LAPORAN TUGAS KECIL II
IF2211 STRATEGI ALGORITMA

MEMBANGUN KURVA BEZIER DENGAN ALGORITMA TITIK TENGAH BERBASIS DIVIDE AND CONQUER

Disusun oleh :
Eduardus Alvito Kristiadi (13522004)
Francesco Michael Kusuma (13522038)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023/2024

1. Pendahuluan

Kurva Bézier adalah kurva halus yang sering digunakan dalam desain grafis, animasi, dan manufaktur. Kurva ini dibuat dengan menghubungkan beberapa titik kontrol, yang menentukan bentuk dan arah kurva. Cara membuatnya cukup mudah, yaitu dengan menentukan titik-titik kontrol dan menghubungkannya dengan kurva. Kurva Bézier memiliki banyak kegunaan dalam kehidupan nyata, seperti pen tool, animasi yang halus dan realistis, membuat desain produk yang kompleks dan presisi, dan membuat font yang indah dan unik. Keuntungan menggunakan kurva Bézier adalah kurva ini mudah diubah dan dimanipulasi, sehingga dapat menghasilkan desain yang presisi dan sesuai dengan kebutuhan.

Sebuah kurva Bézier didefinisikan oleh satu set titik kontrol P0 sampai Pn , dengan n disebut order (n = 1 untuk linier, n = 2 untuk kuadrat, dan seterusnya). Titik kontrol pertama dan terakhir selalu menjadi ujung dari kurva, tetapi titik kontrol antara (jika ada) umumnya tidak terletak pada kurva. Pada gambar 1 diatas, titik kontrol pertama adalah P0, sedangkan titik kontrol terakhir adalah P3. Titik kontrol P1 dan P2 disebut sebagai titik kontrol antara yang tidak terletak dalam kurva yang terbentuk.

Mengulas lebih jauh mengenai bagaimana sebuah kurva Bézier bisa terbentuk, misalkan diberikan dua buah titik P0 dan P1 yang menjadi titik kontrol, maka kurva Bézier yang terbentuk adalah sebuah garis lurus antara dua titik. Kurva ini disebut dengan kurva Bézier linier. Misalkan terdapat sebuah titik Q0 yang berada pada garis yang dibentuk oleh P0 dan P1, maka posisinya dapat dinyatakan dengan persamaan parametrik berikut.

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

dengan t dalam fungsi kurva Bézier linier menggambarkan seberapa jauh B(t) dari P0 ke P1. Misalnya ketika t = 0.25, maka B(t) adalah seperempat jalan dari titik P0 ke P1. sehingga seluruh rentang variasi nilai t dari 0 hingga 1 akan membuat persamaan B(t) membentuk sebuah garis lurus dari P0 ke P1.

Misalkan selain dua titik sebelumnya ditambahkan sebuah titik baru, sebut saja P2, dengan P0 dan P2 sebagai titik kontrol awal dan akhir, dan P1 menjadi titik kontrol antara. Dengan menyatakan titik Q1 terletak diantara garis yang menghubungkan P1 dan P2, dan membentuk kurva Bézier linier yang berbeda dengan kurva letak Q0 berada, maka dapat dinyatakan sebuah titik baru, R0 yang berada diantara garis yang menghubungkan Q0 dan Q1 yang bergerak membentuk kurva Bézier kuadratik terhadap titik P0 dan P2. Berikut adalah uraian persamaannya.

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

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

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

dengan melakukan substitusi nilai Q0 dan Q1, maka diperoleh persamaan sebagai berikut.

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

Proses ini dapat juga diaplikasikan untuk jumlah titik yang lebih dari tiga, misalnya empat titik akan menghasilkan kurva Bézier kubik, lima titik akan menghasilkan kurva Bézier kuartik, dan seterusnya. Berikut adalah persamaan kurva Bézier kubik dan kuartik dengan menggunakan prosedur yang sama dengan yang sebelumnya.

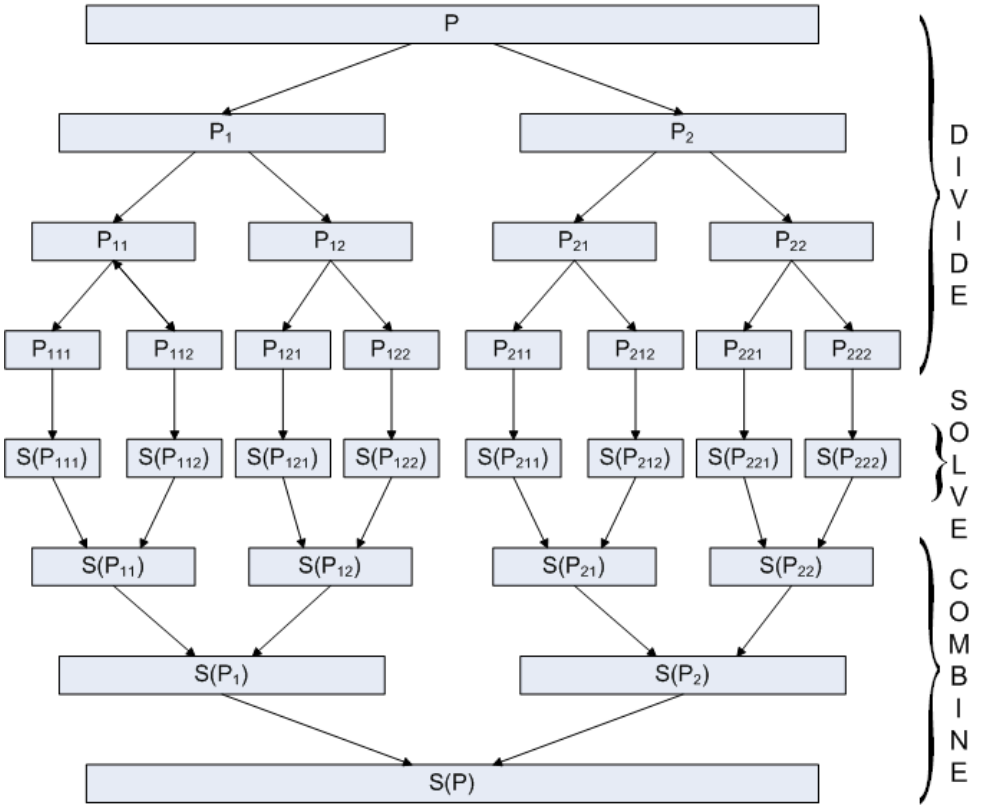
$$S_0 = B(t) = (1-t)^3 P_0 + 3(1-t)^2 tP_1 + 3(1-t)t^2 P_2 + \; t^3 P_3, \qquad t \in [0,1]$$

$$T_0 = B(t) = (1-t)^4 P_0 + 4(1-t)^3 tP_1 + 6(1-t)^2 t^2 P_2 + 4(1-t)t^3 P_3 + \; t^4 P_4, \qquad t \in [0,1]$$

Tentu saja persamaan yang terbentuk sangat panjang dan akan semakin rumit seiring bertambahnya titik. Oleh sebab itu, dalam rangka melakukan efisiensi pembuatan kurva Bézier yang sangat berguna ini, maka Anda diminta untuk mengimplementasikan pembuatan kurva Bézier dengan algoritma titik tengah berbasis divide and conquer.

2. Algoritma Divide and Conquer

Algoritma divide and conquer sendiri berasal dari dua kata yaitu *divide* dan *conquer*. Divide berarti membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya setiap upa-persoalan berukuran hampir sama). Conquer berarti menyelesaikan (solve) masing-masing upa-persoalan (secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar). Divide dan Conquer lalu digabungkan (combined) dengan solusi masing-masing upa-persoalan sehingga membentuk solusi persoalan semula.



Keterangan:
P = persoalan
S = solusi

Dalam algoritma divide and conquer, setiap upa persoalan memiliki karakteristik yang sama persis (*the same type*) dengan karakteristik persoalan semula namun berukuran lebih kecil sehingga metode Divide and Conquer lebih natural diungkapkan dalam skema rekursif.

Kompleksitas algoritma divide and conquer:

$$T(n) = \begin{cases} g(n) & , n \leq n_0 \\ T(n_1) + T(n_2) \ldots + T(n_r) + f(n) & , n > n_0 \end{cases}$$

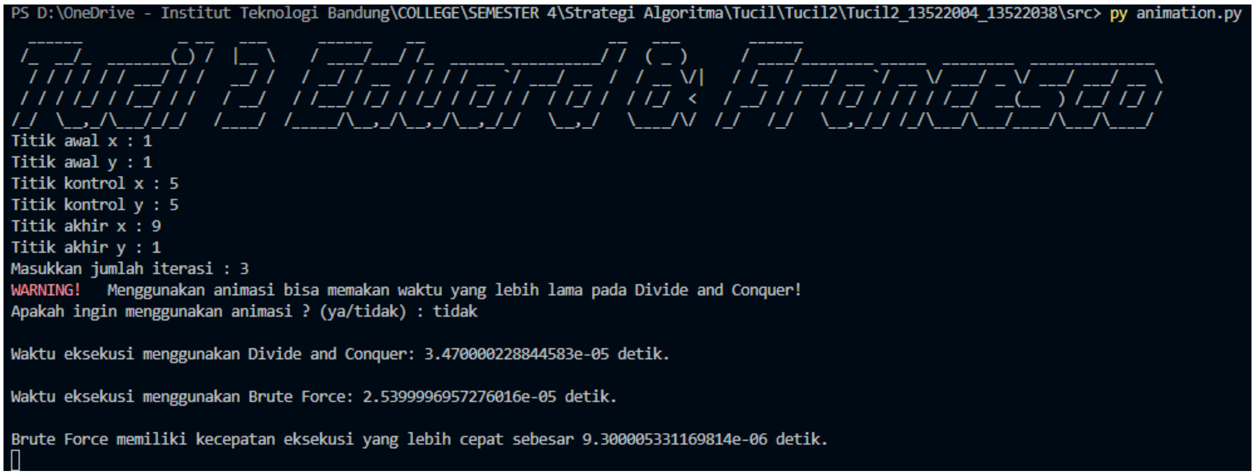
- T(n) : kompleksitas waktu penyelesaian persoalan P yang berukuran n
- g(n) : kompleksitas waktu untuk SOLVE jika n sudah berukuran kecil
- *Tn*1+*Tn*2…+*Tnr* : kompleksitas waktu untuk memproses setiap upa-persoalan

- f(n) : kompleksitas waktu untuk COMBINE solusi dari masing-masing upa-persoalan
- Tahap DIVIDE dapat dilakukan dalam O(1), sehingga tidak dimasukkan ke dalam formula

3. Source Code & Testing

Pada saat program dijalankan, apabila pengguna memilih titik yang range nya cukup jauh dan di luar dari range koordinat grafik dalam matplotlib, maka pengguna dapat menekan tombol pan  di bagian bawah pop-up grafik dan mengarahkannya pada koordinat yang sesuai agar kurva bezier dapat terlihat dengan jelas.

- **Test 1:**

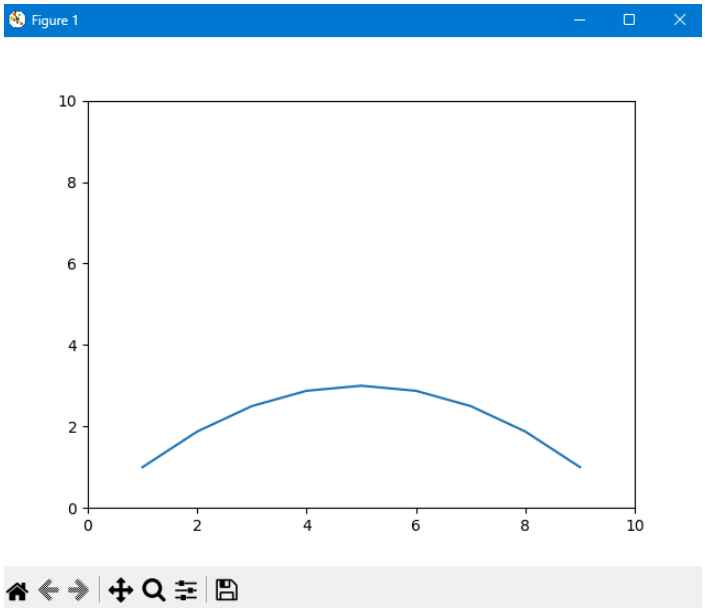


```
PS D:\OneDrive - Institut Teknologi Bandung\COLLEGE\SEMESTER 4\Strategi Algoritma\Tucil\Tucil2\Tucil2_13522004_13522038\src> py animation.py
Titik awal x : 1
Titik awal y : 1
Titik kontrol x : 5
Titik kontrol y : 5
Titik akhir x : 9
Titik akhir y : 1
Masukkan jumlah iterasi : 3
WARNING!   Menggunakan animasi bisa memakan waktu yang lebih lama pada Divide and Conquer!
Apakah ingin menggunakan animasi ? (ya/tidak) : tidak

Waktu eksekusi menggunakan Divide and Conquer: 3.470000228844583e-05 detik.

Waktu eksekusi menggunakan Brute Force: 2.5399996957276016e-05 detik.

Brute Force memiliki kecepatan eksekusi yang lebih cepat sebesar 9.300005331169814e-06 detik.
```

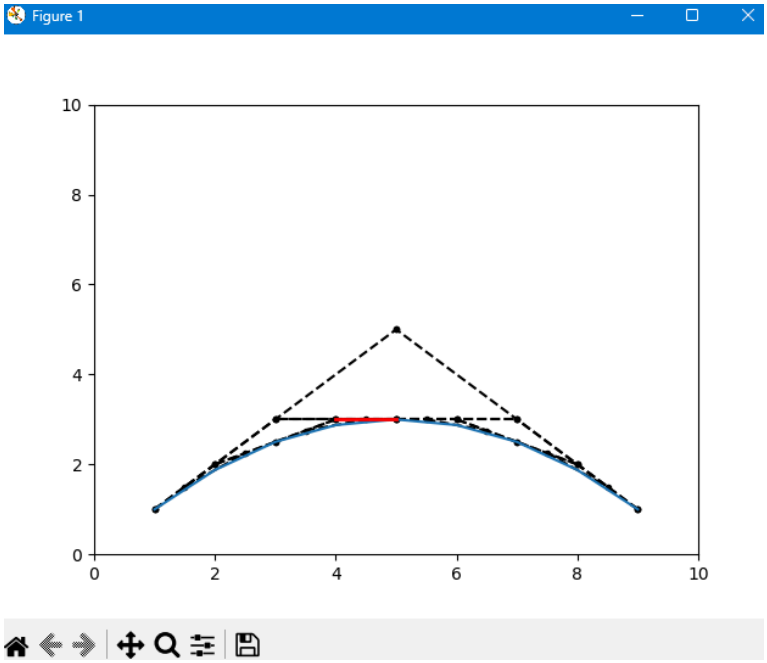



```
PS D:\OneDrive - Institut Teknologi Bandung\COLLEGE\SEMESTER 4\Strategi Algoritma\Tucil\Tucil2\Tucil2_13522004_13522038\src> py animation.py
Titik awal x : 1
Titik awal y : 1
Titik kontrol x : 5
Titik kontrol y : 5
Titik akhir x : 9
Titik akhir y : 1
Masukkan jumlah iterasi : 3
WARNING!   Menggunakan animasi bisa memakan waktu yang lebih lama pada Divide and Conquer!
Apakah ingin menggunakan animasi ? (ya/tidak) : ya

Waktu eksekusi menggunakan Divide and Conquer: 0.00856029999704333 detik.

Waktu eksekusi menggunakan Brute Force: 3.9200000173877925e-05 detik.

Brute Force memiliki kecepatan eksekusi yang lebih cepat sebesar 0.008521099996869452 detik.
```



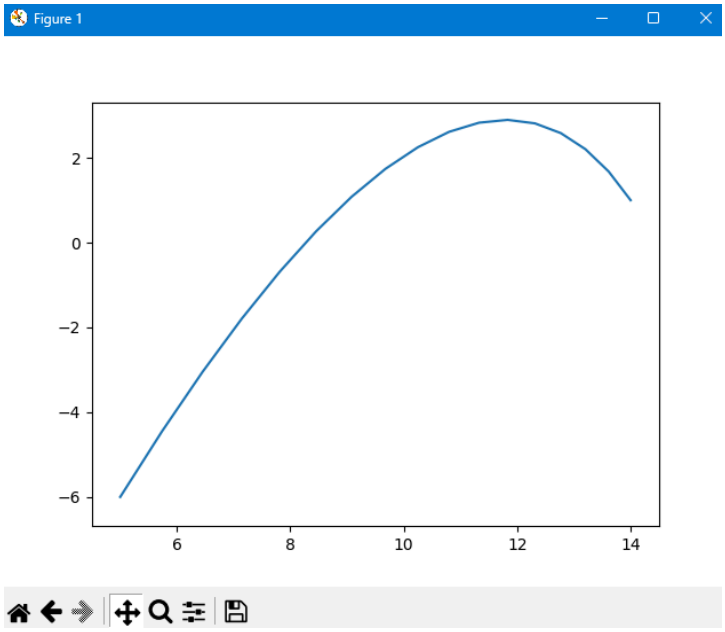
- **Test 2:**

```
PS D:\OneDrive - Institut Teknologi Bandung\COLLEGE\SEMESTER 4\Strategi Algoritma\Tucil\Tucil2\Tucil2_13522004_13522038\src> py animation.py
Titik awal x : 5
Titik awal y : -6
Titik kontrol x : 11
Titik kontrol y : 7
Titik akhir x : 14
Titik akhir y : 1
Masukkan jumlah iterasi : 4
WARNING!   Menggunakan animasi bisa memakan waktu yang lebih lama pada Divide and Conquer!
Apakah ingin menggunakan animasi ? (ya/tidak) : tidak

Waktu eksekusi menggunakan Divide and Conquer: 6.2099999992279336e-05 detik.

Waktu eksekusi menggunakan Brute Force: 4.5000000682193786e-05 detik.

Brute Force memiliki kecepatan eksekusi yang lebih cepat sebesar 1.7099999240599573e-05 detik.
```

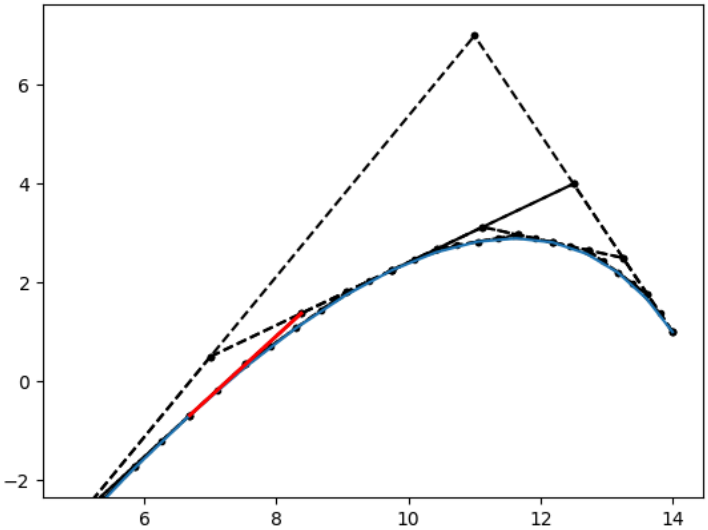



```
PS D:\OneDrive - Institut Teknologi Bandung\COLLEGE\SEMESTER 4\Strategi Algoritma\Tucil\Tucil2\Tucil2_13522004_13522038\src> py animation.py
Titik awal x : 3
Titik awal y : -6
Titik kontrol x : 11
Titik kontrol y : 7
Titik akhir x : 14
Titik akhir y : 1
Masukkan jumlah iterasi : 4
WARNING!   Menggunakan animasi bisa memakan waktu yang lebih lama pada Divide and Conquer!
Apakah ingin menggunakan animasi ? (ya/tidak) : ya

Waktu eksekusi menggunakan Divide and Conquer: 0.01659269999800017 detik.

Waktu eksekusi menggunakan Brute Force: 6.149999535409734e-05 detik.

Brute Force memiliki kecepatan eksekusi yang lebih cepat sebesar 0.016531200002646074 detik.
```



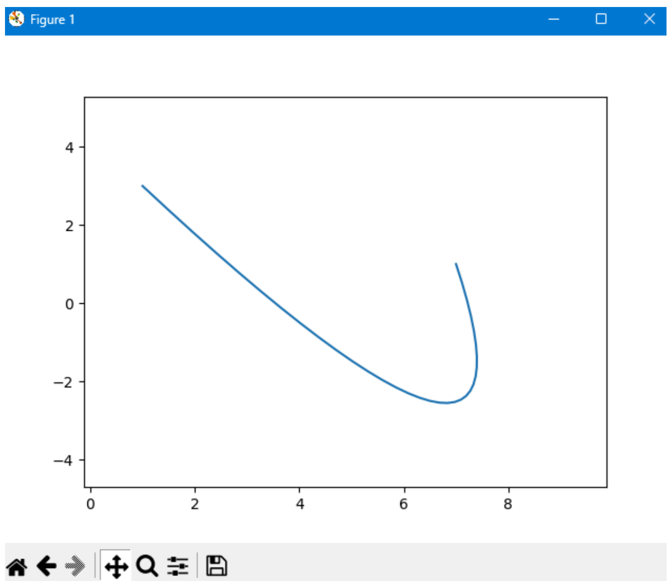
- **Test 3:**

```
PS D:\OneDrive - Institut Teknologi Bandung\COLLEGE\SEMESTER 4\Strategi Algoritma\Tucil\Tucil2\Tucil2_13522004_13522038\src> py animation.py
Titik awal x : 7
Titik awal y : 1
Titik kontrol x : 9
Titik kontrol y : -7
Titik akhir x : 1
Titik akhir y : 3
Masukkan jumlah iterasi : 5
WARNING!   Menggunakan animasi bisa memakan waktu yang lebih lama pada Divide and Conquer!
Apakah ingin menggunakan animasi ? (ya/tidak) : tidak

Waktu eksekusi menggunakan Divide and Conquer: 6.0699996538460255e-05 detik.

Waktu eksekusi menggunakan Brute Force: 6.710000161547214e-05 detik.

Divide and Conquer memiliki kecepatan eksekusi yang lebih cepat sebesar 6.400005077011883e-06 detik.
```

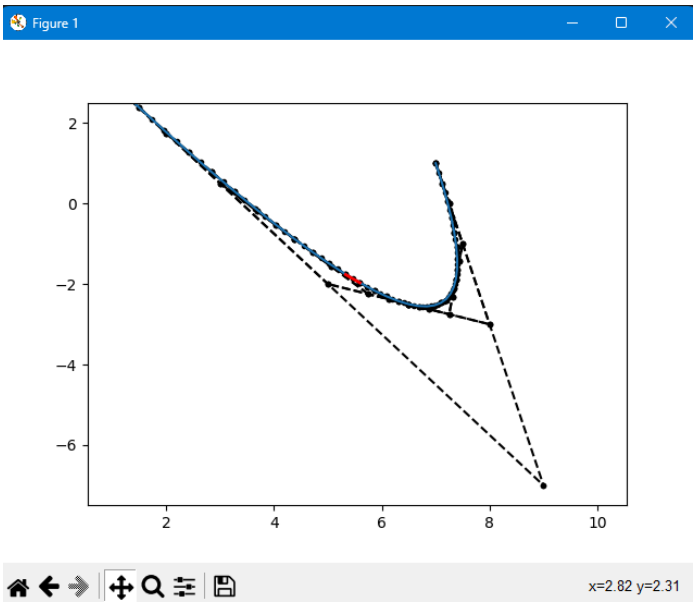



```
PS D:\OneDrive - Institut Teknologi Bandung\COLLEGE\SEMESTER 4\Strategi Algoritma\Tucil\Tucil2\Tucil2_13522004_13522038\src> py animation.py
Titik awal x : 7
Titik awal y : 1
Titik kontrol x : 9
Titik kontrol y : -7
Titik akhir x : 1
Titik akhir y : 3
Masukkan jumlah iterasi : 5
WARNING!   Menggunakan animasi bisa memakan waktu yang lebih lama pada Divide and Conquer!
Apakah ingin menggunakan animasi ? (ya/tidak) : ya

Waktu eksekusi menggunakan Divide and Conquer: 0.03425219999917317 detik.

Waktu eksekusi menggunakan Brute Force: 6.1399994592648e-05 detik.

Brute Force memiliki kecepatan eksekusi yang lebih cepat sebesar 0.034190800004580524 detik.
```



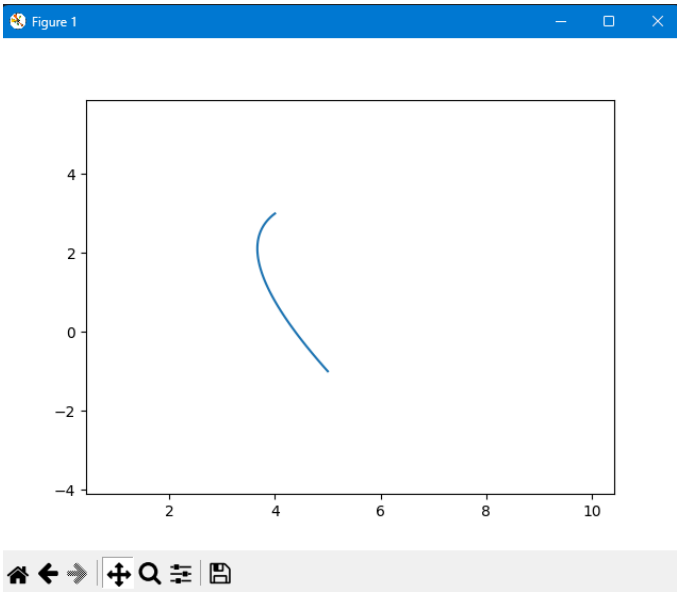
- **Test 4**

```
PS D:\OneDrive - Institut Teknologi Bandung\COLLEGE\SEMESTER 4\Strategi Algoritma\Tucil\Tucil2\Tucil2_13522004_13522038\src> py animation.py
Titik awal x : 5
Titik awal y : -1
Titik kontrol x : 3
Titik kontrol y : 2
Titik akhir x : 4
Titik akhir y : 3
Masukkan jumlah iterasi : 6
WARNING!   Menggunakan animasi bisa memakan waktu yang lebih lama pada Divide and Conquer!
Apakah ingin menggunakan animasi ? (ya/tidak) : tidakk
Apakah ingin menggunakan animasi ? (ya/tidak) : tidak

Waktu eksekusi menggunakan Divide and Conquer: 9.180000051856041e-05 detik.

Waktu eksekusi menggunakan Brute Force: 0.00010199999815085903 detik.

Divide and Conquer memiliki kecepatan eksekusi yang lebih cepat sebesar 1.0199997632298619e-05 detik.
```

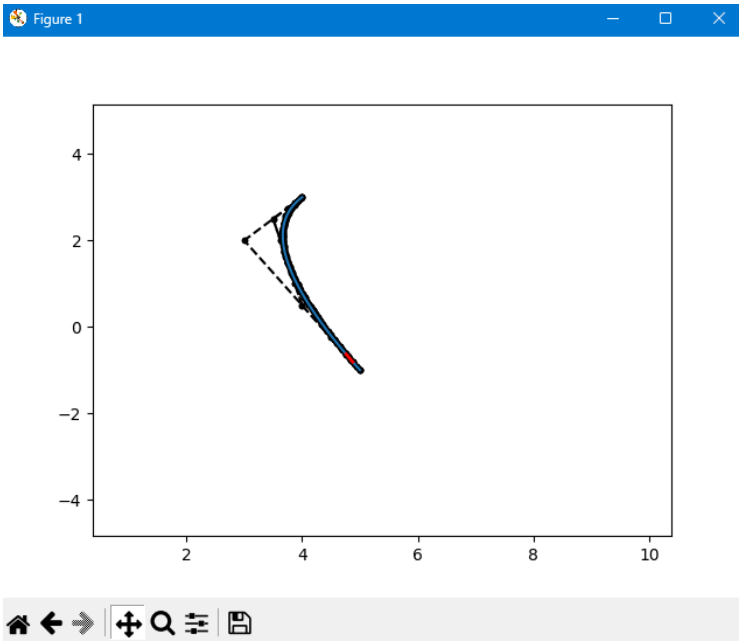



```
PS D:\OneDrive - Institut Teknologi Bandung\COLLEGE\SEMESTER 4\Strategi Algoritma\Tucil\Tucil2\Tucil2_13522004_13522038\src> py animation.py
Titik awal x : 5
Titik awal y : -1
Titik kontrol x : 3
Titik kontrol y : 2
Titik akhir x : 4
Titik akhir y : 3
Masukkan jumlah iterasi : 6
WARNING!   Menggunakan animasi bisa memakan waktu yang lebih lama pada Divide and Conquer!
Apakah ingin menggunakan animasi ? (ya/tidak) : ya

Waktu eksekusi menggunakan Divide and Conquer: 0.06602279999788152 detik.

Waktu eksekusi menggunakan Brute Force: 9.989999671233818e-05 detik.

Brute Force memiliki kecepatan eksekusi yang lebih cepat sebesar 0.06592290000116918 detik.
```



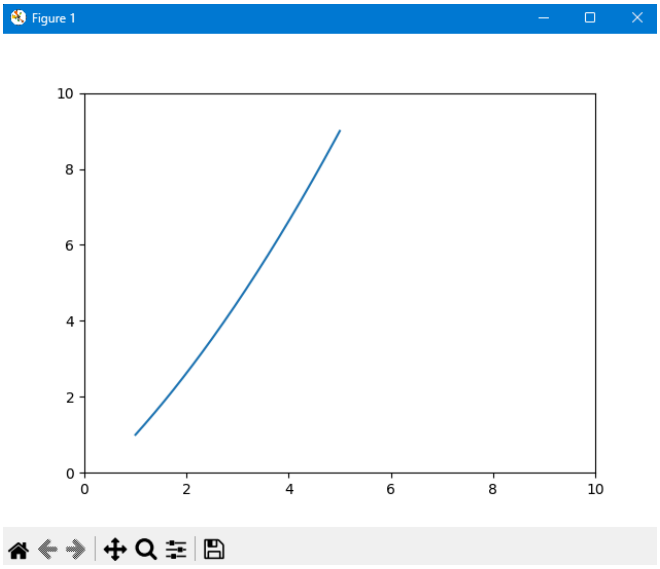
- **Test 5**

```
PS D:\OneDrive - Institut Teknologi Bandung\COLLEGE\SEMESTER 4\Strategi Algoritma\Tucil\Tucil2\Tucil2_13522004_13522038\src> py animation.py
Titik awal x : 1
Titik awal y : 1
Titik kontrol x : 3
Titik kontrol y : 4
Titik akhir x : 5
Titik akhir y : 9
Masukkan jumlah iterasi : 7
WARNING!   Menggunakan animasi bisa memakan waktu yang lebih lama pada Divide and Conquer!
Apakah ingin menggunakan animasi ? (ya/tidak) : tidak

Waktu eksekusi menggunakan Divide and Conquer: 0.0001789000162879005 detik.

Waktu eksekusi menggunakan Brute Force: 0.0010937000042758882 detik.

Divide and Conquer memiliki kecepatan eksekusi yang lebih cepat sebesar 0.0009148000026470982 detik.
```

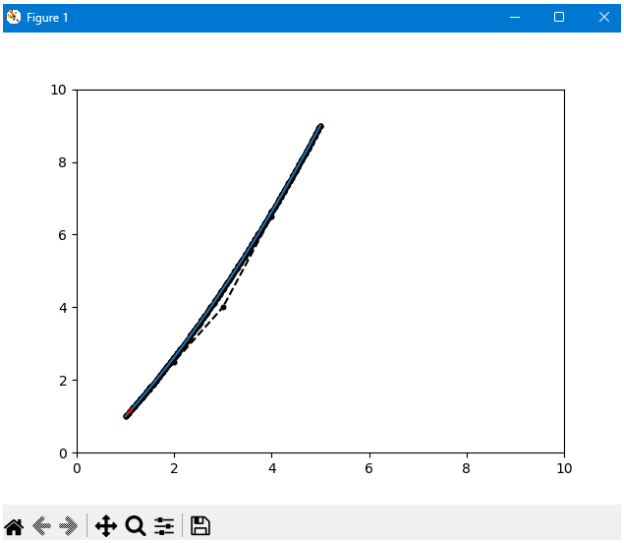



```
PS D:\OneDrive - Institut Teknologi Bandung\COLLEGE\SEMESTER 4\Strategi Algoritma\Tucil\Tucil2\Tucil2_13522004_13522038\src> py animation.py
Titik awal x : 1
Titik awal y : 1
Titik kontrol x : 3
Titik kontrol y : 4
Titik akhir x : 5
Titik akhir y : 9
Masukkan jumlah iterasi : 7
WARNING!   Menggunakan animasi bisa memakan waktu yang lebih lama pada Divide and Conquer!
Apakah ingin menggunakan animasi ? (ya/tidak) : ya

Waktu eksekusi menggunakan Divide and Conquer: 0.13649440000153845 detik.

Waktu eksekusi menggunakan Brute Force: 0.0001836000103712082 detik.

Brute Force memiliki kecepatan eksekusi yang lebih cepat sebesar 0.13631080000050133 detik.
```



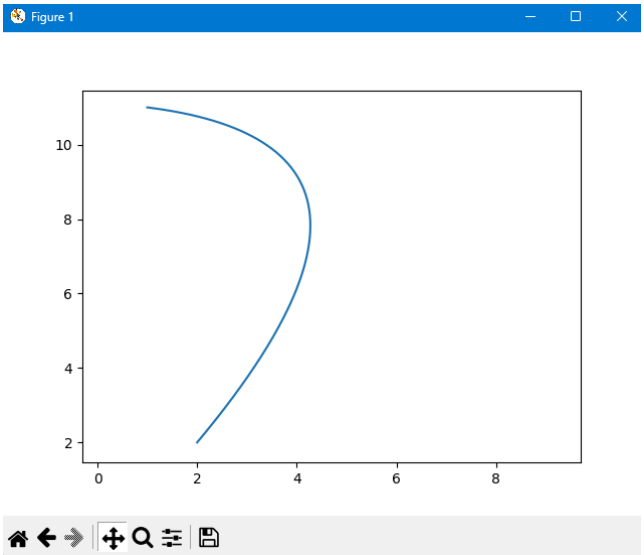
- **Test 6**

```
PS D:\OneDrive - Institut Teknologi Bandung\COLLEGE\SEMESTER 4\Strategi Algoritma\Tucil\Tucil2\Tucil2_13522004_13522038\src> py animation.py
Titik awal x : 2
Titik awal y : 2
Titik kontrol x : 7
Titik kontrol y : 10
Titik akhir x : 1
Titik akhir y : 11
Masukkan jumlah iterasi : 8
WARNING!   Menggunakan animasi bisa memakan waktu yang lebih lama pada Divide and Conquer!
Apakah ingin menggunakan animasi ? (ya/tidak) : tidak

Waktu eksekusi menggunakan Divide and Conquer: 0.0005376000044634566 detik.

Waktu eksekusi menggunakan Brute Force: 0.00037180000072112307 detik.

Brute Force memiliki kecepatan eksekusi yang lebih cepat sebesar 0.0001658000037423335 detik.
```

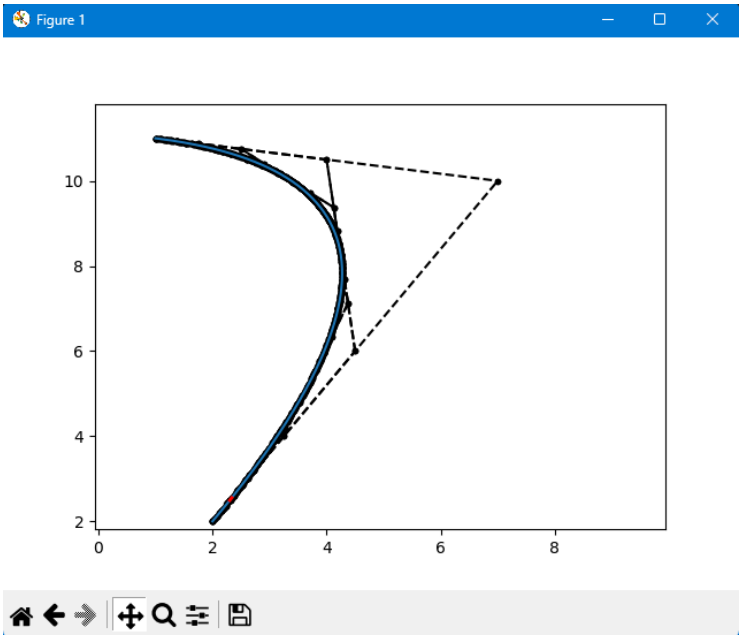


```
PS D:\OneDrive - Institut Teknologi Bandung\COLLEGE\SEMESTER 4\Strategi Algoritma\Tucil\Tucil2\Tucil2_13522004_13522038\src> py animation.py
Titik awal x : 2
Titik awal y : 2
Titik kontrol x : 7
Titik kontrol y : 10
Titik akhir x : 1
Titik akhir y : 11
Masukkan jumlah iterasi : 8
WARNING!   Menggunakan animasi bisa memakan waktu yang lebih lama pada Divide and Conquer!
Apakah ingin menggunakan animasi ? (ya/tidak) : ya

Waktu eksekusi menggunakan Divide and Conquer: 0.25913540000328794 detik.

Waktu eksekusi menggunakan Brute Force: 0.00036299999919719994 detik.

Brute Force memiliki kecepatan eksekusi yang lebih cepat sebesar 0.25877240000409074 detik.
```



4. Analisis

a. Algoritma Bruteforce dalam Membangun Kurva Bezier

Pada metode brute force yang diimplementasikan dalam fungsi bruteForce, kurva dihitung secara langsung menggunakan rumus. Metode ini mengiterasi melalui berbagai nilai t dari 0 sampai 1, dengan langkah sebesar 1/(2^iterasi), di mana iterasi adalah kedalaman rekursi yang ditentukan oleh pengguna. Untuk setiap nilai t, posisi titik pada kurva dihitung menggunakan formula Bezier kuadratik:

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

Di mana P0 adalah titik awal, P1 adalah titik kontrol, P2 adalah titik akhir dan t adalah parameter yang berjalan dari 0 ke 1. Nilai $B(t)$ ini kemudian digunakan untuk menentukan posisi x dan y dari titik pada kurva dan disimpan dalam array xBrute dan yBrute.

Keuntungan dari pendekatan brute force adalah kesederhanaannya; sangat mudah untuk diimplementasikan dan dimengerti. Namun, kelemahannya adalah efisiensinya. Metode ini bisa sangat lambat, terutama untuk nilai iterasi yang tinggi, karena jumlah perhitungan yang dibutuhkan meningkat secara eksponensial.

Implementasi dalam Program

Dalam program tersebut, pengguna diminta untuk memasukkan titik awal, titik kontrol, dan titik akhir kurva Bezier, serta jumlah iterasi yang diinginkan. Fungsi bruteForce kemudian dipanggil dengan parameter ini, dan kurva dihitung menggunakan metode brute force yang dijelaskan di atas.

Setelah perhitungan selesai, waktu yang dibutuhkan untuk menjalankan algoritma brute force dihitung dan ditampilkan. Logic untuk menghitung dan membandingkan kedua metode telah diimplementasikan dengan baik.

Secara keseluruhan, pendekatan brute force untuk menghitung kurva Bezier adalah metode yang berguna untuk memahami dasar-dasar bagaimana kurva ini dihitung, meskipun dalam prakteknya, metode yang lebih efisien seringkali diperlukan untuk aplikasi yang lebih kompleks dan perhitungan yang rumit.

b. Algoritma Divide and Conquer dalam Membangun Kurva Bezier

Pemodelan algoritma Divide and Conquer dalam membangun kurva Bezier adalah sebagai berikut.

- Dekomposisi permasalahan: Algoritma memulai dengan satu set titik awal, titik kontrol, dan titik akhir. Pada setiap iterasi, algoritma menghitung titik-titik tengah (midpoints) antara setiap pasangan titik berturut-turut dan menggambar garis antara mereka. Ini secara efektif membagi kurva menjadi bagian-bagian yang lebih kecil.
- Pemanggilan Rekursif: Dengan setiap iterasi, proses pemisahan diulang pada setiap segmen yang baru dibentuk, hingga batas iterasi yang ditentukan tercapai. Pada tahap ini, titik-titik yang dihasilkan mendekati kurva Bezier yang sebenarnya.
- Kombinasi Hasil: Setelah batas iterasi tercapai, titik-titik yang dihasilkan (yang kini cukup dekat dengan kurva Bezier yang sebenarnya) disimpan. Proses ini menghasilkan set titik yang, ketika digabungkan, membentuk aproksimasi kurva Bezier.
- Plotting: Titik-titik yang dihasilkan kemudian diplot untuk menampilkan kurva Bezier.

Implementasi dalam Program

- Input Pengguna: Program meminta pengguna memasukkan titik awal, titik kontrol (dapat lebih dari satu), dan titik akhir kurva Bezier, serta jumlah iterasi yang diinginkan.
- Inisialisasi: Array x dan y diisi dengan koordinat x dan y dari titik-titik ini, sedangkan ansX dan ansY akan menyimpan titik-titik hasil dari algoritma DnC.
- Proses Divide and Conquer: Fungsi dncBezier dijalankan dengan titik-titik input, dimulai dengan iterasi 0. Fungsi ini mengimplementasikan logika DnC:
    - Jika jumlah titik saat ini sama dengan 1, titik tersebut ditambahkan ke hasil akhir.

    - Jika tidak, fungsi menghitung titik-titik tengah antara setiap pasangan titik berturut-turut dan melakukan pemanggilan rekursif dengan set titik baru ini.
    - Proses ini diulang hingga jumlah iterasi yang diinginkan tercapai.
- Penggabungan dan Plotting: Setelah semua iterasi selesai, ansX dan ansY mengandung titik-titik yang mendekati kurva Bezier. Titik-titik ini kemudian diplot untuk menampilkan kurva.

c. Perbandingan Solusi Brute Force dengan Divide and Conquer

- **Kompleksitas Algoritma**
    a. Divide and Conquer (DnC)

    Pendekatan DnC mengurai masalah menjadi upa-masalah yang lebih kecil, menyelesaikannya secara rekursif, kemudian menggabungkan hasilnya. Kompleksitas waktu algoritma ini tergantung pada jumlah iterasi dan cara algoritma membagi masalah. Dalam kasus kurva Bezier, setiap pemanggilan rekursif membagi kurva menjadi dua bagian, menyelesaikan masing-masing secara rekursif. Ini berarti jumlah pemanggilan rekursif dan karenanya, kompleksitas waktu, meningkat secara eksponensial dengan jumlah iterasi. Secara teoretis, kompleksitasnya adalah O(2^n), di mana n adalah jumlah iterasi. Namun, karena setiap iterasi hanya menghitung beberapa titik tengah dan membagi kurva lebih lanjut, beban kerja per iterasi tidak terlalu besar.

    b. Brute Force

    Pendekatan brute force secara langsung menghitung nilai-nilai pada kurva menggunakan definisi matematis kurva Bezier, tanpa memecah masalah menjadi sub-masalah. Dalam implementasi ini, kompleksitas waktu bergantung pada jumlah titik yang dihitung, yang merupakan fungsi dari jumlah iterasi. Dengan setiap iterasi, jumlah titik yang dihitung meningkat eksponensial, mirip dengan DnC, karena t bergerak dalam inkremen dari 1/(2^n). Dengan demikian, kompleksitas waktu pendekatan ini juga eksponensial, yaitu O(2^n).
- Efisiensi Waktu: Dalam kasus pengujian tertentu, waktu eksekusi antara DnC dan brute force mungkin berbeda. DnC cenderung lebih efisien ketika kurva memiliki banyak detail atau ketika diperlukan tingkat presisi yang tinggi, karena dapat memusatkan upaya komputasi pada area kurva yang lebih kompleks. Di sisi lain, brute force memiliki kinerja yang konsisten terlepas dari detail kurva tetapi mungkin lebih lambat karena harus menghitung setiap titik secara eksplisit. Berdasarkan pengujian, dapat dilihat bahwa algoritma Divide and Conquer unggul dalam jumlah iterasi besar (sekitar 7 atau 8 ke atas), sedangkan untuk jumlah iterasi kecil sebanyak 7 atau 8 ke bawah, terlihat bahwa algoritma BruteForce lebih unggul.
- Kualitas Output: Kualitas kurva yang dihasilkan oleh kedua metode ini biasanya sangat mirip, tetapi DnC dapat memberikan hasil yang lebih halus dengan jumlah iterasi yang sama jika algoritma di-tweak untuk fokus pada segmen kurva yang lebih rumit.

d. Bonus (Visualisasi)

Dalam mengerjakan bonus, visualisasi garis-garis pembentuk kurva Bezier telah dibuat dan dijadikan melalui fungsi drawLines yang dipanggil di dalam findMidPoint setiap kali garis baru antara dua titik dibuat. Setiap garis ini diperoleh dari proses membagi segmen kurva menjadi dua bagian, yang secara visual menunjukkan bagaimana kurva Bezier dibentuk melalui proses rekursif. Pada saat program dijalankan, terdapat pilihan pada CLI. Jika pengguna memilih untuk menggunakan animasi (animation == "ya"), maka setiap kali findMidPoint dipanggil, garis yang terbentuk antara dua titik akan ditambahkan ke list lines dan diplot menggunakan drawLines. Jika tidak (animation == “tidak”), maka program hanya akan menampilkan kurva Bezier tanpa kehadiran garis-garis pembentuk kurva. Ini menghasilkan efek animasi dimana pengguna dapat melihat bagaimana garis-garis ini secara bertahap membentuk kurva Bezier secara visual.

5. Kesimpulan

Berdasarkan source code dan hasil pengujian, terbukti bahwa algoritma Divide and Conquer sangat berguna dalam merumuskan pembentukan kurva Bezier serta jauh lebih efisien untuk memecahkan permasalahan Kurva bezier dibandingkan dengan algoritma Brute Force, terutama untuk jumlah titik kontrol dan iterasi yang sangat besar.

6. Daftar Pustaka

https://informatika.stei.itb.ac.id/~rinaldi.munir/

7. Lampiran

7.1 Link Github Repository

Berikut adalah tautan repository untuk tugas kecil II IF2211 Strategi Algoritma ini.
https://github.com/Edvardesu/Tucil2_13522004_13522038

7.2 Checklist

No.	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.	☑	☐