-Force  
  
Masukkan pilihan anda : 1  
Waktu eksekusi program : 0.0001761913299560547 seconds  
=====  
Terima kasih  
=====  
+(.~*~.*+)  
(base) abi@Fabians-MacBook-Air src %
```



b. Percobaan 2

Data

Jumlah iterasi : 4

Titik Kontrol 1 : (5,1)

Titik Kontrol 1 : (1,2)

Titik Kontrol 1 : (12,5)

```
src -- zsh -- 111x30
(base) abi@Fabians-MacBook-Air src % python3 main.py
Selamat datang di Program Visualisasi Kurva Bézier

Untuk menggunakan program ini, silakan persiapkan masukan berikut :
- Jumlah iterasi yang digunakan dalam perhitungan
- Tiga buah titik kontrol (x,y) yang membentuk kurva

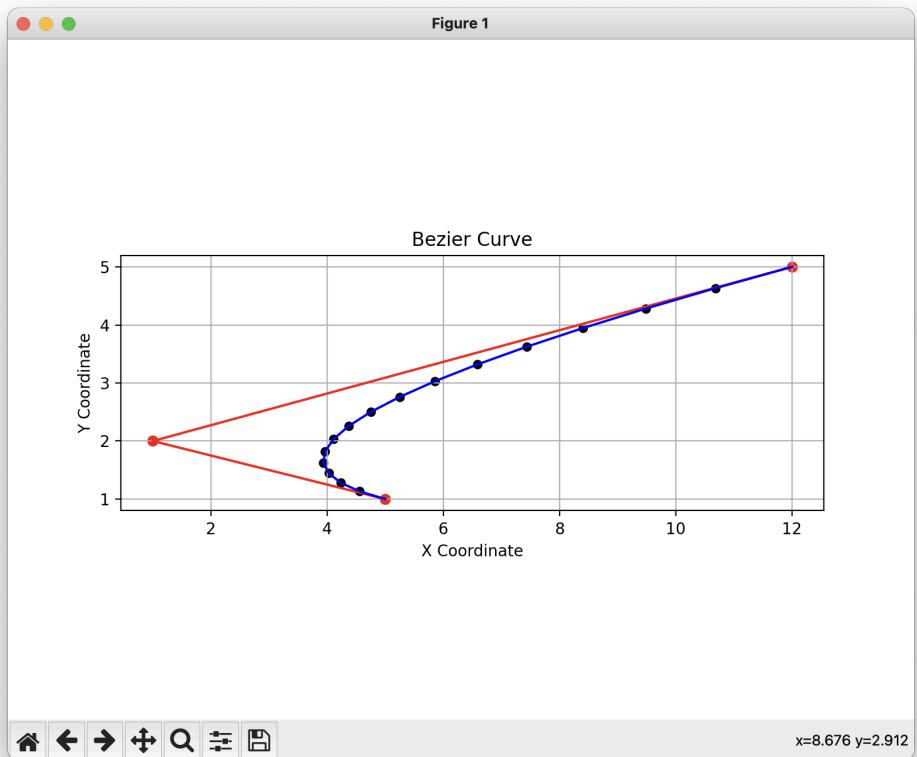
Ayo kita mulai!

Jumlah iterasi : 4
Titik 1 : 5,1
Titik 2 : 1,2
Titik 3 : 12,5

Pilihan metode :
1. Divide and Conquer
2. Brute-Force

Masukkan pilihan anda : 1
Waktu eksekusi program : 0.0007681846618652344 seconds

=====
Terima kasih
=====
+ .~^~. +
(base) abi@Fabians-MacBook-Air src %
```



c. Percobaan 3

Data

Jumlah iterasi : 3

Titik Kontrol 1 : (3,11)

Titik Kontrol 1 : (2,2)

Titik Kontrol 1 : (0,5)

```
src -- zsh -- 111x30
(base) abi@Fabians-MacBook-Air src % python3 main.py
Selamat datang di Program Visualisasi Kurva Bézier

Untuk menggunakan program ini, silakan persiapkan masukan berikut :
- Jumlah iterasi yang digunakan dalam perhitungan
- Tiga buah titik kontrol (x,y) yang membentuk kurva

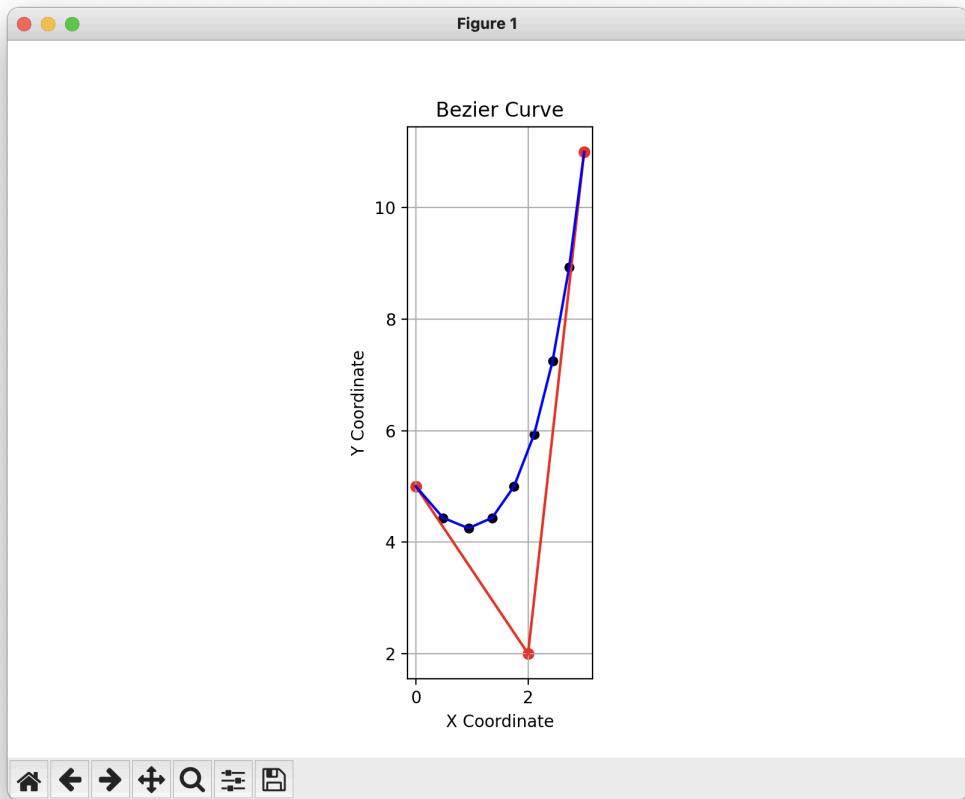
Ayo kita mulai!

Jumlah iterasi : 3
Titik 1 : 3,11
Titik 2 : 2,2
Titik 3 : 0,5

Pilihan metode :
1. Divide and Conquer
2. Brute-Force

Masukkan pilihan anda : 1
Waktu eksekusi program : 0.00031065940856933594 seconds

=====
Terima kasih
=====
+ .+*+.+
(base) abi@Fabians-MacBook-Air src %
```



d. Percobaan 4

Data

Jumlah iterasi : 5

Titik Kontrol 1 : (2,1)

Titik Kontrol 1 : (5,2)

Titik Kontrol 1 : (0,0)

```
src -- zsh -- 111x30
(base) abi@Fabians-MacBook-Air src % python3 main.py
Selamat datang di Program Visualisasi Kurva Bézier

Untuk menggunakan program ini, silakan persiapkan masukan berikut :
- Jumlah iterasi yang digunakan dalam perhitungan
- Tiga buah titik kontrol (x,y) yang membentuk kurva

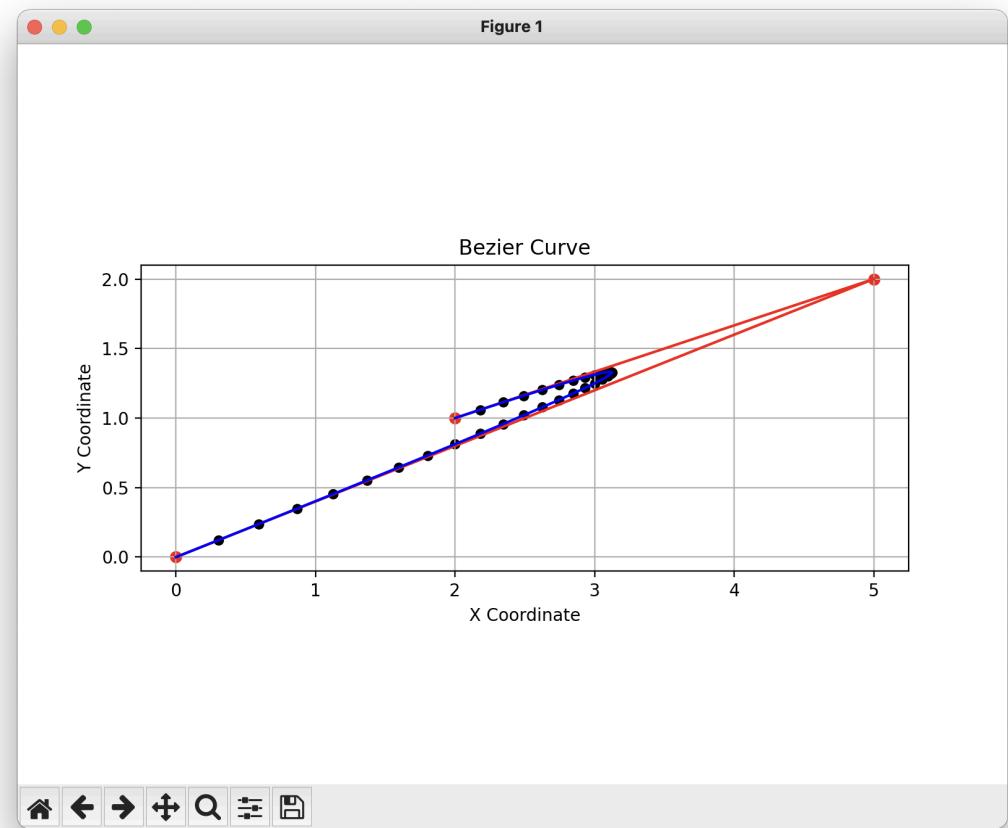
Ayo kita mulai!

Jumlah iterasi : 5
Titik 1 : 2,1
Titik 2 : 5,2
Titik 3 : 0,0

Pilihan metode :
1. Divide and Conquer
2. Brute-Force

Masukkan pilihan anda : 1
Waktu eksekusi program : 0.0021219253540039062 seconds

=====
Terima kasih
=====
+ . . . + . +
(base) abi@Fabians-MacBook-Air src %
```



e. Percobaan 5

Data

Jumlah iterasi : 3
Titik Kontrol 1 : (0,0)
Titik Kontrol 1 : (2,1)
Titik Kontrol 1 : (10,7)

```
src - zsh - 111x30
(base) abi@Fabians-MacBook-Air src % python3 main.py

Selamat datang di Program Visualisasi Kurva Bézier

Untuk menggunakan program ini, silakan persiapkan masukan berikut :
- Jumlah iterasi yang digunakan dalam perhitungan
- Tiga buah titik kontrol (x,y) yang membentuk kurva

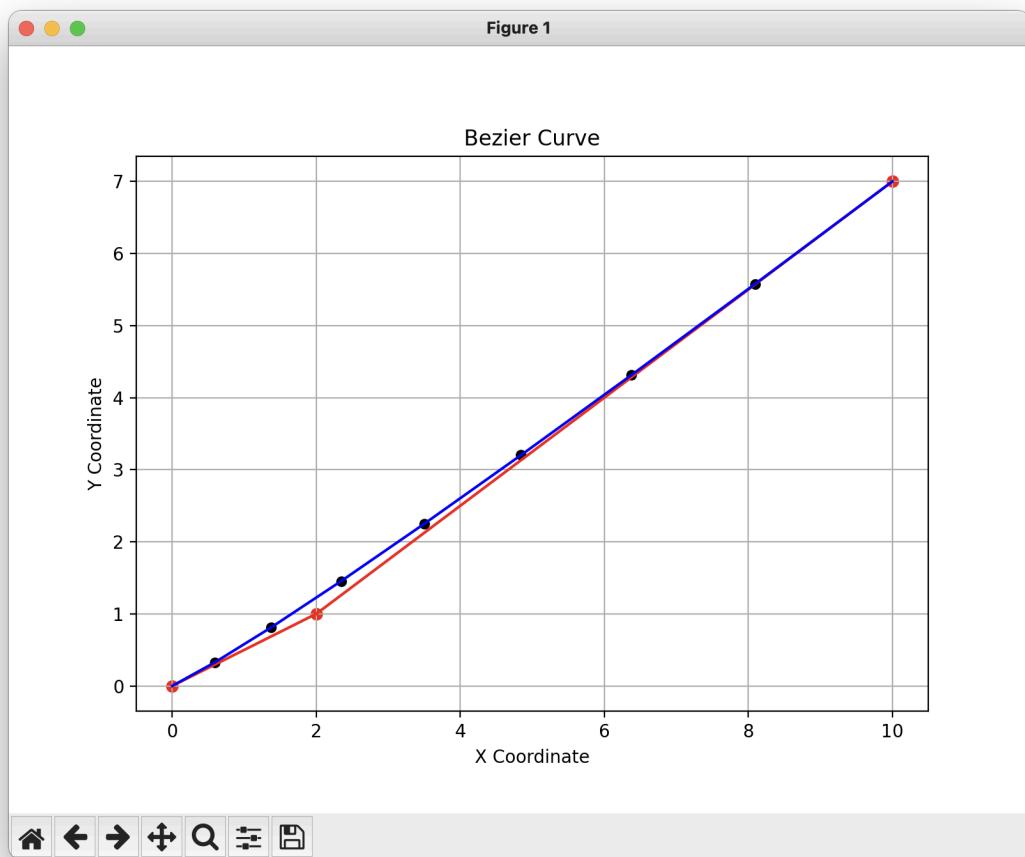
Ayo kita mulai!

Jumlah iterasi : 3
Titik 1 : 0,0
Titik 2 : 2,1
Titik 3 : 10,7

Pilihan metode :
1. Divide and Conquer
2. Brute-Force

Masukkan pilihan anda : 1
Waktu eksekusi program : 0.00032830238342285156 seconds

=====
Terima kasih
=====
+ .•♥•+.+
(base) abi@Fabians-MacBook-Air src %
```



f. Percobaan 6

Data

Jumlah iterasi : 7

Titik Kontrol 1 : (-3,-2)

Titik Kontrol 1 : (5,7)

Titik Kontrol 1 : (8,3)

```
src -- zsh -- 111x30
(base) abi@Fabians-MacBook-Air src % python3 main.py
Selamat datang di Program Visualisasi Kurva Bézier

Untuk menggunakan program ini, silakan persiapkan masukan berikut :
- Jumlah iterasi yang digunakan dalam perhitungan
- Tiga buah titik kontrol (x,y) yang membentuk kurva

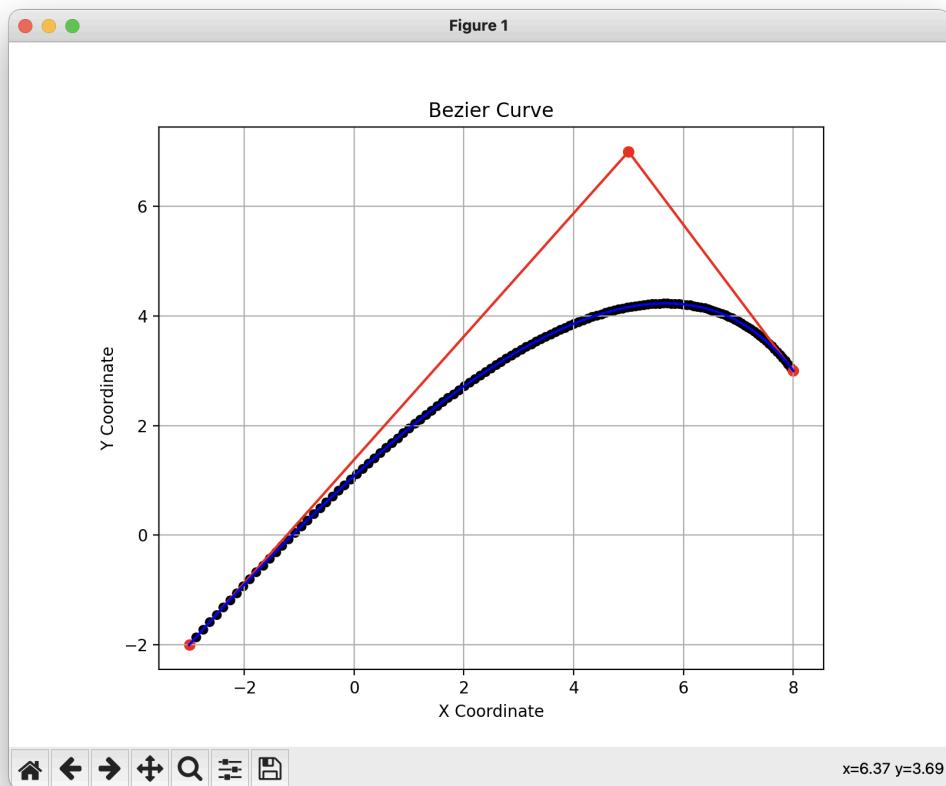
Ayo kita mulai!

Jumlah iterasi : 7
Titik 1 : -3,-2
Titik 2 : 5,7
Titik 3 : 8,3

Pilihan metode :
1. Divide and Conquer
2. Brute-Force

Masukkan pilihan anda : 1
Waktu eksekusi program : 0.0027990341186523438 seconds

=====
Terima kasih
=====
+ .X^X. +
(base) abi@Fabians-MacBook-Air src %
```



2. Algoritma *Brute Force*

a. Percobaan 1

Data

Jumlah iterasi : 2

Titik Kontrol 1 : (1,1)

Titik Kontrol 1 : (2,2)

Titik Kontrol 1 : (3,1)

```

src -- zsh - 111x30
(base) abi@Fabians-MacBook-Air src % python3 main.py
Selamat datang di Program Visualisasi Kurva Bézier

Untuk menggunakan program ini, silakan persiapkan masukan berikut :
- Jumlah iterasi yang digunakan dalam perhitungan
- Tiga buah titik kontrol (x,y) yang membentuk kurva

Ayo kita mulai!

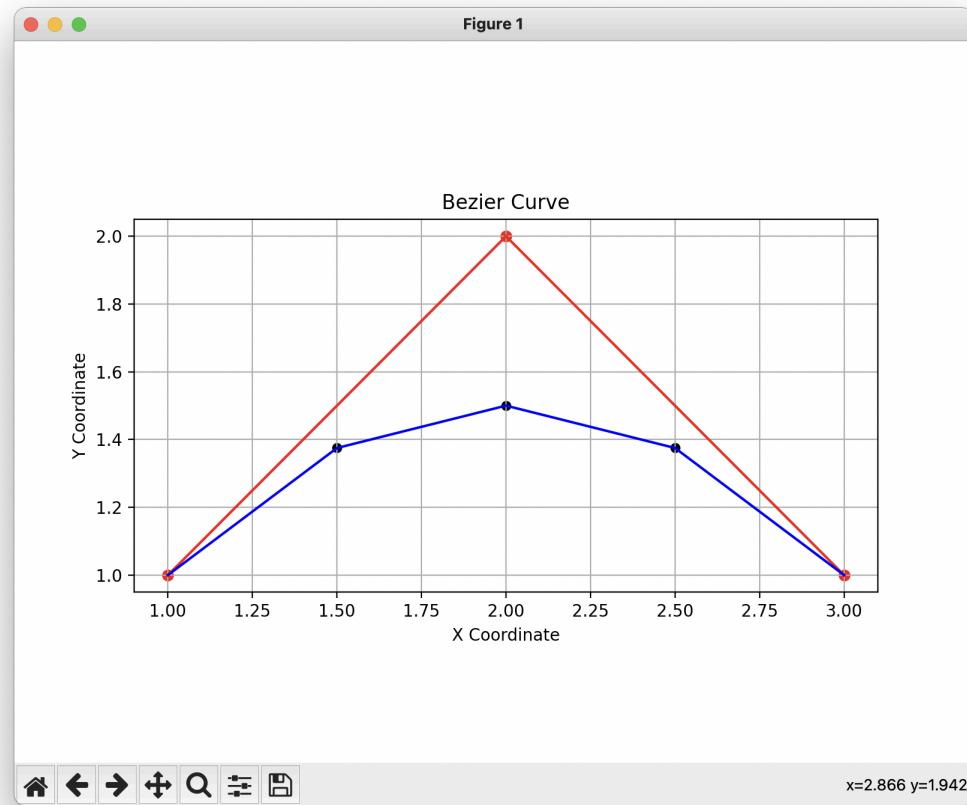
Jumlah iterasi : 2
Titik 1 : 1,1
Titik 2 : 2,2
Titik 3 : 3,1

Pilihan metode :
1. Divide and Conquer
2. Brute-Force

Masukkan pilihan anda : 2
Waktu eksekusi program : 0.00020623207092285156 seconds

=====
Terima kasih
=====
+ .~o~. +
(base) abi@Fabians-MacBook-Air src %

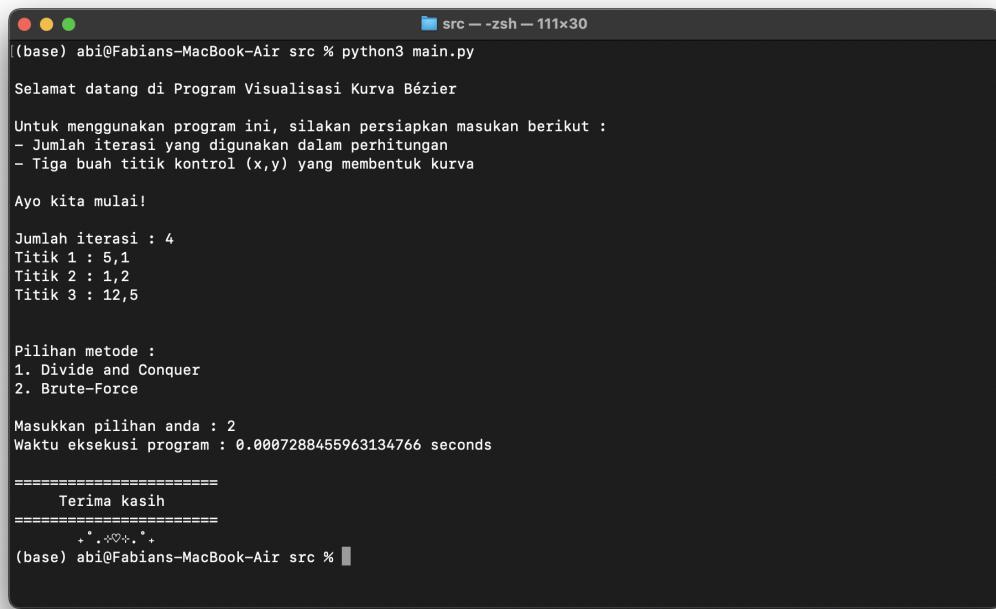
```



b. Percobaan 2

Data

Jumlah iterasi : 4
Titik Kontrol 1 : (5,1)
Titik Kontrol 1 : (1,2)
Titik Kontrol 1 : (12,5)



```
src -- zsh -- 111x30
(base) abi@Fabians-MacBook-Air src % python3 main.py
Selamat datang di Program Visualisasi Kurva Bézier

Untuk menggunakan program ini, silakan persiapkan masukan berikut :
- Jumlah iterasi yang digunakan dalam perhitungan
- Tiga buah titik kontrol (x,y) yang membentuk kurva

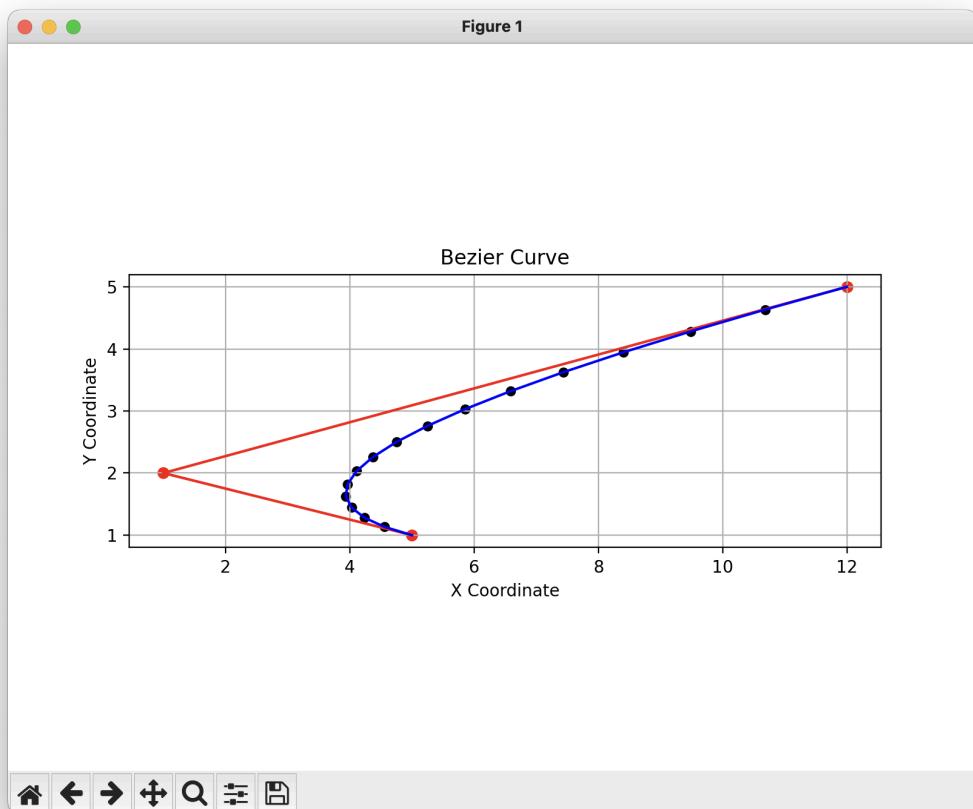
Ayo kita mulai!

Jumlah iterasi : 4
Titik 1 : 5,1
Titik 2 : 1,2
Titik 3 : 12,5

Pilihan metode :
1. Divide and Conquer
2. Brute-Force

Masukkan pilihan anda : 2
Waktu eksekusi program : 0.0007288455963134766 seconds

=====
Terima kasih
=====
+ .*^*.*+
(base) abi@Fabians-MacBook-Air src %
```



c. Percobaan 3

Data

Jumlah iterasi : 3

Titik Kontrol 1 : (3,11)

Titik Kontrol 1 : (2,2)

Titik Kontrol 1 : (0,0.5)

```
src -- zsh -- 111x30
(base) abi@Fabians-MacBook-Air src % python3 main.py
Selamat datang di Program Visualisasi Kurva Bézier

Untuk menggunakan program ini, silakan persiapkan masukan berikut :
- Jumlah iterasi yang digunakan dalam perhitungan
- Tiga buah titik kontrol (x,y) yang membentuk kurva

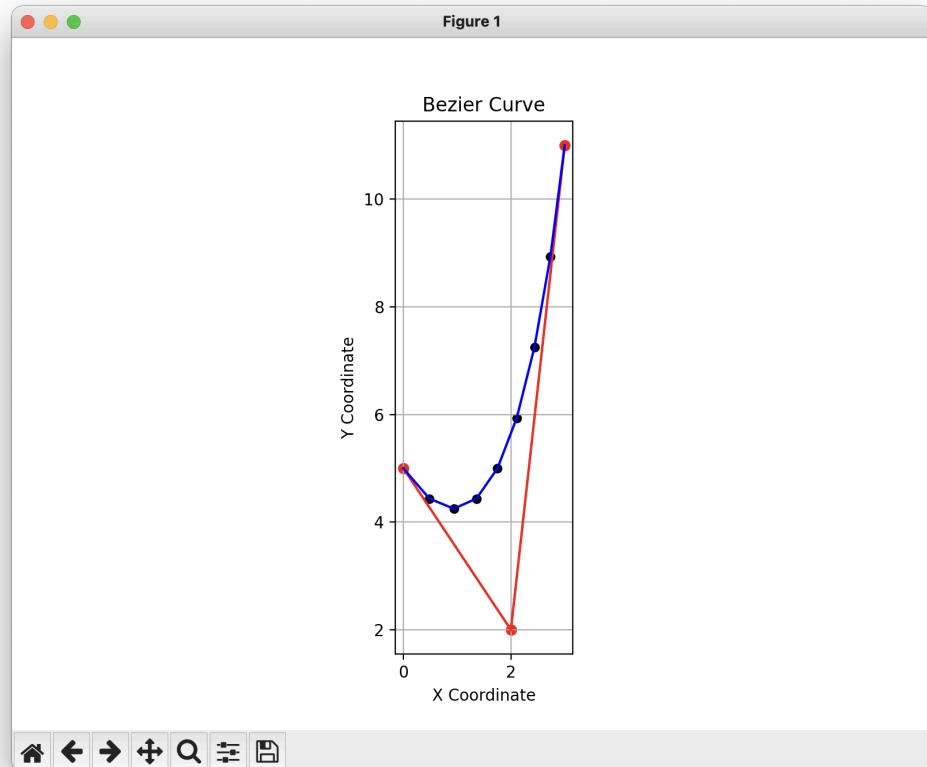
Ayo kita mulai!

Jumlah iterasi : 3
Titik 1 : 3,11
Titik 2 : 2,2
Titik 3 : 0,5

Pilihan metode :
1. Divide and Conquer
2. Brute-Force

Masukkan pilihan anda : 2
Waktu eksekusi program : 0.0005078315734863281 seconds

=====
Terima kasih
=====
+ . . . , .
(base) abi@Fabians-MacBook-Air src %
```



d. Percobaan 4

Data

Jumlah iterasi : 5

Titik Kontrol 1 : (2,1)

Titik Kontrol 1 : (5,2)

Titik Kontrol 1 : (0,0)

```
src -- zsh -- 111x30
(base) abi@Fabians-MacBook-Air src % python3 main.py
Selamat datang di Program Visualisasi Kurva Bézier

Untuk menggunakan program ini, silakan persiapkan masukan berikut :
- Jumlah iterasi yang digunakan dalam perhitungan
- Tiga buah titik kontrol (x,y) yang membentuk kurva

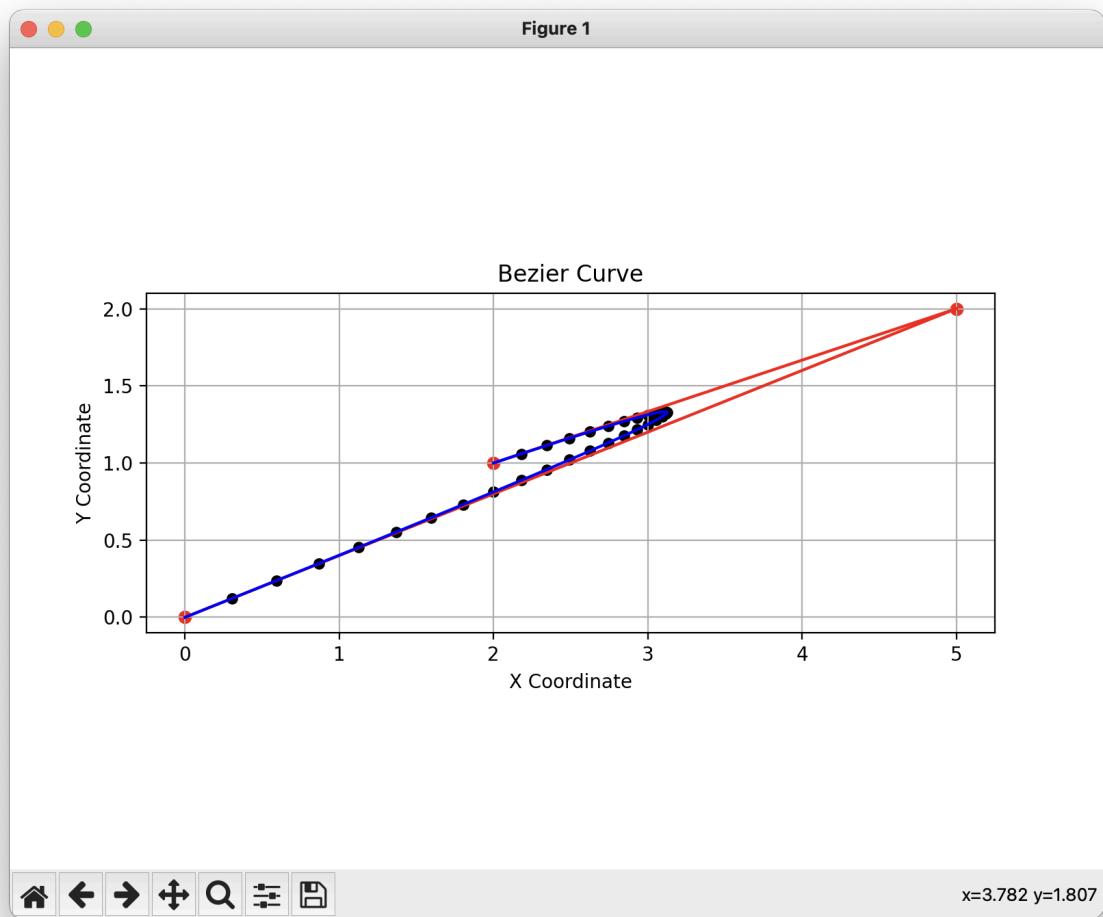
Ayo kita mulai!

Jumlah iterasi : 5
Titik 1 : 2,1
Titik 2 : 5,2
Titik 3 : 0,0

Pilihan metode :
1. Divide and Conquer
2. Brute-Force

Masukkan pilihan anda : 2
Waktu eksekusi program : 0.0009889602661132812 seconds

=====
Terima kasih
=====
+ .+*+.*+
(base) abi@Fabians-MacBook-Air src %
```



e. Percobaan 5

Data

Jumlah iterasi : 3

Titik Kontrol 1 : (0,0)

Titik Kontrol 1 : (2,1)

Titik Kontrol 1 : (10,7)

```
src -- zsh -- 111x30
(base) abi@Fabians-MacBook-Air src % python3 main.py
Selamat datang di Program Visualisasi Kurva Bézier

Untuk menggunakan program ini, silakan persiapkan masukan berikut :
- Jumlah iterasi yang digunakan dalam perhitungan
- Tiga buah titik kontrol (x,y) yang membentuk kurva

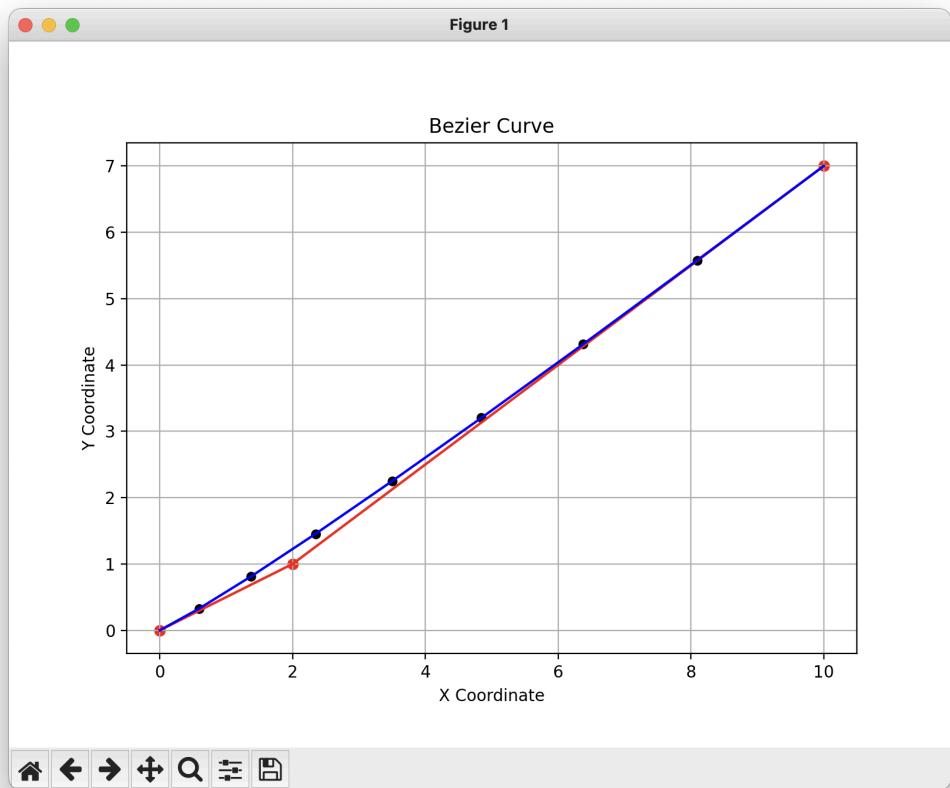
Ayo kita mulai!

Jumlah iterasi : 3
Titik 1 : 0,0
Titik 2 : 2,1
Titik 3 : 10,7

Pilihan metode :
1. Divide and Conquer
2. Brute-Force

Masukkan pilihan anda : 2
Waktu eksekusi program : 0.000331878662109375 seconds

=====
Terima kasih
=====
+ °.*°+.*+
(base) abi@Fabians-MacBook-Air src %
```



f. Percobaan 6

Data

Jumlah iterasi : 7

Titik Kontrol 1 : (-3,-2)

Titik Kontrol 1 : (5,7)

Titik Kontrol 1 : (8,3)

```
src -- zsh -- 111x30
(base) abi@Fabians-MacBook-Air src % python3 main.py
Selamat datang di Program Visualisasi Kurva Bézier

Untuk menggunakan program ini, silakan persiapkan masukan berikut :
- Jumlah iterasi yang digunakan dalam perhitungan
- Tiga buah titik kontrol (x,y) yang membentuk kurva

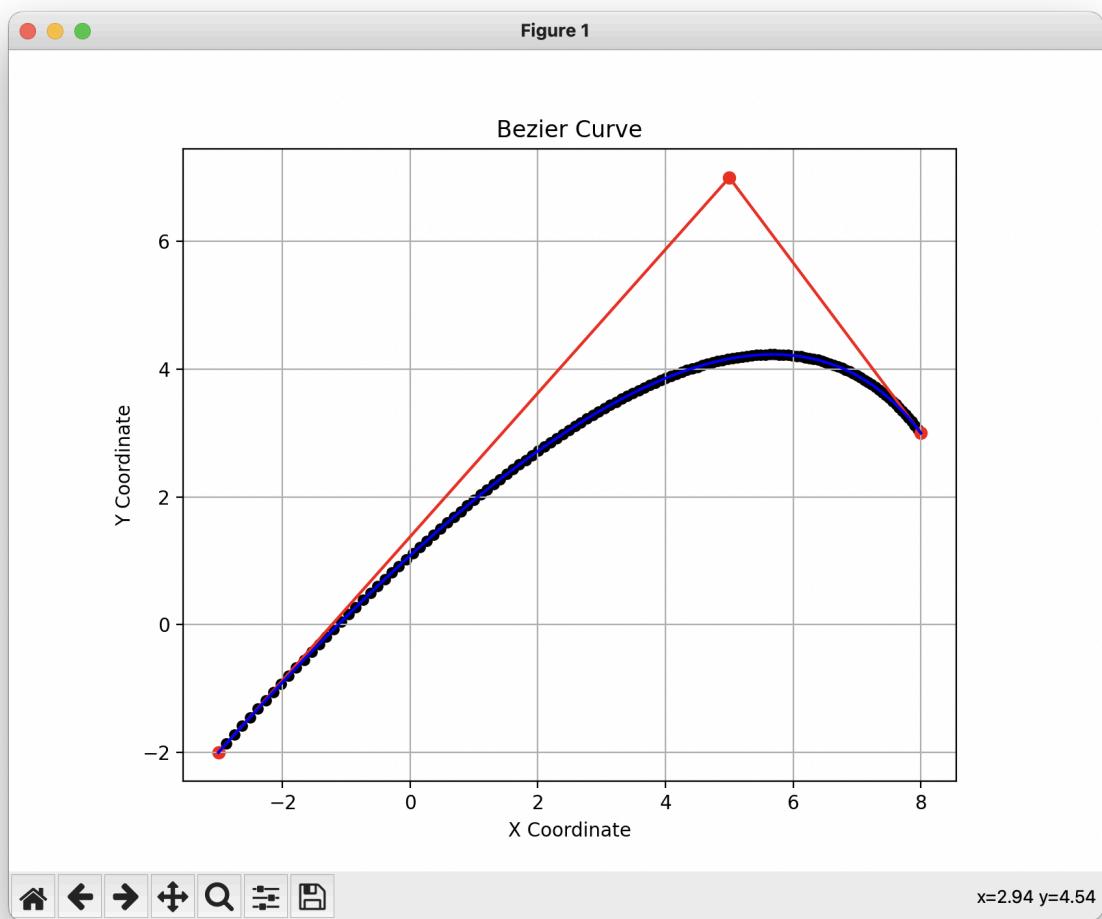
Ayo kita mulai!

Jumlah iterasi : 7
Titik 1 : -3,-2
Titik 2 : 5,7
Titik 3 : 8,3

Pilihan metode :
1. Divide and Conquer
2. Brute-Force

Masukkan pilihan anda : 2
Waktu eksekusi program : 0.004083156585693359 seconds

=====
Terima kasih
=====
+ .~^~. +
(base) abi@Fabians-MacBook-Air src %
```



E. Perbandingan Metode dalam Mencari Solusi

Untuk membandingkan algoritma mana yang lebih efisien dalam mencari solusi dari permasalahan ini sebenarnya harus dihitung kompleksitasnya. Namun mohon maaf saya belum mengetahui bagaimana cara menghitung kompleksitas. Dari *program testing* didapatkan bahwa algoritma *divide and conquer* cenderung lebih cepat dalam menemukan solusi. Oleh karena itu, algoritma *divide and conquer* lebih tepat untuk digunakan dalam menyelesaikan masalah ini.

F. Lampiran

Pranala Github : https://github.com/fabianradenta/Tucil2_13522105