TUGAS KECIL 2

IF2211 STRATEGI ALGORITMA

SEMESTER II TAHUN 2023/2024

"Aproksimasi Kurva Bezier dengan Algoritma *Divide and Conquer*"

OLEH:

Ahmad Hasan Albana 13522041

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

KATA PENGANTAR

Puji syukur kita panjatkan ke hadirat Allah SWT, 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.

Penyusunan makalah ini tidak lepas dari bantuan berbagai pihak. Kami mengucapkan terima kasih kepada dosen 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, 19 Maret 2024,

Ahmad Hasan Albana (13522041)

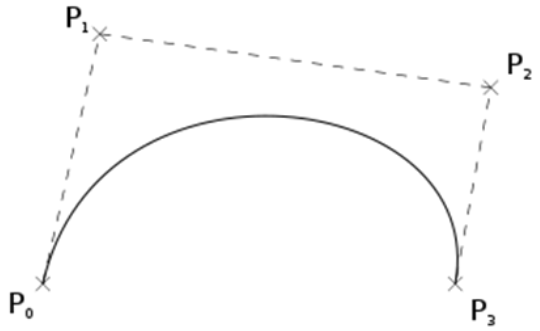
DAFTAR ISI

KATA PENGANTAR **1**
DAFTAR ISI **2**
BAB I **3**
1.1. Abstraksi 3
BAB II **5**
2.1. Algoritma Divide and Conquer 5
2.2. Algoritma Brute Force 7
BAB III **9**
3.1 Implementasi Kode Divide and Conquer 9
3.2 Implementasi Kode Brute Force 10
BAB IV **11**
4.1 Hasil Pengujian 11
4.2 Pembahasan Hasil Pengujian 13
DAFTAR PUSTAKA **14**
LAMPIRAN **15**
Repository **15**

BAB I
DESKRIPSI MASALAH

1.1. Abstraksi

Kurva Bezier adalah kurva yang sangat sering digunakan dalam desain grafis, animasi, dan desain rekayasa. Kurva ini dibuat dengan menetapkan beberapa titik kontrol yang menentukan arah dan lengkungan kurva. Kurva berawal dari titik kontrol pertama dan bergerak melengkung mengikuti arah posisi relatif titik kontrol lain (walaupun tidak tepat mengenai titik kontrol), sebelum akhirnya berhenti tepat pada titik kontrol terakhir. Kurva Bezier merupakan salah satu penemuan terpenting dalam dunia desain digital dan memiliki banyak aplikasi di dunia nyata, seperti *pen tool*, animasi yang halus dan realistis, membuat desain produk yang kompleks dan presisi, dan membuat font yang indah dan unik. Kelebihan dari kurva bezier ini adalah kurva dapat dibuat serta dimanipulasi dengan mudah serta memiliki presisi yang tinggi.



Gambar 1.1 Kurva Bezier Kubik
(Sumber: https://id.wikipedia.org/wiki/Kurva_B%C3%A9zier)

Pada Tugas Kecil 2 ini, penulis merancang program untuk menerapkan algoritma Divide and Conquer yang telah dipelajari di kelas untuk melakukan aproksimasi hasil kurva bezier. Dilakukan juga aproksimasi hasil kurva bezier menggunakan algoritma *brute force* sehingga dapat dibandingkan antara proses perhitungan secara *brute force* dengan secara *divide and conquer* serta waktu eksekusi masing-masing.

Pada Tugas Kecil 2 ini, penulis menerapkan spesifikasi bonus yaitu untuk membuat program yang dapat menangani sembarang *n* buah titik kontrol lebih dari 3 dan menampilkan setiap iterasi pada algoritma *divide and conquer* sebagai visualisasi proses pembentukan kurva bezier.

Program ini dibuat menggunakan bahasa Python dan dijalankan melalui CLI, serta menerima input berupa:

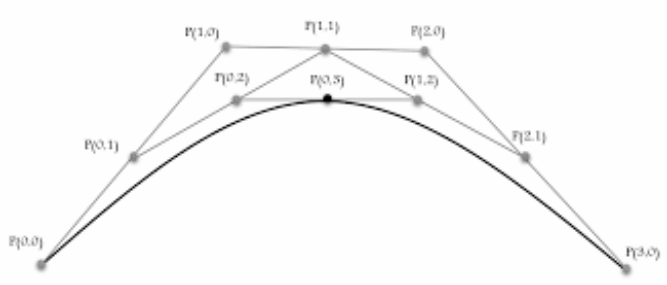
1) Pilihan metode input, yaitu dapat langsung dari keyboard atau melalui input file yang berisi poin poin yang disebutkan dibawah ini.
2) Jumlah titik kontrol *n* (harus lebih dari sama dengan 3 titik),
3) Jumlah iterasi (harus lebih dari sama dengan 1 iterasi), serta
4) Koordinat masing-masing titik kontrol, yaitu (x_1, y_1), (x_2, y_2), …, (x_n, y_n).

Setelah menerima input, program akan melakukan kalkulasi menggunakan algoritma *divide and conquer* dan *brute force*, lalu menampilkan kedua hasil pembentukan aproksimasi kurva bezier tersebut menggunakan kakas *matplotlib*.

BAB II
IMPLEMENTASI ALGORITMA

2.1. Algoritma *Divide and Conquer*

Algoritma *divide and conquer* adalah salah satu pendekatan algoritma untuk menyelesaikan berbagai persoalan komputasi. Ide utama dari algoritma ini yaitu membagi persoalan (*divide*) menjadi beberapa upa-persoalan hingga upa-persoalan tersebut dapat diselesaikan secara langsung (*solve*), lalu kemudian menggabungkan (*combine*) semua solusi dari upa-persoalan tersebut untuk menjawab persoalan awal. Karena sifatnya yang terus membagi upa-persoalan hingga ukurannya cukup kecil untuk langsung diselesaikan, maka pendekatan ini umumnya diimplementasikan secara rekursif.



Gambar 2.1 Pembentukan Kurva Bézier Kubik dengan Divide and Conquer
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian4.pdf)

Pada program ini, algoritma *divide and conquer* untuk mengaproksimasi kurva bezier diimplementasikan dengan tahapan berikut. Misalkan terdapat n buah titik kontrol, yaitu $P1_1$, $P1_2$, dan seterusnya hingga $P1_n$, dengan jumlah aproksimasi yang diinginkan adalah sebanyak *x* kali.

1) Pertama, buat garis yang menghubungkan setiap titik kontrol secara berurutan, yaitu garis yang menghubungkan $P1_1$ dengan $P1_2$, garis yang menghubungkan $P1_2$ dengan $P1_3$, hingga menghubungkan $P1_{n-1}$ dengan $P1_n$.
2) Buat titik yang berada di tengah masing-masing garis, misalnya $P2_1$ yang berada di tengah garis yang menghubungkan $P1_1$ dengan $P1_2$, hingga $P2_{n-1}$ yang berada di tengah garis yang menghubungkan $P1_{n-1}$ dengan $P1_n$.
3) Ulangi langkah ke 1 dan 2 dengan menggunakan setiap titik tengah yang didapat sebagai titik kontrol, hingga didapatkan titik PN_1.
4) Didapat himpunan solusi titik pada iterasi pertama adalah ($P1_1$,PN_1,$P1_n$). Untuk mendapatkan himpunan solusi titik pada iterasi selanjutnya, ulangi langkah 1, 2, dan 3 dengan menggunakan ($P1_1$,$P2_1$,...,PN_1) dan (PN_1,$PN\text{-}1_2$,...,$P1_n$), masing-masing sebagai himpunan titik kontrol yang baru. Gabungkan seluruh himpunan solusi yang didapat dengan aturan titik harus tetap terurut (titik berada sebelum titik lain, tetap harus berada sebelum titik lain tersebut walau jaraknya berbeda) dan menghapus anggota yang sama.
5) Ulangi langkah ke 4 hingga didapatkan himpunan solusi titik pada iterasi *x* yang diinginkan.

Berdasarkan prosedur *divide and conquer* diatas, didapat kompleksitas pembangun aproksimasi kurva bezier dengan iterasi *x* dan *n* buah titik kontrol menggunakan algoritma *divide and conquer* adalah sebagai berikut

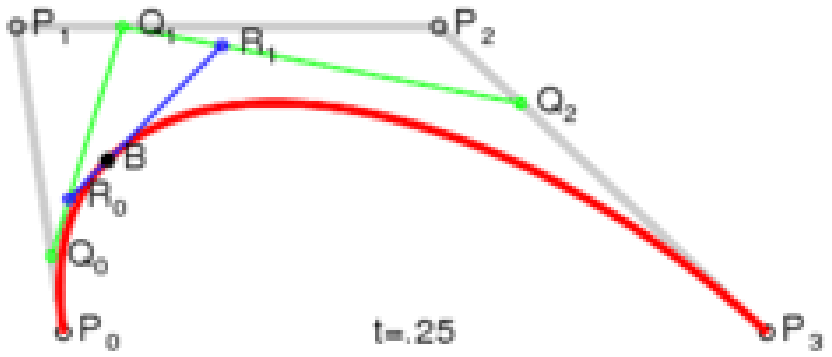
$$T(x, n) = (\sum_{i=0}^{x} 2^i)(\sum_{i=1}^{n-1} i) = (2\text{^}(x + 1) - 1)(n * (n - 2))$$

$$T(x, n) = O(2\text{^}x * n * n)$$

dengan hanya memperhitungkan proses pencarian titik tengah antara dua titik, (2^(*x* + 1) - 1) adalah banyak iterasi pencarian 1 buah titik solusi dari *n* buah titik kontrol, dan (n * (n - 2)) adalah banyak operasi pencarian titik tengah antara 2 titik untuk menemukan 1 buah titik solusi dari *n* buah titik kontrol.

2.2. Algoritma *Brute Force*

Sebagai pembanding, diimplementasikan pula algoritma aproksimasi kurva bezier menggunakan pendekatan *brute force*. Algoritma *brute force* adalah pendekatan yang lempang (*straightforward*) untuk memecahkan suatu persoalan. Algoritma *brute force* umumnya didasarkan pada pernyataan pada persoalan (*problem statement*), dan definisi/konsep yang dilibatkan. Algoritma *brute force* memecahkan persoalan dengan sangat sederhana, langsung, dan jelas caranya (*obvious way*).



Gambar 2.1 Pembentukan Kurva Bézier Kubik dengan Brute Force
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian4.pdf)

Pada program ini, algoritma *brute force* dirancang berdasarkan definisi umum kurva bezier yaitu dimana setiap titik pada suatu garis yang dibentuk oleh titik lain yang terbentuk memiliki perbandingan jarak ke kedua titik pembentuk garis yang sama. Dalam notasi matematika, hal tersebut dapat ditulis menjadi

$$\mathbf{Q}_0 = \mathbf{B}(t) = (1 - t)\,\mathbf{P}_0 + t\mathbf{P}_1\,, \quad 0 \le t \le 1$$

$$\mathbf{Q}_1 = \mathbf{B}(t) = (1 - t)\,\mathbf{P}_1 + t\mathbf{P}_2\,, \quad 0 \le t \le 1$$

$$\mathbf{R}_0 = \mathbf{B}(t) = (1 - t)\,\mathbf{Q}_0 + t\mathbf{Q}_1\,, \quad 0 \le t \le 1$$

dengan titik $\mathbf{P}_0,\mathbf{P}_1,...,\mathbf{P}_n$ adalah himpunan titik kontrol, titik $\mathbf{Q}_0,\mathbf{Q}_1,...,\mathbf{Q}_{n-1}$ adalah himpunan titik tengah dari himpunan titik $\mathbf{P}$, titik $\mathbf{R}_0,\mathbf{R}_1,...,\mathbf{R}_{n-2}$ adalah titik tengah dari himpunan titik $\mathbf{Q}$, dan seterusnya hingga diperoleh himpunan titik yang hanya memiliki 1 anggota. Berdasarkan persamaan diatas, didapatkan rumus sebagai berikut:

$$\mathbf{B}(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i$$
$$= (1-t)^n \mathbf{P}_0 + \binom{n}{1} (1-t)^{n-1} t \mathbf{P}_1 + \cdots +$$
$$\binom{n}{n-1} (1-t) t^{n-1} \mathbf{P}_{n-1} + t^n \mathbf{P}_n, \qquad 0 \leqslant t \leqslant 1$$

Sehingga untuk mencari titik-titik pembangun aproksimasi kurva bezier dengan iterasi *x*, program hanya memerlukan 2*x*+1 titik untuk membangun kurva tersebut. Oleh karena itu, didapat kompleksitas pembangun aproksimasi kurva bezier dengan iterasi *x* dan *n* buah titik kontrol menggunakan algoritma *brute force* adalah sebagai berikut

$$T(x, n) = (2x + 1)((n + n + 2) * n)$$
$$T(x, n) = O(x * n * n)$$

dengan hanya memperhitungkan operasi perkalian, (2*x* + 1) adalah banyak perulangan yang dibutuhkan untuk mencari seluruh titik solusi, (*n* + *n* + 2) adalah kompleksitas kombinasi ditambah dengan operasi perpangkatan sebanyak *n* dan 2 buah perkalian antara kombinasi, perpangkatan, dan titik kontrol, dan *n* terakhir merupakan banyak perulangan perkalian antara kombinasi, perpangkatan, dan titik kontrol.

BAB III
IMPLEMENTASI KODE PROGRAM

3.1 Implementasi Kode *Divide and Conquer*

```
def BaseBezier(pointSet: List[Point]) -> List[Point]:
    global timeDnC
    lenSet = len(pointSet)
    pointList = pointSet

    counterIdx = 1
    resultList = [pointSet[0], pointSet[lenSet-1]]

    while (lenSet > 2):
        pointListTemp = []

        time_start = 1000*time.perf_counter()
        for i in range(1, lenSet):
            pointListTemp.append(GetMiddlePoint(pointList[i-1], pointList[i]))
        time_stop = 1000*time.perf_counter()
        a = time_stop - time_start
        timeDnC += a

        x = np.array([i.x for i in pointList])
        y = np.array([i.y for i in pointList])
        ax1.plot(x,y, color = 'black', linewidth = 1, ls = 'dashed')

        pointList = pointListTemp
        lenSet -= 1
        resultList.insert(counterIdx, pointList[lenSet-1])
        resultList.insert(counterIdx, pointList[0])
        counterIdx += 1

    resultList.insert(counterIdx, GetMiddlePoint(pointList[0], pointList[1]))

    return resultList
def BezierPointGenerator(pointSet: List[Point], iteration: int, nControl: int):
    global color_list
    global iterationNow

    if (color_list[iterationNow-1][0] == 0):
        label = "Iteration " + str(iterationNow)
        color_list[iterationNow-1][0] = 1
        iterationNow += 1
    else:
        label = ""

    listPoint = [pointSet[i] for i in range(0, len(pointSet), nControl-1)]
    x = np.array([i.x for i in listPoint])
    y = np.array([i.y for i in listPoint])

    hex_color = '#{:02x}{:02x}{:02x}'.format(*color_list[iteration-1][1])
    ax1.plot(x,y, color = hex_color, linewidth = 1, label = label)

    if (iteration != 0):
        setBaru = BaseBezier(pointSet)

        BezierPointGenerator(setBaru[:nControl], iteration-1, nControl)
        BezierPointGenerator(setBaru[nControl-1:], iteration-1, nControl)

def DivideNConquer(nControl:int, controlSet: List[Point], iteration: int):
    BezierPointGenerator(controlSet, iteration, nControl)
    strDnC = "Execution Time: " + str(round(timeDnC,3)) + " ms"
    fig1.suptitle(strDnC, fontsize=10)
```

Fungsi BaseBezier adalah fungsi yang berguna untuk mencari 1 buah titik solusi dengan *n* buah titik kontrol. Fungsi BezierPointGenerator adalah fungsi yang berguna untuk mengiterasi sebanyak iterasi yang diinginkan dengan cara membagi titik hasil dari iterasi sebelumnya menjadi dua dan kembali memproses titik tersebut menggunakan fungsi BaseBezier. Fungsi DivideNConquer hanya berperan sebagai fungsi yang dipanggil pada program utama.

3.2 Implementasi Kode *Brute Force*

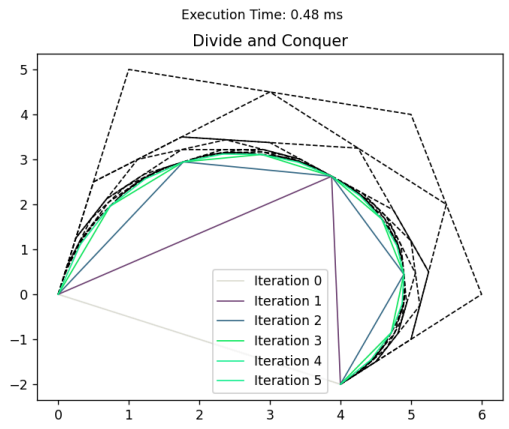
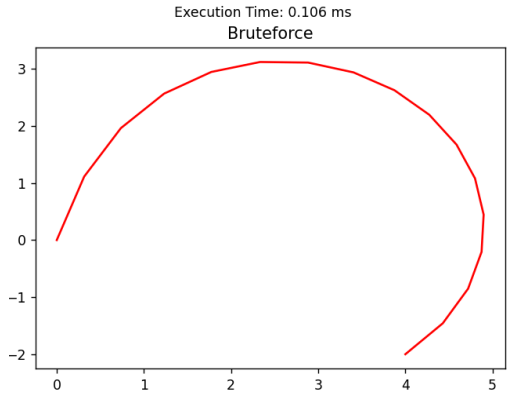
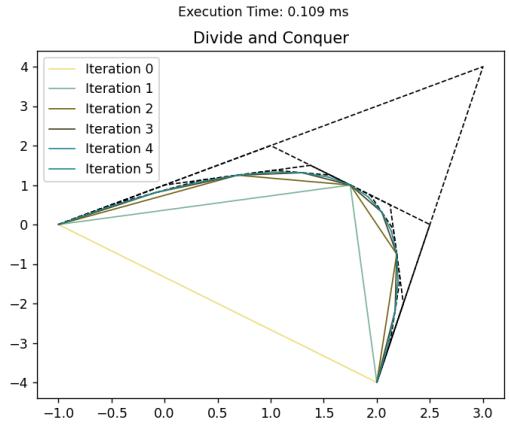
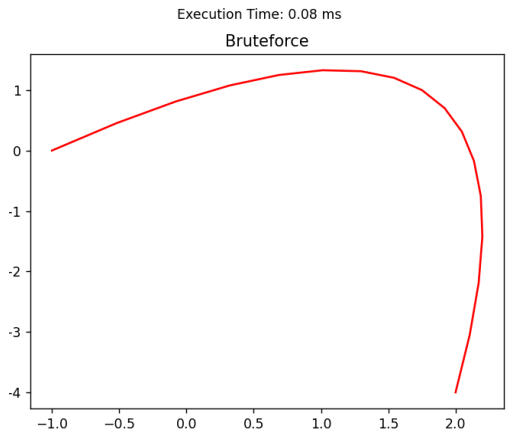
```
def findPointBruteforce(nControl:int, controlSet: List[Point], xValue: int):
    resultX = 0
    resultY = 0
    for i in range(nControl):
        a = ((1 - xValue)**(nControl - 1 - i))*(xValue**i)*getCombination(nControl-1, i)
        resultX += a*controlSet[i].x
        resultY += a*controlSet[i].y
    return resultX, resultY

def BruteForce(nControl: int, controlSet: List[Point], iteration: int):
    global timeBF
    xResult = []
    yResult = []

    time_start = time.perf_counter()

    i = 0
    while i <= 1:
        x, y = findPointBruteforce(nControl, controlSet, i)

        xResult.append(x)
        yResult.append(y)
        i += (1/(2**iteration))

    time_stop = time.perf_counter()
    a = 1000*time_stop - 1000*time_start
    timeBF += a

    ax2.plot(np.array(xResult), np.array(yResult), color = 'r')

    strBF = "Execution Time: " + str(round(timeBF,3)) + " ms"
    fig2.suptitle(strBF, fontsize=10)
```

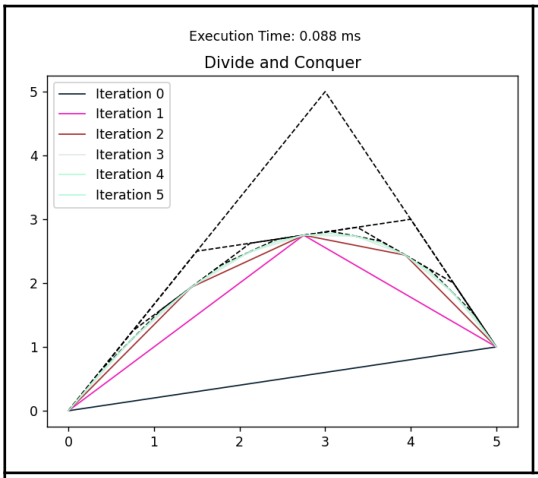
Fungsi findPointBruteforce adalah fungsi yang mengembalikan titik baru yang merupakan hasil dari rumus B(t) yang telah didapatkan pada bagian 2.2 dengan t yang digunakan adalah nilai dari xValue. Fungsi BruteForce adalah fungsi yang berguna untuk mengiterasi setiap nilai dari xValue yang perlu dicari sesuai dengan nilai iterasi yang diinginkan dan juga sebagai fungsi yang dipanggil di program utama untuk menggunakan algoritma *brute force*.

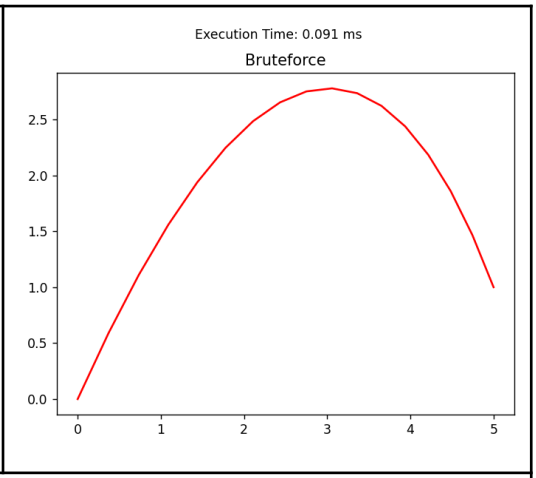
BAB IV
HASIL PENGUJIAN DAN ANALISIS

4.1 Hasil Pengujian

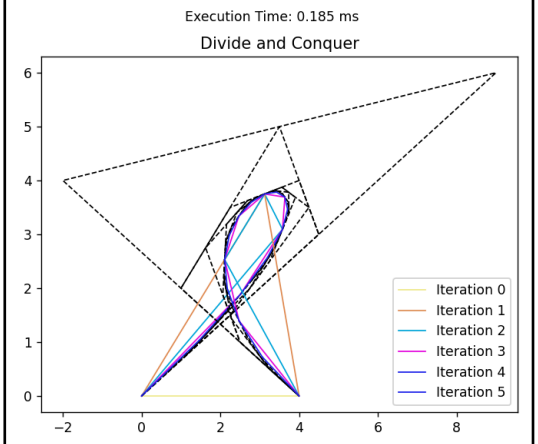
Berikut adalah hasil pengujian dengan menggunakan data uji yang berada pada folder *test.*

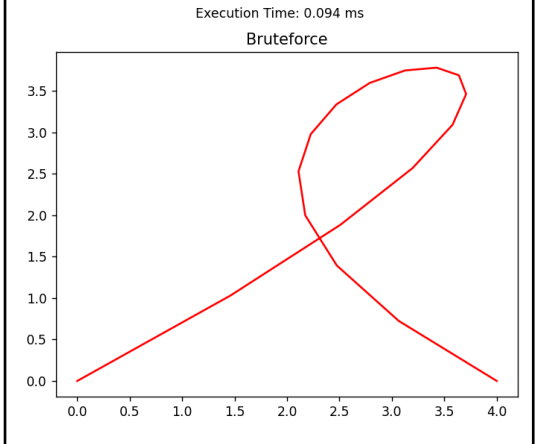
DIVIDE AND CONQUER	BRUTE FORCE
test1.txt	
	
test2.txt	
	
test3.txt	



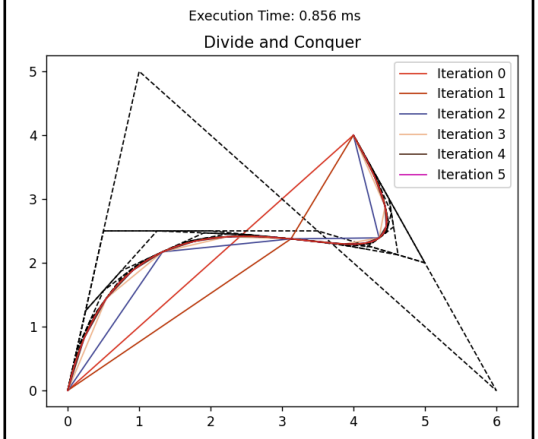


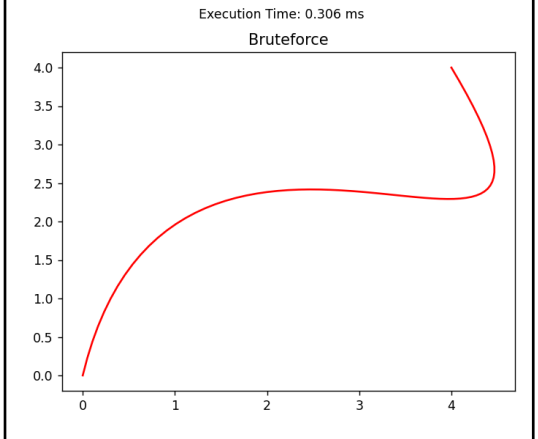
test4.txt



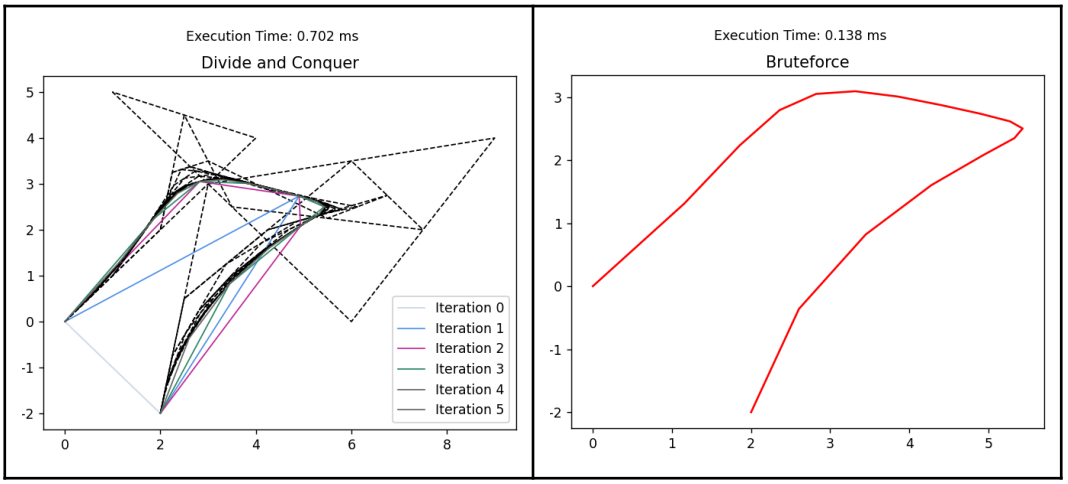


test5.txt





test6.txt



4.2 Pembahasan Hasil Pengujian

Pada bab 2, telah diketahui bahwa kompleksitas algoritma *divide and conquer* adalah O(2^x * n * n), sedangkan kompleksitas algoritma *brute force* adalah O(x * n * n). Dengan berdasar pada kompleksitas tersebut, dapat kita ketahui bahwa algoritma *divide and conquer* akan memiliki waktu eksekusi yang lebih lama daripada algoritma *brute force*.

Berdasarkan hasil pengujian diatas, terlihat bahwa algoritma *divide and conquer* memang terbukti memiliki waktu eksekusi yang lebih lama daripada algoritma *brute force*. Lalu, dapat dilihat pola juga untuk kedua algoritma dengan kasus nilai n dan nilai x yang sama. Untuk nilai n yang lebih kecil, perbandingan antara waktu eksekusi kedua algoritma juga lebih kecil. Contohnya untuk n = 2, perbandingan waktu eksekusinya bahkan tidak sampai dua kali lipat, sedangkan untuk n = 7, perbandingan waktu eksekusinya dapat mencapai lima kali lipat. Hal tersebut juga sejalan dengan perbandingan kompleksitas kedua algoritma yang telah dilakukan.

DAFTAR PUSTAKA

1. 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, diakses pada 19 Maret 2024).

LAMPIRAN

Repository

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

https://github.com/Bana-man/Tucil2_13522041.git