

LAPORAN TUGAS KECIL 2 IF2211

STRATEGI ALGORITMA

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



Dosen Pengampu : Dr. Nur Ulfa Maulidevi, S.T, M.Sc.

Disusun oleh:

Maria Flora Renata S. (13522010)

Amalia Putri (13522042)

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG**

2024

KATA PENGANTAR

Puji syukur kita panjatkan ke hadirat Tuhan Yang Maha Kuasa, karena atas rahmat dan karunia-Nya, kami dapat menyelesaikan makalah ini dengan judul "Membangun Kurva Bézier dengan Algoritma Titik Tengah berbasis *Divide and Conquer*". Makalah ini disusun sebagai salah satu tugas mata kuliah Strategi Algoritma, dengan fokus pada penerapan konsep Algoritma *Divide and Conquer* pada pembuatan Kurva Bézier.

Penyusunan makalah ini tidak lepas dari bantuan berbagai pihak. Kami mengucapkan terima kasih kepada dosen beserta asisten pembimbing mata kuliah Strategi Algoritma yang telah memberikan bimbingan dan panduan selama proses pembuatan makalah ini. Semoga makalah ini dapat memberikan kontribusi positif dalam pemahaman dan pengembangan suatu aplikasi dari konsep Algoritma *Divide and Conquer*. Akhir kata, kami menyampaikan permohonan maaf atas segala keterbatasan dalam makalah ini, dan kami menerima dengan terbuka segala kritik dan saran yang bersifat membangun.

Bandung, Maret 2024,

Maria Flora Renata S. (13522010)

Amalia Putri (13522042)

DAFTAR ISI

KATA PENGANTAR.....	1
DAFTAR ISI.....	2
1. Pendahuluan.....	3
2. Analisis Implementasi Algoritma Divide and Conquer.....	4
3. Analisis Implementasi Algoritma Brute Force (Algoritma Pembandingan)....	5
4. Source Code Program Implementasi.....	6
5. Tangkapan Layar Hasil Keluaran dari Masukan.....	7
6. Hasil Analisis Perbandingan Kedua Solusi Algoritma.....	9
7. Implementasi Bonus.....	9
DAFTAR PUSTAKA.....	11
LAMPIRAN.....	12
Daftar Pembagian Tugas.....	12
Pengecekan Program.....	13
Repository.....	13

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 P_0 sampai P_n , 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. Titik kontrol pertama adalah P_0 , sedangkan titik kontrol terakhir adalah P_3 . Titik kontrol P_1 dan P_2 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 P_0 dan P_1 yang menjadi titik kontrol, maka kurva Bézier yang terbentuk adalah sebuah garis lurus antara dua titik. Kurva ini disebut dengan kurva Bézier linier. 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.

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. Analisis Implementasi Algoritma Divide and Conquer

Dalam mengimplementasikan algoritma *divide and conquer*, terdapat analisis dari implementasi algoritma sebagai berikut.

1) Konsep algoritma

Ada dua ide dasar yang digunakan dalam algoritma *divide and conquer* yang diimplementasikan. Pertama, pada suatu kurva Bézier, titik tengah dari kurva tersebut bisa didapatkan dengan mencari titik tengah dari setiap titik kontrol yang bersebelahan, lalu mencari titik tengah dari titik-titik tersebut sampai hanya terbentuk satu titik tengah. Ide kedua, sebagian dari sebuah kurva Bézier juga merupakan sebuah kurva Bézier.

Algoritma *divide and conquer* yang diimplementasikan memiliki tiga langkah, yaitu:

- (1) Mencari titik tengah kurva secara rekursif
- (2) Menghitung jarak antara titik tengah dengan titik kontrol di kiri dan kanan
- (3) Membagi kurva menjadi kurva kiri dan kurva kanan, lalu mengaplikasikan kembali ketiga langkah secara rekursif ke bagian yang jaraknya masih lebih besar dari jarak yang ditolerir.

2) Kompleksitas waktu algoritma

Pada langkah pertama, dilakukan pencarian titik tengah dari dua titik sebanyak $n - 1$ kali, di mana n adalah jumlah titik kontrol. Akan dihasilkan sebanyak $n - 1$ titik tengah, dan proses akan diulangi sampai hanya terdapat 3 titik tengah. Kompleksitas waktu pada langkah ini adalah $O(n)$.

Pada langkah kedua, dilakukan dua kali perhitungan jarak euclidean, sehingga kompleksitas waktu pada langkah ini adalah $O(1)$.

Pada langkah ketiga, dilakukan penambahan elemen ke dalam larik sebanyak n kali saat membagi kurva menjadi dua bagian, di mana n adalah jumlah titik kontrol awal kurva. Penambahan elemen ini dapat dilakukan dua kali, atau tidak dilakukan sama sekali untuk kasus basis. Oleh karena itu, kompleksitas waktu pada langkah ini adalah $O(n)$.

Ketiga langkah tersebut akan dilakukan secara rekursif pada kurva yang juga akan terus dibelah menjadi dua bagian. Oleh karena itu, kompleksitas waktu total dari algoritma ini adalah $O(\log n)$.

3. Analisis Implementasi Algoritma Brute Force (Algoritma Pembanding)

Dalam mengimplementasikan algoritma *brute force*, terdapat analisis dari implementasi algoritma sebagai berikut.

1) Konsep algoritma

(1) Basis rekursi

Untuk menghentikan proses rekursi, maka perlu dicek jumlah titik kontrol yang tersisa. Jika hanya tersisa 1 titik kontrol, rekursi akan berhenti dan mengembalikan titik kontrol paling terakhir.

(2) Konstruksi kurva dengan rekursi

Diketahui bahwa untuk menentukan titik kontrol (terbaru) didapatkan dari persamaan

$$\begin{aligned}Q_0 &= B(t) = (1 - t)P_0 + tP_1, & t \in [0, 1] \\Q_1 &= B(t) = (1 - t)P_1 + tP_2, & t \in [0, 1] \\R_0 &= B(t) = (1 - t)Q_0 + tQ_1, & t \in [0, 1]\end{aligned}$$

Gambar 1. Persamaan kurva Bézier kuadratik

Dari persamaan tersebut, dapat dilakukan proses iterasi sebanyak jumlah titik kontrol yang ada dan dilakukan rekursi dengan $t \in [0 \dots 1]$ hingga berhenti pada kondisi basis. Setiap rekursi akan menyebabkan jumlah titik kontrol semakin berkurang dan semakin mendekati titik pada kurva.

2) Kompleksitas waktu algoritma

Pada setiap langkah rekursi, dilakukan $n - 1$ interpolasi, di mana n adalah jumlah titik kontrol. Lalu, pada langkah berikutnya akan terdapat $n - 1$ titik hingga akhirnya mencapai 1 titik. Pembuatan kurva Bézier dengan algoritma *brute force* akan menghasilkan kompleksitas waktu (secara kasar) $O(n^2)$ dari penjumlahan deret $1 + 2 + \dots + n = \frac{n(n+1)}{2}$. Namun terdapat faktor linear, yakni m dan pada algoritma ini penulis menggunakan faktor linear $m = 100$. Dari gabungan kedua faktor tersebut, dapat disimpulkan bahwa kompleksitas

waktu algoritma untuk menggambar kurva Bezier menggunakan algoritma *brute force* adalah $O(m \cdot n^2)$.

4. Source Code Program Implementasi

Dalam mengimplementasikan algoritma *brute force*, terdapat analisis dari implementasi algoritma sebagai berikut.

1) Algoritma *divide and conquer*

```
1 def bezier_divide_and_conquer(ControlPoints, Tolerance):
2     #STEP 1: Calculate first final point
3     Results = []
4     NewControl, Mids = ToThree(ControlPoints)
5
6     #STEP 2: Check distance between result points
7     LeftDist = CheckDistance(ControlPoints[0], NewControl)
8     RightDist = CheckDistance(NewControl, ControlPoints[-1])
9
10    #STEP 3: Recursively divide curve until it's under tolerance
11    MidsLeft = []
12    MidsRight = []
13    if (LeftDist <= Tolerance):
14        Results.append(ControlPoints[0])
15        Results.append(NewControl)
16    elif (LeftDist > Tolerance):
17        Left = DivideCurveLeft(ControlPoints, NewControl, Mids)
18        NewControlsLeft, MidsLeft, MidsLeft2, MidsRight2 = bezier_divide_and_conquer(Left, Tolerance)
19        Results.extend(NewControlsLeft)
20
21    if (RightDist <= Tolerance):
22        #NewControl already added to Result, dont need to add it again
23        Results.append(ControlPoints[-1])
24
25    elif (RightDist > Tolerance):
26        Right = DivideCurveRight(ControlPoints, NewControl, Mids)
27        NewControlRight, MidsRight, MidsLeft2, MidsRight2 = bezier_divide_and_conquer(Right, Tolerance)
28        Results.extend(NewControlRight)
29    return Results, Mids, MidsLeft, MidsRight
30
31 def ToThree (ControlPoints):
32     if len(ControlPoints) == 3:
33         Mids = [MidPoint(ControlPoints[0], ControlPoints[1]), MidPoint(ControlPoints[1], ControlPoints[2])]
34         NewControl = MidPoint(Mids[0], Mids[1])
35         return NewControl, [Mids]
36     else:
37         Mids = []
38         TempMid1 = []
39         for i in range (len(ControlPoints)-1):
40             TempMid1.append(MidPoint(ControlPoints[i], ControlPoints[i+1]))
41         Mids.extend(TempMid1)
42         NewControl, TempMid = ToThree(TempMid1)
43         Mids.extend(TempMid)
44         return NewControl, Mids
45
46 def DivideCurveLeft (ControlPoints, NewControl, MiddlePoints):
47     Left = [ControlPoints[0]]
48     for i in range (len(MiddlePoints)):
49         Left.append(MiddlePoints[i][0])
50     Left.append(NewControl)
51     return Left
52
53 def DivideCurveRight (ControlPoints, NewControl, MiddlePoints):
54     Right = [NewControl]
55     for i in reversed(MiddlePoints):
56         Right.append(i[-1])
57     Right.append(ControlPoints[-1])
58     return Right
59
60 def MidPoint(Point1, Point2):
61     return ((Point1[0]+Point2[0]) * 0.5, (Point1[1]+Point2[1]) * 0.5)
62
63 def CheckDistance(Point1, Point2):
64     return ((abs(Point1[0]-Point2[0])**2 + abs(Point1[1]-Point2[1])**2)**0.5)
```

Gambar 2. Algoritma *divide and conquer*

2) Algoritma *brute force*

```

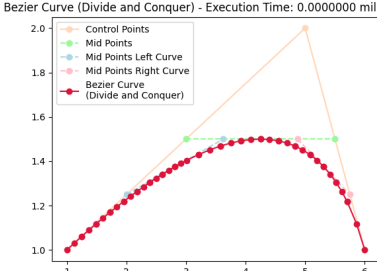
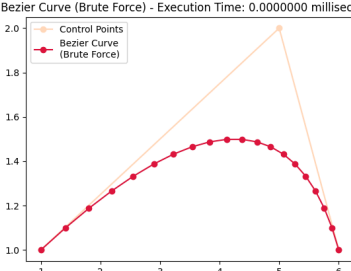
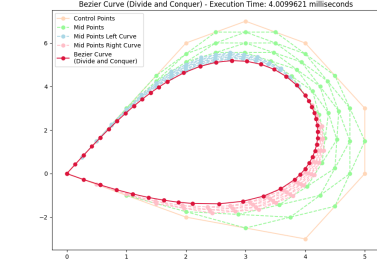
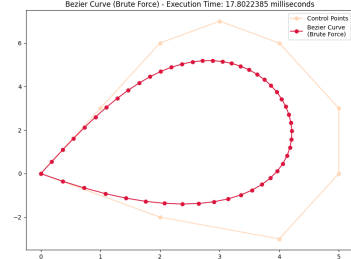
1 def bezier_brute_force(points, t):
2     if len(points) == 1:
3         return points[0]
4     else:
5         new_points = []
6         for i in range(len(points) - 1):
7             x = (1 - t) * points[i][0] + t * points[i + 1][0]
8             y = (1 - t) * points[i][1] + t * points[i + 1][1]
9             new_points.append((x, y))
10        return bezier_brute_force(new_points, t)

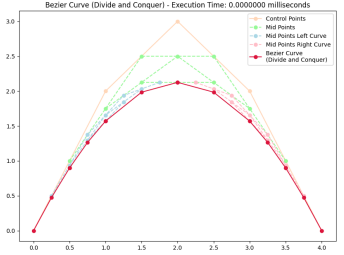
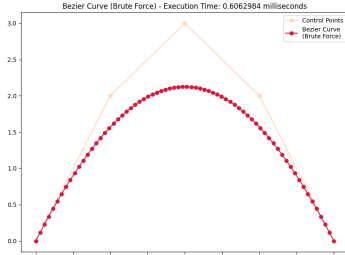
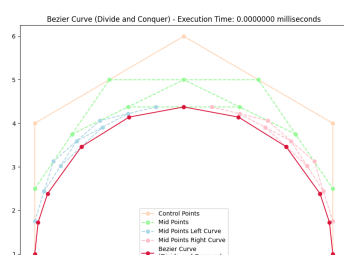
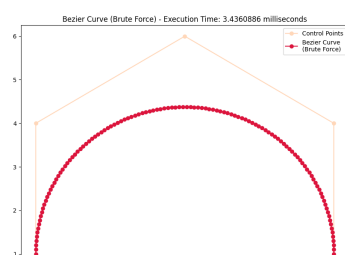
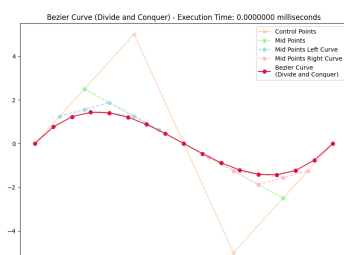
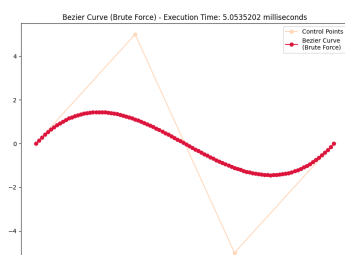
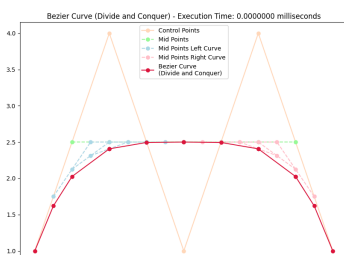
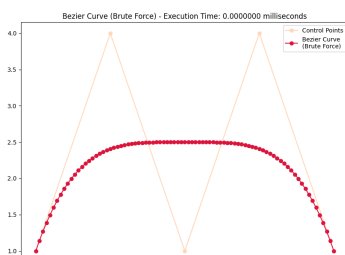
```

Gambar 3. Algoritma *brute force*

5. Tangkapan Layar Hasil Keluaran dari Masukan

Hasil implementasi dari masukan dan keluaran program dari algoritma *divide and conquer* dan algoritma *brute force*, yakni sebagai berikut.

Masukan	Algoritma <i>Divide and Conquer</i>	Algoritma <i>Brute Force</i>
TC 1 Toleransi (Iterasi): 0.2 (10 titik): (1,1), (5,2), (6,1)	 <p>Bezier Curve (Divide and Conquer) - Execution Time: 0.000000 milliseconds</p> <p>Hasil: 0,0 ms</p>	 <p>Bezier Curve (Brute Force) - Execution Time: 0.000000 milliseconds</p> <p>Hasil: 0,0 ms</p>
TC 2 Toleransi (Iterasi): 0.5 (10 titik): (0, 0), (1, 3), (2, 6), (3, 7), (4, 6), (5, 3), (5, 0), (4, -3), (2, -2), (0, 0)	 <p>Bezier Curve (Divide and Conquer) - Execution Time: 4.009621 milliseconds</p> <p>Hasil: 4,0 ms</p>	 <p>Bezier Curve (Brute Force) - Execution Time: 17.802385 milliseconds</p> <p>Hasil: 17,8 ms</p>

<p>TC 3 Toleransi (Iterasi): 0.5</p> <p>(5 titik): (0, 0), (1, 2), (2, 3), (3, 2), (4, 0)</p>	 <p>Hasil: 0,0 ms</p>	 <p>Hasil: 0,6 ms</p>
<p>TC 4 Toleransi (Iterasi): 1.2</p> <p>(5 titik): (1, 1), (1, 4), (3, 6), (5, 4), (5, 1)</p>	 <p>Hasil: 0,0 ms</p>	 <p>Hasil: 3,4 ms</p>
<p>TC 5 Toleransi (Iterasi): 1</p> <p>(4 titik): (0, 0), (3, 5), (6, -5), (9, 0)</p>	 <p>Hasil: 0,0 ms</p>	 <p>Hasil: 5,0 ms</p>
<p>TC 6 Toleransi (Iterasi): 0.85</p> <p>(5 titik): (0, 0), (2, 2), (4, 0), (6, 2), (8, 0)</p>	 <p>Hasil: 0,0 ms</p>	 <p>Hasil: 0,0 ms</p>

Tabel 1. Tangkapan Layar I/O

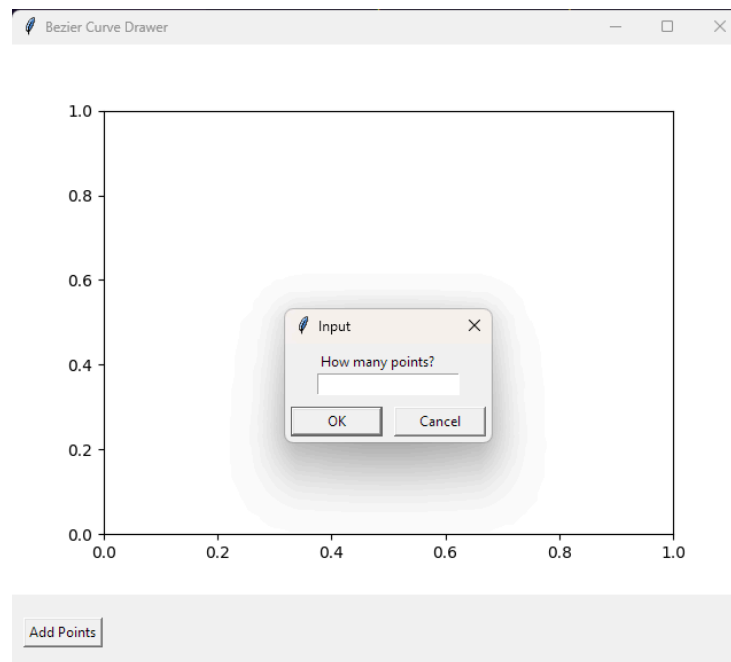
6. Hasil Analisis Perbandingan Kedua Solusi Algoritma

Berdasarkan hasil tangkapan hasil keluaran program dari masukan *test cases* yang ada, solusi dengan algoritma *divide and conquer* dapat dieksekusi dengan lebih cepat dibandingkan solusi dengan algoritma *brute force*. Hasil kurva yang dibentuk oleh solusi *brute force* lebih halus dibandingkan dengan algoritma *divide and conquer*, tetapi hasil algoritma *divide and conquer* juga cukup halus. Dilihat dari hasil implementasi dan perhitungan kompleksitas algoritma, terbukti algoritma *divide and conquer* lebih sangkil dan mangkus daripada algoritma *brute force* dalam pembuatan kurva Bézier.

7. Implementasi Bonus

(1) Program dapat membuat kurva untuk n titik kontrol

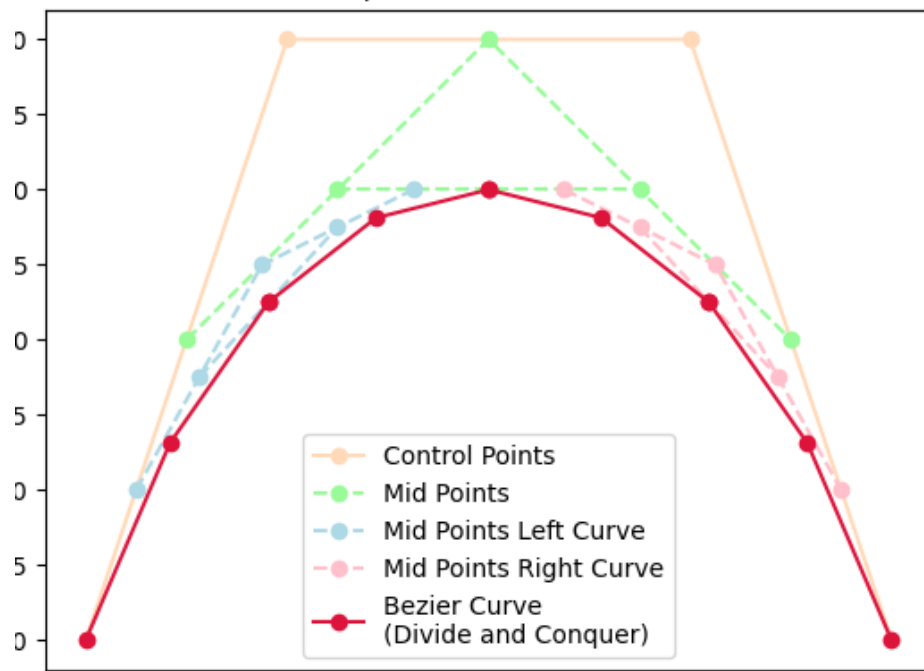
Untuk pembuatan kurva dengan n titik kontrol, digunakan algoritma yang sama baik untuk algoritma *brute force* dan *divide and conquer*. Program akan meminta masukan berupa jumlah titik kontrol untuk kurva yang akan dibentuk.



Gambar 4. Permintaan masukan berupa jumlah titik kontrol

(2) Program dapat melakukan visualisasi proses pembuatan kurva

Selagi memasukkan titik-titik kurva, program akan menunjukkan kurva yang terbentuk setelah setiap penambahan titik. Setiap gambar kurva tersebut juga akan disimpan bersama gambar kurva final pada folder `./test/Output`. Selain itu, program juga dapat menunjukkan beberapa titik tengah utama pada algoritma *divide and conquer*. Tidak semua titik tengah ditunjukkan untuk kejelasan gambar.



Gambar 5. Titik tengah pada algoritma *divide and conquer*

DAFTAR PUSTAKA

1. Munir, Rinaldi. “Algoritma Divide and Conquer (Bagian 1)” (online).
([https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-\(2024\)-Bagian1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian1.pdf), diakses pada 13 Maret 2024).
2. Munir, Rinaldi. “Algoritma Divide and Conquer (Bagian 2)” (online).
([https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-\(2024\)-Bagian2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian2.pdf), diakses pada 13 Maret 2024).
3. Munir, Rinaldi. “Algoritma Divide and Conquer (Bagian 3)” (online).
([https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-\(2024\)-Bagian3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian3.pdf), diakses pada 13 Maret 2024).
4. Munir, Rinaldi. “Algoritma Divide and Conquer (Bagian 4)” (online).
([https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-\(2024\)-Bagian4.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian4.pdf), diakses pada 13 Maret 2024).

LAMPIRAN

Daftar Pembagian Tugas

KEGIATAN		PIC
ALGORITMA	DnC	13522010
	Bruteforce	13522042
	GUI	13522010 13522042
LAPORAN	Analisis dan implementasi dalam algoritma brute force sebagai algoritma pembandingan	13522042
	Analisis dan implementasi dalam algoritma divide and conquer	13522010
	Source code program implementasi	13522010 13522042
	Tangkapan layar I/O	13522010 13522042
	Hasil analisis perbandingan solusi brute force dengan divide and conquer	13522010 13522042
	Penjelasan mengenai implementasi bonus	13522010 13522042
BONUS	Program dapat membuat kurva untuk n titik kontrol	13522010 13522042
	Program dapat melakukan visualisasi proses pembuatan kurva	13522010 13522042

Tabel 2. Pembagian tugas dalam pembuatan Tugas Besar

Pengecekan Program

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.	✓	

Tabel 3. Tabel Pengecekan Program

Repository

Link Repository dari Tugas Kecil 02 IF2211 Strategi Algoritma adalah sebagai berikut.

https://github.com/amaliap21/Tucil2_13522010_13522042.git