LAPORAN TUGAS KECIL 2 IF2211 STRATEGI ALGORITMA

MENCARI PASANGAN TITIK TERDEKAT 3D DENGAN ALGORITMA DIVIDE AND CONQUER

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

Disusun oleh :

Mohd Farhan Fahrezy	**13521106**
Frankie Huang	**13521092**

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2022/2023

Daftar Isi

BAB I

DESKRIPSI MASALAH

Persoalan mencari pasangan titik terdekat dengan Algoritma *Divide and Conquer* telah dijelaskan dalam kuliah Strategi Algoritma. Dalam kuliah tersebut, persoalan dirumuskan untuk titik pada bidang datar (2D). Pada Tugas Kecil 2 kali ini penulis diminta untuk mengembangkan algoritma mencari sepasang titik terdekat pada bidang 3D. Misalkan terdapat n buah titik pada ruang 3D. Setiap titik P di dalam ruang dinyatakan dengan koordinat P = (x, y, z). Carilah sepasang titik yang mempunyai jarak terdekat satu sama lain. Jarak dua buah titik P1 = (x1, y1, z1) dan P2 = (x2, y2, z2) dihitung dengan rumus Euclidean berikut:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Pada tugas kecil ini, penulis diminta untuk membuat program dalam dalam Bahasa Python untuk mencari sepasang titik yang jaraknya terdekat satu sama lain dengan menerapkan *algoritma divide and conquer* untuk penyelesaiannya, dan membandingkan hasilnya dengan hasil Algoritma *Brute Force*. Penulis juga diminta untuk melakukan generalisasi program sehingga dapat mencari pasangan titik terdekat untuk sekumpulan vektor di R^n dan setiap vektor dinyatakan dalam bentuk x = (x1, x2,…, xn).

BAB II
TEORI DASAR

A. Algoritma Brute Force

Brute Force merupakan salah satu pendekatan yang lempang (straight-forward) dalam hal pemecahan suatu masalah atau persoalan dengan sangat sederhana (simple), langsung, dan jelas(obvious way). Algoritma Brute Force sering kali disebut juga sebagai algoritma naif (naive algorithm).

Karakteristik algoritma brute force umumnya tidak "pintar" dan tidak mangkus, karena ia membutuhkan jumlah langkah yang besar dalam penyelesaiannya, sehingga terkadang algoritma brute force disebut juga algoritma naif (naïve algorithm). Algoritma brute force seringkali merupakan pilihan yang kurang disukai karena ketidakmangkusannya itu, tetapi dengan mencari pola pola yang mendasar, keteraturan, atau trik-trik khusus, biasanya akan membantu kita menemukan algoritma yang lebih cerdas dan lebih mangkus. Untuk masalah yang ukurannya kecil, kesederhanaan brute force biasanya lebih diperhitungkan daripada ketidakmangkusannya. Algoritma brute force sering digunakan sebagai basis bila membandingkan beberapa alternatif algoritma yang mangkus. Algoritma brute force seringkali lebih mudah diimplementasikan daripada algoritma yang lebih canggih, dan karena kesederhanaannya, kadang-kadang algoritma brute force dapat lebih mangkus (ditinjau dari segi implementasi)

B. Algoritma Divide and Conquer

Divide and Conquer adalah salah satu algoritma dalam pemrograman yang dilakukan dengan memecah persoalan menjadi persoalan-persoalan yang lebih kecil dengan tipe yang sama hingga menjadi cukup sederhana untuk diselesaikan secara langsung. Solusi dari persoalan kecil tersebut kemudian digabungkan sehingga membentuk solusi persoalan semula.

Objek dari persoalan yang menggunakan algoritma ini umumnya adalah objek multi-value, seperti sebuah array. Tiap upa-persoalan memiliki karakteristik yang serupa dengan karakteristik persoalan semula, sehingga metode divide and conquer lebih natural diungkapkan dalam skema rekursif.

Perancangan algoritma ini umumnya terbagi menjadi dua, yaitu *easy divide hard combine* dan *hard divide easy combine*. Perancangan yang pertama adalah kasus dimana pembagian masalah cukup sederhana (seperti membagi array menjadi dua bagian), namun tahap

penggabungannya cukup kompleks (terdapat pengecekan-pengecekan untuk mengambil solusi yang benar dari upa-persoalan).

BAB III
ALGORITMA

Secara singkat penjelasan algoritma *Closest Pair Problem* dengan *Divide and Conquer* yang digunakan adalah sebagai berikut.

1. Cek kasus terbaik, yaitu:
    1. Jika jumlah titiknya 1, maka kembalikan tak hingga.
    2. Jika jumlah titiknya 2, maka kembalikan jarak antara kedua titik.
2. Urutkan titik berdasarkan nilai salah satu sumbu, kemudian belah kedua titik tersebut menjadi dua bagian yang sama rata.
3. Lalu kita panggil kembali fungsi divide and conquer untuk masing-masing titik yang berada di kiri dan titik yang berada di kanan.
4. Bandingkan jarak antara kedua titik, lalu simpan nilai terkecilnya (*delta*).
5. Tarik garis sepanjang *delta* dari garis tengah, kemudian simpan semua titik yang berada pada daerah tersebut ke dalam *strip*.
6. Cek dua jenis kemungkinan:
    1. Jika dimensi yang dikerjakan >= 3, maka panggil kembali fungsi divide and conquer pada array *strip* dengan 1 dimensi yang lebih rendah
    2. Jika dimensi yang dikerjakan = 2, maka secara iteratif, cari jarak terkecil di antara titik tersebut, kemudian bandingkan dengan nilai *delta*. Kembalikan nilai yang lebih kecil.

Algoritma *Quick Sort* dengan *Divide and Conquer* yang diimplementasikan dibagi ke dalam 2 buah fungsi. Terdapat fungsi utama untuk membagi array dan sebuah fungsi lain untuk mencari indeks partisi dari array.

Algoritma fungsi utama:

1. Cek kasus terbaik, yaitu jika panah kanan berada di indeks yang lebih kecil dari panah kiri
2. Panggil fungsi partisi, kemudian panggil fungsi quickSort untuk elemen di kiri indeks partisi dan elemen di kanan indeks partisi
3. Kembalikan array

Algoritma fungsi mencari indeks partisi:

1. Set sebuah elemen dari array menjadi pivot (dalam program dipilih elemen dengan indeks tertinggi)
2. Cari sebuah elemen dari kiri yang lebih besar atau sama dengan dari pivot
3. Cari sebuah elemen dari kanan yang lebih kecil atau sama dengan pivot (dalam kasus ini nilainya adalah pivot)
4. Pertukarkan kedua nilai, lalu kembalikan indeks dari pivot.

BAB IV

SOURCE CODE

A. bruteForce.py

```python
from utilities import calculateDistance

def closestPair(arrayCoordinate):
    closest = 1000000
    tempi = -1
    tempj = -1
    count = 0
    for i in range(len(arrayCoordinate)-1):
        for j in range(i+1,len(arrayCoordinate)):
            count+=1
            temp = calculateDistance(arrayCoordinate[i],arrayCoordinate[j],len(arrayCoordinate[i]))
            if temp<closest:
                closest = temp
                tempi = i
                tempj = j
    return tempi,tempj
```

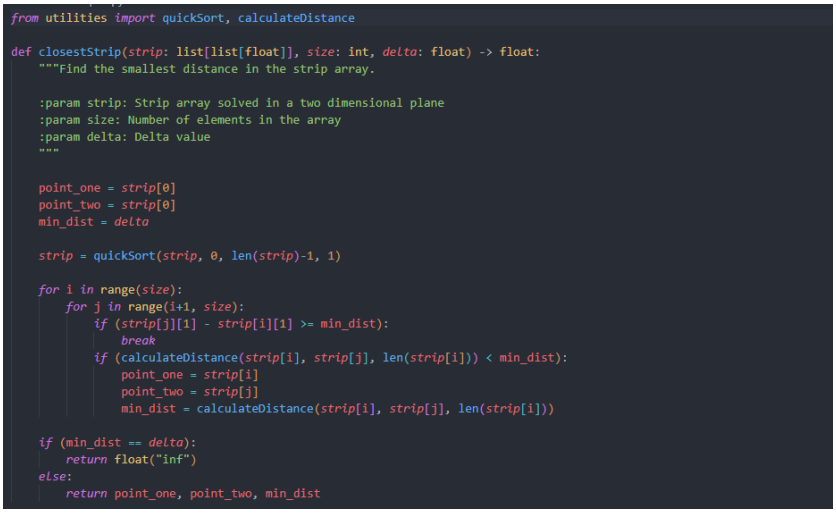
Fungsi closestPair() digunakan untuk menentukan dua titik dalam *arrayCoordinate* yang memiliki jarak terdekat dengan menggunakan algoritma *brute force.*

B. displayCoordinate.py

```python
import matplotlib.pyplot as plt

def displayCoordinate(arrayCoordinate, c1, c2):
    # c1 dan c2 adalah 2 koordinat yang akan digaris
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    # Masukin semua koordinat ke diagram
    for i in range(len(arrayCoordinate)):
        x, y, z = arrayCoordinate[i]
        ax.scatter(x, y, z, c='g',s=1)

    # Buat garis diantara c1 dan c2
    x1, y1, z1 = c1
    x2, y2, z2 = c2
    ax.scatter(x1, y1, z1, c='r',s=10)
    ax.scatter(x2, y2, z2, c='r',s=10)
    ax.plot([x1, x2], [y1, y2], [z1, z2], c='r')

    ax.set_xlabel('Sumbu X')
    ax.set_ylabel('Sumbu Y')
    ax.set_zlabel('Sumbu Z')
    plt.show()
```

Fungsi displayCoordinate() digunakan untuk menampilkan grafik pesebaran titik dengan 2 titik terdekat memiliki warna merah dan sisanya hijau. Fungsi ini menggunakan *library* matplotlib untuk menampilkan grafik.

C. divideAndConquer.py

	Fungsi closestStrip() digunakan untuk menentukan jarak terkecil di dalam strip.
	Fungsi divideAndConquer() digunakan untuk menemukan jawaban persoalan dengan menggunakan algoritma *divide and conquer*.
	Fungsi solveDivideAndConquer() digunakan sebagai fungsi antara yang memanggil fungsi divideAndConquer() dan memberikan keluaran index dari titik terdekat

D. generateCoordinate.py

```python
import random

def generateCoordinate(n,r):
    arrayOfCoordinate = [ [0 for i in range(r)] for j in range(n)]
    for i in range(n):
        for j in range(r):
            arrayOfCoordinate[i][j] = random.uniform(-1000,1000)
    return arrayOfCoordinate

def printCoordinate(coordinate):
    print("(",end="")
    for i in range(len(coordinate)):
        if (i != len(coordinate)-1):
            print(str(coordinate[i])+",",end="")
        else:
            print(coordinate[i],end="")
    print(")",end="")

def printArrayCoordinate(arrayCoordinate):
    for i in range(len(arrayCoordinate)):
        printCoordinate(arrayCoordinate[i])
        print()
```

Fungsi generateCoordinate() digunakan untuk menggenerate titik random sejumlah n pada sebuah dengan vektor domain R^r.
Prosedur printCoordinate() dan printArrayCoordinate() digunakan untuk mencetak array ke layar.

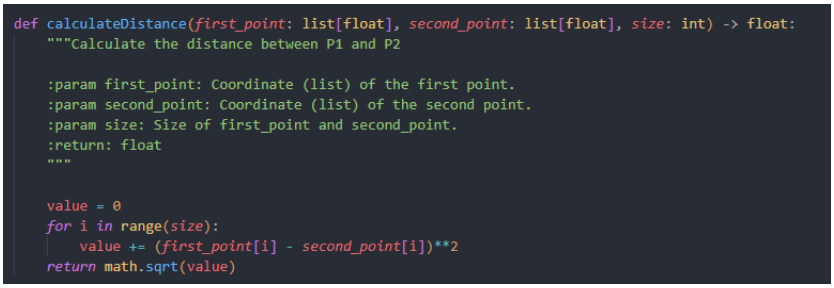
E. utilities.py

```python
def partition(Array: list[list[float]], left: int, right: int, dimension: int) -> (list[list[float]], int):
    """Function to find the partition index

    :param Array: Array of points
    :param left: Index of the left-most value
    :param right: Index of the right-most value
    :param dimension: The dimension to be sorted by
    :return: (list[list[float]], int)
    """

    pivot = Array[right][dimension-1]

    i = left-1

    for j in range(left, right):
        if (Array[j][dimension-1] <= pivot):
            i += 1
            Array[i], Array[j] = Array[j], Array[i]

    Array[i+1], Array[right] = Array[right], Array[i+1]

    return (Array, i + 1)
```

Fungsi partition() digunakan untuk menentukan index dari partisi array.

```python
def quickSort(Array: list[list[float]], left: int, right: int, dimension: int) -> list[list[float]]:
    """Function to sort the array A by the value of the d-th dimension

    :param Array: Array of points
    :param left: Index of the left-most value
    :param right: Index of the right-most value
    :param dimension: The dimension to be sorted by (1, 2, 3, ...)
    :return: list[list[float]]
    """

    if (left < right):
        Array, index = partition(Array, left, right, dimension)

        quickSort(Array, left, index-1, dimension)

        quickSort(Array, index+1, right, dimension)

    return Array
```

Fungsi quickSort() digunakan untuk melakukan sorting pada array berdasarkan value dari dimensi ke-d.

	Fungsi calculateDistance() digunakan untuk menentukan jarak antara 2 titik.
	Fungsi pointToStr() digunakan untuk mengubah suatu titik menjadi string.

F. main.py

	Fungsi main() digunakan untuk menjalankan program secara utuh. User dapat memasukkan jumlah titik (n) dan dimensi dari titik (R) serta menentukan algoritma yang akan digunakan. Program akan memberikan luaran hasil pasangan terdekat, jarak, waktu eksekusi dan visualisasi titik.

G. mainWindow.py

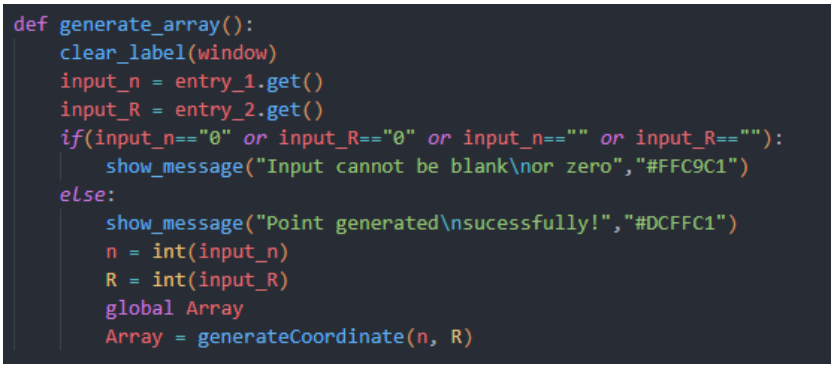
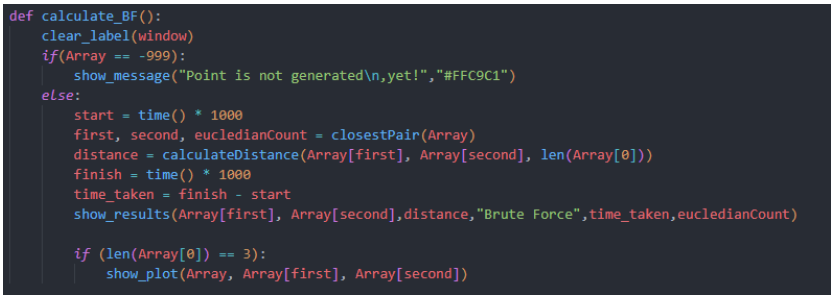
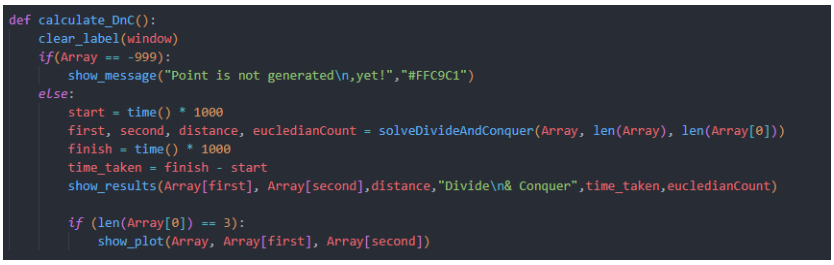
```
window = Tk()

window.geometry("1280x720")
window.configure(bg = "#2E2E48")
window.title("Divide & Conquer")


canvas = Canvas(
    window,
    bg = "#2E2E48",
    height = 720,
    width = 1280,
    bd = 0,
    highlightthickness = 0,
    relief = "flat"
)

canvas.place(x = 0, y = 0)
image_image_1 = PhotoImage(
    file=relative_to_assets("image_1.png"))
image_1 = canvas.create_image(
    640.0,
    360.0,
    image=image_image_1
)

entry_image_1 = PhotoImage(
    file=relative_to_assets("entry_1.png"))
entry_bg_1 = canvas.create_image(
    211.0,
    307.0,
    image=entry_image_1
)
entry_1 = Entry(
    window,
    bd=0,
    bg="#1E1E2E",
    fg="#FFFFFF",
    font=("Arial Black",20),
    selectbackground="#29293F",
    justify="center",
    insertbackground="#FFFFFF",
    highlightthickness=0,
    validate="key",
    validatecommand=(window.register(validate_int), '%S')
)
```

```
entry_1.place(
    x=56.0,
    y=281.0,
    width=310.0,
    height=50.0
)

entry_image_2 = PhotoImage(
    file=relative_to_assets("entry_2.png"))
entry_bg_2 = canvas.create_image(
    211.0,
    480.0,
    image=entry_image_2
)
entry_2 = Entry(
    window,
    bd=0,
    bg="#1E1E2E",
    fg="#FFFFFF",
    font=("Arial Black",20),
    selectbackground="#29293F",
    justify="center",
    insertbackground="#FFFFFF",
    highlightthickness=0,
    validate="key",
    validatecommand=(window.register(validate_int), '%S')
)
entry_2.place(
    x=56.0,
    y=454.0,
    width=310.0,
    height=50.0
)

button_image_1 = PhotoImage(
    file=relative_to_assets("button_1.png"))
button_1 = Button(
    image=button_image_1,
    borderwidth=0,
    highlightthickness=0,
    command=calculate_DnC,
    relief="flat"
)
```

```
button_1.place(
    x=24.0,
    y=631.0,
    width=182.0,
    height=73.0
)

button_image_2 = PhotoImage(
    file=relative_to_assets("button_2.png"))
button_2 = Button(
    image=button_image_2,
    borderwidth=0,
    highlightthickness=0,
    command=calculate_BF,
    fg="#29293F",
    relief="flat"
)
button_2.place(
    x=215.0,
    y=631.0,
    width=182.0,
    height=73.0
)

button_image_3 = PhotoImage(
    file=relative_to_assets("button_3.png"))
button_3 = Button(
    image=button_image_3,
    borderwidth=0,
    highlightthickness=0,
    command=generate_array,
    relief="flat"
)
button_3.place(
    x=16.0,
    y=533.0,
    width=386.0,
    height=86.0
)
```

```
button_image_4 = PhotoImage(
    file=relative_to_assets("button_4.png"))
button_4 = Button(
    image=button_image_4,
    borderwidth=0,
    highlightthickness=0,
    command=open_github,
    relief="flat"
)
button_4.place(
    x=338.0,
    y=72.0,
    width=79.0,
    height=84.0
)
window.resizable(False, False)
window.mainloop()
```

File mainWindow.py digunakan untuk menjalankan algoritma *divide and conquer* melalui GUI menggunakan library tkinter.

	Prosedur generate_array() digunakan untuk mengisi variable global 'Array' dengan array dari koordinat.
	Prosedur calculate_BF() digunakan saat button BruteForce ditekan dan memberikan output yang diminta menggunakan algoritma *brute force.*
	Prosedur calculate_DnC() digunakan saat button Divide&Conquer ditekan dan memberikan output yang diminta menggunakan algoritma *divide and conquer.*
	Fungsi validate_int() digunakan untuk memvalidasi input user pada GUI agar hanya dapat memasukkan integer.

```python
def show_results(point1,point2,dist,algorithm,time,eucledian):
    point_1 = pointToStr(point1)
    point_2 = pointToStr(point2)
    result_p1 = Label(window,
                      text = point_1,
                      font=("Arial Black",10),
                      justify="left",
                      bg="#2E2E48",
                      fg="#DCFFC1")
    result_p1.place(
    x=450.0,
    y=414.0,
    )
    result_p2 = Label(window,
                      text = point_2,
                      font=("Arial Black",10),
                      justify="left",
                      bg="#2E2E48",
                      fg="#DCFFC1")
    result_p2.place(
    x=450.0,
    y=436.0,
    )
    result_dist = Label(window,
                      text = str(round(dist,2)),
                      font=("Arial Black",10),
                      justify="center",
                      bg="#2E2E48",
                      fg="#DCFFC1")
    result_dist.place(
    x=505.0,
    y=485.0,
    )
    result_algorithm = Label(window,
                      text = algorithm,
                      font=("Arial Black",10),
                      justify="center",
                      bg="#2E2E48",
                      fg="#DCFFC1")
    result_algorithm.place(
    x=490.0,
    y=585.0,
    )
```

```python
    executed = Label(window,
                      text = str(round(time,2)) + " ms",
                      font=("Arial Black",10),
                      anchor="w",
                      bg="#2E2E48",
                      fg="#DCFFC1")
    executed.place(
    x=500.0,
    y=649.0,
    )
    eucledianCount = Label(window,
                      text = eucledian,
                      font=("Arial Black",10),
                      justify="center",
                      bg="#2E2E48",
                      fg="#DCFFC1")
    eucledianCount.place(
    x=510.0,
    y=530.0,
    )

def show_plot(arrayCoordinate, c1, c2):
    # c1 dan c2 adalah 2 koordinat yang akan digaris
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    # Masukin semua koordinat ke diagram
    for i in range(len(arrayCoordinate)):
        x, y, z = arrayCoordinate[i]
        ax.scatter(x, y, z, c='g',s=1)

    # Buat garis diantara c1 dan c2
    x1, y1, z1 = c1
    x2, y2, z2 = c2
    ax.scatter(x1, y1, z1, c='r',s=10)
    ax.scatter(x2, y2, z2, c='r',s=10)
    ax.plot([x1, x2], [y1, y2], [z1, z2], c='r')

    ax.set_xlabel('Sumbu X')
    ax.set_ylabel('Sumbu Y')
    ax.set_zlabel('Sumbu Z')

    canvas_figure = FigureCanvasTkAgg(fig, master=canvas)
    canvas_figure.draw()
    canvas_figure.get_tk_widget().place(x=640, y=87, width=600, height=600)

    canvas.graph = canvas_figure
```

```python
def show_message(message,color):
    messageShow = Label(window,
                      text = message,
                      font=("Arial Black",10),
                      anchor="w",
                      justify="left",
                      bg="#393952",
                      fg=color)
    messageShow.place(
    x=450.0,
    y=250.0,
)

def clear_label(window):
    for child in window.winfo_children():
        if isinstance(child, Label):
            child.destroy()
```

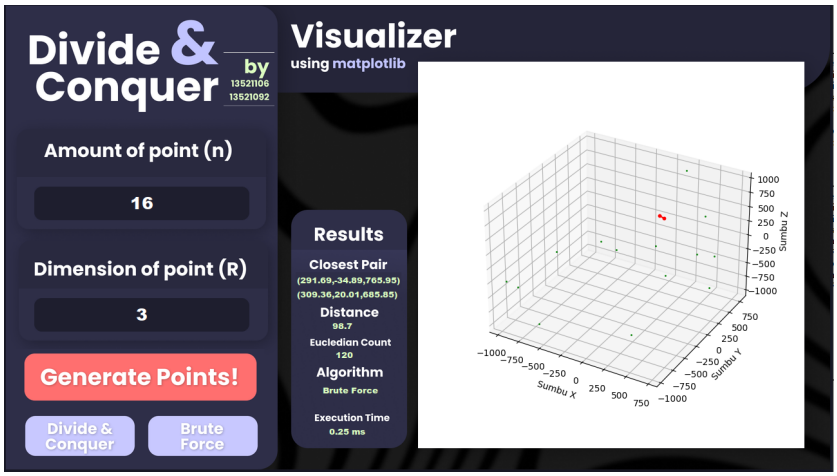
Prosedur show_result(), show_plot(), show_message() masing-masing digunakan untuk menampilkan data hasil (execute time, hasil perhitungan algoritma), menampilkan grafik, dan menampilkan pesan custom pada GUI.

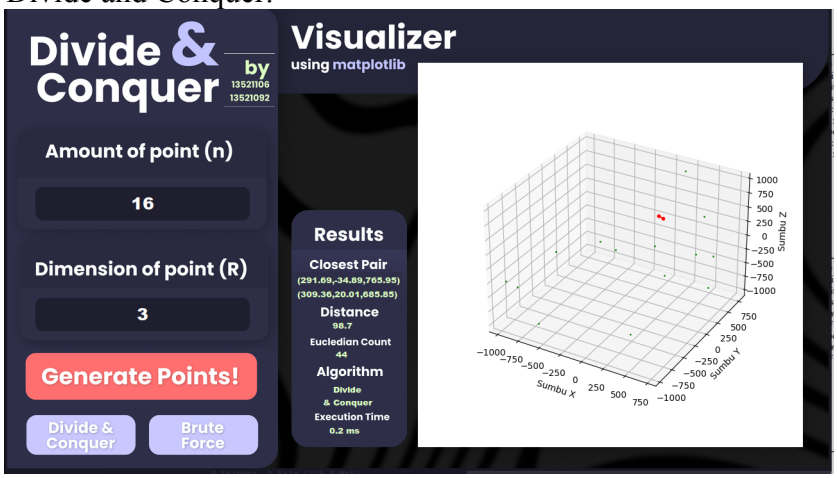
Prosedur clear_label() digunakan untuk menghapus semua output temporary pada GUI.

BAB V

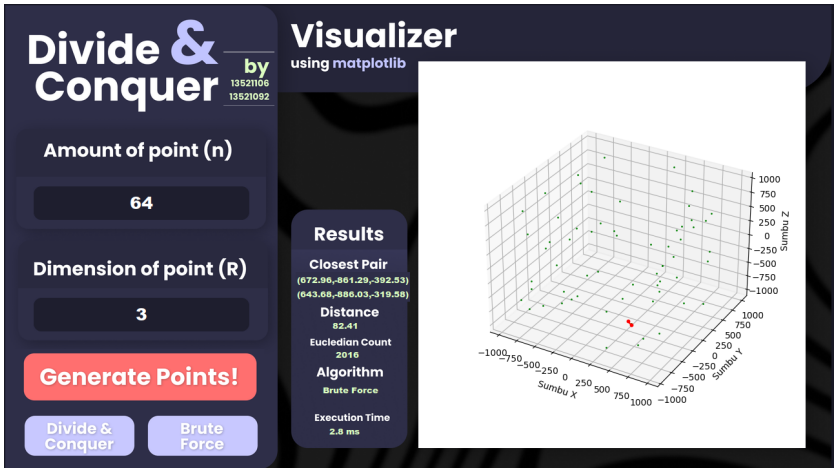
EKSPERIMEN

A. Test Case

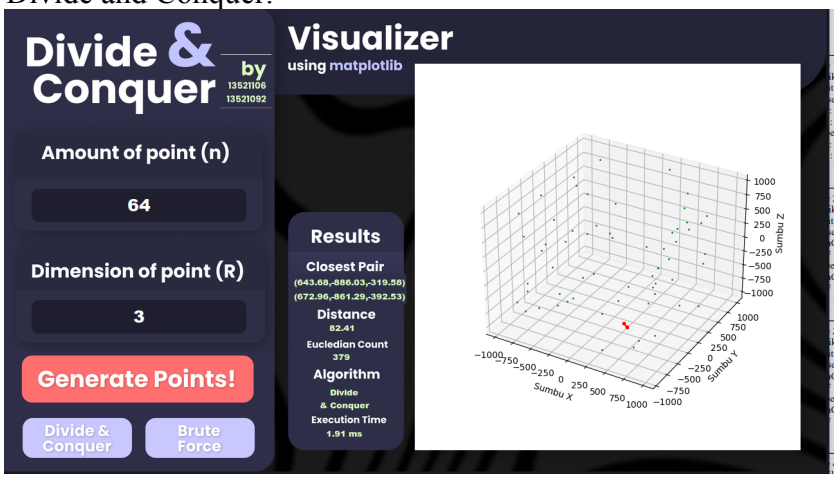
<table>
<tr>
<td>Brute Force:
 

Divide and Conquer:
 

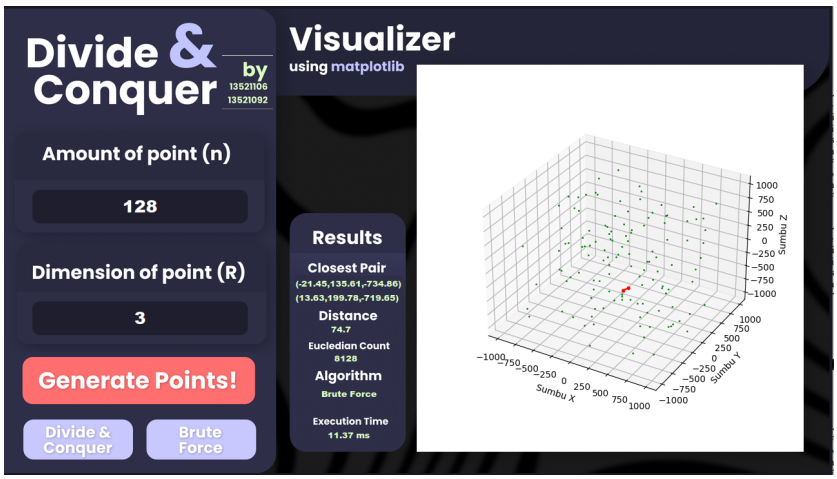
</td>
<td>Test Case 1
Jumlah titik (n) : 16
Dimensi titik (R) : 3
Waktu eksekusi :
• DnC : 0.2 ms
• BF : 0.25 ms
Jumlah operasi Euclidean:
• DnC : 44
• BF : 120
Apakah hasil pasangan sama?</td>
</tr>
<tr>
<td>Brute Force:</td>
<td>Test Case 2
Jumlah titik (n) : 64
Dimensi titik (R) : 3
Waktu eksekusi :
• DnC : 1.91 ms
• BF : 2.8 ms
Jumlah operasi Euclidean:
• DnC : 379
• BF : 2016</td>
</tr>
</table>



Divide and Conquer:



Brute Force:



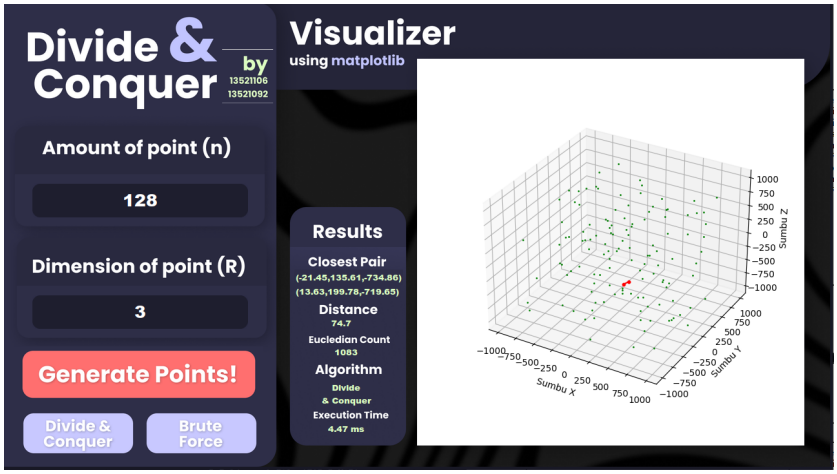
Divide and Conquer:

Test Case 3
Jumlah titik (n) : 128
Dimensi titik (R) : 3
Waktu eksekusi :

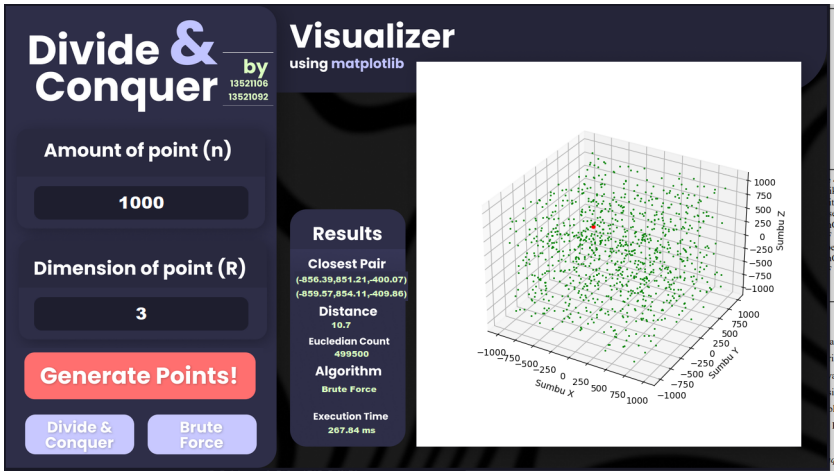
- DnC : 4.47 ms
- BF : 11.37 ms

Jumlah operasi Euclidean:

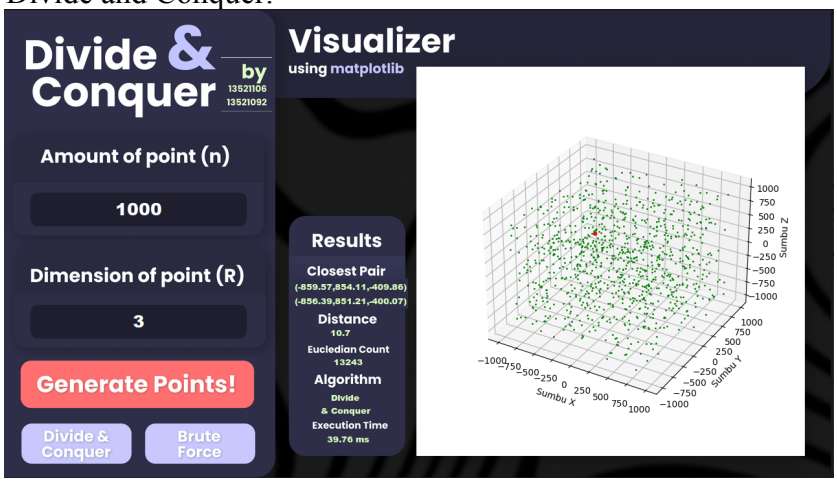
- DnC : 1083
- BF : 8128



Brute Force:



Divide and Conquer:



Test Case 4
Jumlah titik (n) : 1000
Dimensi titik (R) : 3
Waktu eksekusi :

- DnC : 39.76 ms
- BF : 267.84ms

Jumlah operasi Euclidean:

- DnC : 13243
- BF :499500

B. Analisis

Dari percobaan yang dilakukan, dapat dilihat bahwa untuk titik dengan jumlah dibawah sama dengan 128 titik, algoritma *Divide and Conquer* masih dapat memberikan solusi *closest pair* jika dibandingkan dengan algoritma *Brute Force*. Namun, untuk titik yang berada di atas 128, misalnya 1000, algoritma *Divide and Conquer* mulai memberikan solusi *closest pair* yang berbeda jika dibandingkan dengan algoritma *Brute Force*. Hal ini disebabkan oleh semakin padatnya volume ruangan yang diisi titik yang digenerate secara random. Pada program ini, penulis memasang range angka random dari -1000 sampai 1000. Oleh karena itu, galat perhitungan akan semakin tinggi jika menggunakan algoritma *Divide and Conquer*.

Di lain sisi, jumlah operasi Euclidean pada algoritma *Divide and Conquer* jauh lebih sedikit dibandingkan dengan jumlah operasi Euclidean pada algoritma *Brute Force*. Hal ini disebabkan karena algoritma *Divide and Conquer* memiliki teknik pencarian yang tidak mengharuskan algoritma untuk mengecek seluruh kemungkinan pasangan. Hal tersebut juga membuat waktu eksekusi menjadi lebih cepat dengan perbandingan waktu eksekusi algoritma *Divide and Conquer* dan *Brute Force* sekitar 1:2-6 dengan ukuran titik 100-1000 dan berada pada ruang vektor R^3.

BAB VI
KESIMPULAN

Pada tugas ini telah dibuat sebuah program penyelesaian persoalan mencari pasangan titik terdekat dengan bahasa pemrograman Python. Dalam pengerjaannya digunakan Algoritma *Divide and Conquer* untuk mencari jawaban dari persoalan tersebut dan membandingkan hasilnya dengan hasil dari Algoritma *Brute Force*. Dari percobaan yang dilakukan, dapat dilihat bahwa untuk titik dengan jumlah dibawah sama dengan 128 titik, Algoritma *Divide and Conquer* masih dapat memberikan solusi *closest pair* jika dibandingkan dengan Algoritma *Brute Force*. Namun, untuk titik yang berada di atas 128, misalnya 1000, Algoritma *Divide and Conquer* mulai memberikan solusi *closest pair* yang berbeda jika dibandingkan dengan Algoritma *Brute Force*.

Selain itu, dalam pengerjaan Tugas Kecil 2 ini, penulis mendapatkan banyak ilmu yang baru dalam pengerjaannya, seperti mendapatkan *insight* mengenai aplikasi dari algoritma *divide and conquer*, pemanfaatan berbagai *library* Python serta pengalaman membangun GUI desktop menggunakan *library* tkinter Python. Harapan penulis untuk tugas kedepannya agar diberikan waktu pengerjaan yang lebih lapang agar dapat memberikan hasil yang lebih maksimal.

Daftar Referensi

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf [diakses pada 28/02/2023]

Lampiran

Link Repository Github: https://github.com/farhanfahreezy/Tucil2_13521092_13521106

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa ada kesalahan.	V	
2. Program berhasil running	V	
3. Program dapat menerima masukan dan menuliskan luaran.	V	
4. Luaran program sudah benar (solusi closest pair benar)	V	
5. Bonus 1 dikerjakan	V	
6. Bonus 2 dikerjakan	V	