Tugas Kecil IF2211 Strategi Algoritma

Laporan Mencari Pasangan Titik Terdekat 3D dengan Algoritma *Divide and Conquer*

Oleh:

Ghazi Akmal Fauzan

K02 / 13521058

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

A. Algoritma Divide and Conquer

Secara singkat, algoritma *divide and conquer* memiliki prinsip memecah masalah yang ada menjadi beberapa bagian kecil sehingga lebih mudah untuk diselesaikan. Langkah-langkah umum dari algoritma *divide and conquer* adalah:

1. ***Divide*** : Membagi masalah menjadi beberapa sub-masalah yang mirip dengan masalah awal tetapi ukurannya lebih kecil (idealnya hampir sama besar).
2. ***Conquer*** : Memecahkan (resolve) setiap sub-masalah (secara rekursif).
3. ***Combine*** : Menggabungkan solusi dari setiap masalah sehingga membentuk solusi masalah yang asli.

Pada tugas kecil ini, penulis ditugaskan untuk mengembangkan sebuah algoritma untuk menemukan pasangan titik terdekat pada bidang 3D. Misalkan ada n titik dalam ruang 3D. Setiap titik P dalam ruang diwakili oleh koordinat P = (x, y, z). Temukan sepasang titik yang jarak terpendek satu sama lain.

Pertama-tama, list titik-titik akan disorting secara berurut meningkat dengan prosedur sort. Kemudian, algoritma *divide and conquer* akan memeriksa jumlah titik yang diberikan, apabila hanya terdiri dari 2 titik maka jarak antara kedua titik tersebut akan langsung dihitung menggunakan rumus jarak (basis genap). Apabila terdapat 3 titik, algoritma akan melakukan perbandingan jarak antara ketiga titik menggunakan metode *brute force* (basis ganjil).

Apabila terdapat lebih dari 3 titik, algoritma akan membagi titik-titik menjadi dua bagian sama besar hingga hanya terdapat dua atau tiga titik. Kemudian akan dicari jarak terdekat pada setiap bagian dengan memanggil secara rekursif algoritma *divide and conquer*. Lalu jarak terdekat pada kedua bagian akan dibandingkan dan titik-titik yang menjadi pasangan terdekat akan terpilih.

Terakhir, algoritma akan mencari jarak terdekat antara titik-titik yang berada di dekat bidang pemisah (strip). Hal itu dilakukan dengan mengumpulkan titik-titik pada suatu daerah yang memiliki jarak mendekati jarak minimum. Kemudian, jarak antara setiap pasang titik pada daerah tersebut akan dibandingkan dengan metode *brute force*. Apabila ditemukan jarak terdekat yang lebih kecil daripada jarak terdekat sebelumnya, maka pasangan titik baru tersebut akan dipilih sebagai pasangan jarak terdekat.

B. Algoritma Brute Force

Algoritma *brute force* adalah suatu pendekatan yang lempang (*straightforward*) untuk memecahkan suatu persoalan. Algoritma ini biasanya didasarkan pada pernyataan pada persoalan (*problem statement*) dan definisi konsep yang dilibatkan. Algoritma *brute force* memecahkan persoalan dengan sangat sederhana, langsung, dan dengan cara yang jelas (*obvious way*). Karakteristik algoritma *brute force* umumnya tidak cerdas, karena membutuhkan jumlah langkah yang besar dalam penyelesaiannya. Algoritma *brute force* dapat disebut juga algoritma naif (*naïve algorithm*).

Dalam tugas kecil kali ini algoritma *brute force* diimplementasikan sebagai perbandingan dengan algoritma *divide and conquer*. Secara singkat, algoritma *brute force* ini menghitung jarak setiap pasang titik yang ada dan menyimpan pasangan titik dengan jarak terdekat.

C. Source Program

1. bruteforce.py
   a. Setup

```python
from utility import distance
```

   b. bruteforce(points)

```python
# Brute force algorithm
def bruteforce(points):
    minDistance = distance(points[0], points[1])
    point1 = points[0]
    point2 = points[1]
    nCalculation = 0
    for i in range(len(points)):
        for j in range(i+1, len(points)):
            dis = distance(points[i], points[j])
            nCalculation += 1
            if dis < minDistance:
                minDistance = dis
                point1 = points[i]
                point2 = points[j]
    return minDistance, point1, point2, nCalculation
```

2. colors.py

```python
# Color and UI codes for terminal output

BLACK = "\033[0;30m"
RED = "\033[0;31m"
GREEN = "\033[0;32m"
BROWN = "\033[0;33m"
BLUE = "\033[0;34m"
PURPLE = "\033[0;35m"
CYAN = "\033[0;36m"
LIGHT_GRAY = "\033[0;37m"
DARK_GRAY = "\033[1;30m"
LIGHT_RED = "\033[1;31m"
LIGHT_GREEN = "\033[1;32m"
YELLOW = "\033[1;33m"
LIGHT_BLUE = "\033[1;34m"
LIGHT_PURPLE = "\033[1;35m"
LIGHT_CYAN = "\033[1;36m"
WHITE = "\033[1;37m"
BOLD = "\033[1m"
FAINT = "\033[2m"
ITALIC = "\033[3m"
UNDERLINE = "\033[4m"
BLINK = "\033[5m"
NEGATIVE = "\033[7m"
CROSSED = "\033[9m"
RESET = "\033[0m"
```

3. command.py
    a. Setup

```
from colors import *
```

    b. commandStart()

```
# Start/Exit
def commandStart():
    print(WHITE     + "═══════════════════════════════════════════")
    print(LIGHT_RED + "|                START/EXIT                |")
    print(WHITE     + "═══════════════════════════════════════════")
    print(LIGHT_RED + "1." + WHITE + " START")
    print(LIGHT_RED + "2." + WHITE + " EXIT")
```

    c. commandAlgorithm()

```
# Pick algorithm
def commandAlgorithm():
    print(WHITE     + "═══════════════════════════════════════════")
    print(LIGHT_RED + "|              PICK ALGORITHM              |")
    print(WHITE     + "═══════════════════════════════════════════")
    print(LIGHT_RED + "1." + WHITE + " BRUTE FORCE")
    print(LIGHT_RED + "2." + WHITE + " DIVIDE AND CONQUER")
    print(LIGHT_RED + "3." + WHITE + " BOTH")
```

    d. commandInputOption()

```
# Input options
def commandInputOption():
    print(WHITE     + "═══════════════════════════════════════════")
    print(LIGHT_RED + "|              INPUT OPTIONS               |")
    print(WHITE     + "═══════════════════════════════════════════")
    print(LIGHT_RED + "1." + WHITE + " RANDOM")
    print(LIGHT_RED + "2." + WHITE + " MANUAL")
    print(LIGHT_RED + "3." + WHITE + " FILE")
```

    e. commandSave()

```
# Save solution
def commandSave():
    print(WHITE     + "═══════════════════════════════════════════")
    print(LIGHT_RED + "|              SAVE SOLUTION?              |")
    print(WHITE     + "═══════════════════════════════════════════")
    print(LIGHT_RED + "1." + WHITE + " YES")
    print(LIGHT_RED + "2." + WHITE + " NO")
```

    f. commandInput1()

```
# 1/2 Input command
def commandInput1():
    while (True):
        Input = input(WHITE + ">> " + RESET)
        try:
            Input = int(Input)
            if (Input == 1 or Input == 2):
                break
            else:
                print(LIGHT_RED + "\nPlease enter a valid input! (1/2)" + RESET)
        except ValueError:
            print(LIGHT_RED + "\nInput is not an integer! Please re-enter." + RESET)

    return Input
```

g. commandInput2()

```python
# 1/2/3 Input command
def commandInput2():
    while (True):
        Input = input(WHITE + ">> " + RESET)
        try:
            Input = int(Input)
            if (Input == 1 or Input == 2 or Input == 3):
                break
            else:
                print(LIGHT_RED + "\nPlease enter a valid input! (1/2/3)" + RESET)
        except ValueError:
            print(LIGHT_RED + "\nInput is not an integer! Please re-enter." + RESET)

    return Input
```

h. pointInput()

```python
# nPoint and dimension input
def pointInput():
    while (True):
        nPoint = input(WHITE + "\nEnter the number of points: " + RESET)
        try:
            nPoint = int(nPoint)
            if nPoint < 2:
                print(LIGHT_RED + "\nThe minimum number of points is 2! Please re-enter." + RESET)
            else:
                break
        except ValueError:
            print(LIGHT_RED + "\nInput is not an integer! Please re-enter." + RESET)

    while (True):
        dimension = input(WHITE + "Enter the number of dimensions: " + RESET)
        try:
            dimension = int(dimension)
            if dimension < 1:
                print(LIGHT_RED + "\nThe minimum number of dimensions is 1! Please re-enter.\n" + RESET)
            else:
                break
        except ValueError:
            print(LIGHT_RED + "\nInput is not a number! Please re-enter.\n" + RESET)

    return nPoint, dimension
```

4. dividenconquer.py

a. Setup

```python
from utility import distance
```

b. dividenconquer(points, nCalculation = 0)

```python
# Divide and conquer algorithm
def dividenconquer(points, nCalculation = 0):
    # Calculate the distance between two points if there are 2 points
    if len(points) == 2:
        nCalculation += 1
        return distance(points[0], points[1]), points[0], points[1], nCalculation

    # Brute force if there are 3 points
    elif len(points) == 3:
        minDistance = distance(points[0], points[1])
        point1 = points[0]
        point2 = points[1]
        for i in range(len(points)):
            for j in range(i+1, len(points)):
                dis = distance(points[i], points[j])
                nCalculation += 1
                if dis < minDistance:
                    minDistance = dis
                    point1 = points[i]
                    point2 = points[j]
        return minDistance, point1, point2, nCalculation
```

```python
    # Recursive if there are more than 3 points
    else:
        mid = len(points) // 2
        leftMinDistance, leftPoint1, leftPoint2, nCalculation = dividenconquer(points[:mid], nCalculation)
        rightMinDistance, rightPoint1, rightPoint2, nCalculation = dividenconquer(points[mid:], nCalculation)
        if leftMinDistance < rightMinDistance:
            minDistance = leftMinDistance
            point1 = leftPoint1
            point2 = leftPoint2
        else:
            minDistance = rightMinDistance
            point1 = rightPoint1
            point2 = rightPoint2

        midPoint = points[mid][0]
        midPoints = []
        for i in range(len(points)):
            if (midPoint - minDistance) < points[i][0] < (midPoint + minDistance):
                midPoints.append(points[i])

        for i in range(len(midPoints)):
            for j in range(i+1, len(midPoints)):
                calculate = True
                for k in range(len(midPoints[i])):
                    if abs(midPoints[i][k] - midPoints[j][k]) > minDistance:
                        calculate = False
                        break
                if calculate:
                    dis = distance(midPoints[i], midPoints[j])
                    nCalculation += 1
                    if dis < minDistance:
                        minDistance = dis
                        point1 = midPoints[i]
                        point2 = midPoints[j]

        return minDistance, point1, point2, nCalculation
```

5. main.py
   a. Setup

```python
import os
import time
import random
import platform

from colors import *
from splash import *
from command import *
from plot import *
from utility import *
from dividenconquer import *
from bruteforce import *
```

   b. main()

```python
def main():
    # Show splash screen
    splash()

    # Start program
    commandStart()
    process = commandInput1()

    # Loop until user exit
    while (process == 1):
        # Pick input option
        print("")
        commandInputOption()
        option = commandInput2()
        points = []

        # Option 1: Random input
        if (option == 1):
            nPoint, dimension = pointInput()
            for i in range(nPoint):
                point = []
                for j in range(dimension):
                    point.append(random.uniform(lowerLimit, upperLimit))
                points.append(point)

        # Option 2: Manual input
        elif (option == 2):
            nPoint, dimension = pointInput()
            print("")
            for i in range(nPoint):
                point = []
                for j in range(dimension):
                    while True:
                        inputNumber = input(WHITE + "Input Point " + str(i+1) + " Dimension " + str(j+1) + ": " + RESET)
                        try:
                            inputNumber = float(inputNumber)
                            point.append(inputNumber)
                            break
                        except ValueError:
                            print(LIGHT_RED + "\nInput is not a number! Please re-enter.\n" + RESET)
                points.append(point)
```

```
    # Option 3: File input
    else:
        print("")
        while (True):
            print(LIGHT_GREEN + "Input format (example.txt). Put inside 'test' folder." + RESET)
            fileName = input(WHITE + "Input Filename: " + RESET)
            if os.path.isfile("test/" + fileName):
                file = open("test/" + fileName, "r")
                for line in file:
                    point = []
                    for i in line.split():
                        point.append(float(i))
                    points.append(point)
                file.close()
                break
            else:
                print(LIGHT_RED + "\nFile not found! Please re-enter.\n" + RESET)

    # Pick algorithm
    print("")
    commandAlgorithm()
    algorithm = commandInput2()

    # Brute Force
    if (algorithm == 1) or (algorithm == 3):
        # Start timer
        startBF = time.time()

        # Main algorithm
        pointsBF = sort(points)
        minDistanceBF, point1BF, point2BF, nCalculationBF = bruteforce(pointsBF)

        # End timer
        endBF = time.time()

        # Display result
        print(WHITE + "\n==========" + LIGHT_RED + " BRUTE FORCE " + WHITE + "==========")
        print(WHITE + "Two points with shortest distance: ")
        print(WHITE + "Point 1: " + YELLOW + ", ".join("{:.2f}".format(p) for p in point1BF))
        print(WHITE + "Point 2: " + YELLOW + ", ".join("{:.2f}".format(p) for p in point2BF))
        print(WHITE + "Distance: " + YELLOW + "{:.2f}".format(minDistanceBF))
        print(WHITE + "Number of calculation: " + YELLOW + "{}".format(nCalculationBF))
        print(WHITE + "Execution time: " + YELLOW + "{:.2f} ms".format((endBF - startBF) * 1000))
        print(WHITE + "Processor: " + YELLOW + "{}".format(platform.processor()) + RESET)
        plot("BRUTE FORCE", pointsBF, point1BF, point2BF, None)
```

```
    # Divide and Conquer
    if (algorithm == 2) or (algorithm == 3):
        # Start timer
        startDnC = time.time()

        # Main algorithm
        pointsDnC = sort(points)
        minDistanceDnC, point1DnC, point2DnC, nCalculationDnC = dividenconquer(pointsDnC)

        # End timer
        endDnC = time.time()

        # Display result
        print(WHITE + "\n==========" + LIGHT_RED + " DIVIDE AND CONQUER " + WHITE + "==========")
        print(WHITE + "Two points with shortest distance:")
        print(WHITE + "Point 1: " + YELLOW + ", ".join("{:.2f}".format(p) for p in point1DnC))
        print(WHITE + "Point 2: " + YELLOW + ", ".join("{:.2f}".format(p) for p in point2DnC))
        print(WHITE + "Distance: " + YELLOW + "{:.2f}".format(minDistanceDnC))
        print(WHITE + "Number of calculation: " + YELLOW + "{}".format(nCalculationDnC))
        print(WHITE + "Execution time: " + YELLOW + "{:.2f} ms".format((endDnC - startDnC) * 1000))
        print(WHITE + "Processor: " + YELLOW + "{}".format(platform.processor()) + RESET)
        plot("DIVIDE AND CONQUER", pointsDnC, point1DnC, point2DnC, None)
```

```
        # Save option
        print("")
        commandSave()
        save = commandInput1()

        # Save to file
        if (save == 1):
            saveConfig = input(str(WHITE + "\nInput Filename: " + RESET))

            if not os.path.exists("test"):
                os.mkdir("test")
            if not os.path.exists("test/" + saveConfig):
                os.mkdir("test/" + saveConfig)

            # Save Brute Force
            if (algorithm == 1):
                with open("test/" + saveConfig + "/" + saveConfig + ".txt", "w") as f:
                    f.write("Points:\n")
                    for i in range(len(pointsBF)):
                        f.write("(" + ", ".join("{:.2f}".format(p) for p in pointsBF[i]) + ")\n")
                    f.write("\n==========" + " BRUTE FORCE " + "==========\n")
                    f.write("Two points with shortest distance: \n")
                    f.write("Point 1: " + ", ".join("{:.2f}".format(p) for p in point1BF) + "\n")
                    f.write("Point 2: " + ", ".join("{:.2f}".format(p) for p in point2BF) + "\n")
                    f.write("Distance: " + "{:.2f}".format(minDistanceBF) + "\n")
                    f.write("Number of calculation: " + "{}".format(nCalculationBF) + "\n")
                    f.write("Execution time: " + "{:.2f} ms".format((endBF - startBF) * 1000) + "\n")
                    f.write("Processor: " + "{}".format(platform.processor()) + "\n")
                plot("BRUTE FORCE", pointsBF, point1BF, point2BF, "test/" + saveConfig + "/" + saveConfig + ".png")

            # Save Divide and Conquer
            elif (algorithm == 2):
                with open("test/" + saveConfig + "/" + saveConfig + ".txt", "a") as f:
                    f.write("Points:\n")
                    for i in range(len(pointsDnC)):
                        f.write("(" + ", ".join("{:.2f}".format(p) for p in pointsDnC[i]) + ")\n")
                    f.write("\n==========" + " DIVIDE AND CONQUER " + "==========\n")
                    f.write("Two points with shortest distance:\n")
                    f.write("Point 1: " + ", ".join("{:.2f}".format(p) for p in point1DnC) + "\n")
                    f.write("Point 2: " + ", ".join("{:.2f}".format(p) for p in point2DnC) + "\n")
                    f.write("Distance: " + "{:.2f}".format(minDistanceDnC) + "\n")
                    f.write("Number of calculation: " + "{}".format(nCalculationDnC) + "\n")
                    f.write("Execution time: " + "{:.2f} ms".format((endDnC - startDnC) * 1000) + "\n")
                    f.write("Processor: " + "{}".format(platform.processor()) + "\n")
                plot("DIVIDE AND CONQUER", pointsDnC, point1DnC, point2DnC, "test/" + saveConfig + "/" + saveConfig + ".png")
```

```
            # Save both
            else:
                with open("test/" + saveConfig + "/" + saveConfig + ".txt", "w") as f:
                    f.write("Points:\n")
                    for i in range(len(points)):
                        f.write("(" + ", ".join("{:.2f}".format(p) for p in points[i]) + ")\n")

                    f.write("\n==========" + " BRUTE FORCE " + "==========\n")
                    f.write("Two points with shortest distance: \n")
                    f.write("Point 1: " + ", ".join("{:.2f}".format(p) for p in point1BF) + "\n")
                    f.write("Point 2: " + ", ".join("{:.2f}".format(p) for p in point2BF) + "\n")
                    f.write("Distance: " + "{:.2f}".format(minDistanceBF) + "\n")
                    f.write("Number of calculation: " + "{}".format(nCalculationBF) + "\n")
                    f.write("Execution time: " + "{:.2f} ms".format((endBF - startBF) * 1000) + "\n")
                    f.write("Processor: " + "{}".format(platform.processor()) + "\n")

                    f.write("\n==========" + " DIVIDE AND CONQUER " + "==========\n")
                    f.write("Two points with shortest distance:\n")
                    f.write("Point 1: " + ", ".join("{:.2f}".format(p) for p in point1DnC) + "\n")
                    f.write("Point 2: " + ", ".join("{:.2f}".format(p) for p in point2DnC) + "\n")
                    f.write("Distance: " + "{:.2f}".format(minDistanceDnC) + "\n")
                    f.write("Number of calculation: " + "{}".format(nCalculationDnC) + "\n")
                    f.write("Execution time: " + "{:.2f} ms".format((endDnC - startDnC) * 1000) + "\n")
                    f.write("Processor: " + "{}".format(platform.processor()) + "\n")
                plot("BRUTE FORCE", pointsBF, point1BF, point2BF, "test/" + saveConfig + "/" + saveConfig + "BF.png")
                plot("DIVIDE AND CONQUER", pointsDnC, point1DnC, point2DnC, "test/" + saveConfig + "/" + saveConfig + "DnC.png")

            # Display message file saved
            print(LIGHT_GREEN + "\nAdditional Information Added into txt File" + RESET)
            print(LIGHT_GREEN + "File saved!" + RESET)

        # Try again option, continue loop if yes
        print(LIGHT_GREEN + "\nDo you want to try again?\n" + RESET)
        commandStart()
        process = commandInput1()

    print(LIGHT_GREEN + "\nThank you for using Shortest Distance Solver!\n" + RESET)
```

6. plot.py
    a. Setup

```
import matplotlib.pyplot as plt
from PIL import Image, ImageDraw, ImageFont
from colors import *
```

b. plot(title, points, point1, point2, saveConfig)

```python
# Plot Matplotlib
def plot(title, points, point1, point2, saveConfig):
    x = []
    y = []
    z = []
    for point in points:
        if len(point) ≥ 1:
            x.append(point[0])
        if len(point) ≥ 2:
            y.append(point[1])
        if len(point) ≥ 3:
            z.append(point[2])

    fig = plt.figure()
    if len(point) ≤ 3:
        # 1D plot
        if len(point) == 1:
            y = [0] * len(x)
            plt.scatter(x, y, c='black', alpha=1)
            plt.scatter(point1[0], 0, c='red')
            plt.scatter(point2[0], 0, c='red')
            xLine = [point1[0], point2[0]]
            yLine = [0, 0]
            plt.plot(xLine, yLine, c='red')
            plt.title(title)

        # 2D plot
        elif len(point) == 2:
            plt.scatter(x, y, c='black', alpha=1)
            plt.scatter(point1[0], point1[1], c='red')
            plt.scatter(point2[0], point2[1], c='red')
            xLine = [point1[0], point2[0]]
            yLine = [point1[1], point2[1]]
            plt.plot(xLine, yLine, c='red')
            plt.title(title)
```

```python
        # 3D plot
        elif len(point) == 3:
            ax = fig.add_subplot(111, projection='3d')
            if (point1[0]) in x:
                x.remove(point1[0])
            if (point1[1]) in y:
                y.remove(point1[1])
            if (point1[2]) in z:
                z.remove(point1[2])
            if (point2[0]) in x:
                x.remove(point2[0])
            if (point2[1]) in y:
                y.remove(point2[1])
            if (point2[2]) in z:
                z.remove(point2[2])
            ax.scatter(x, y, z, c='black', alpha=1)
            ax.scatter(point1[0], point1[1], point1[2], c='red')
            ax.scatter(point2[0], point2[1], point2[2], c='red')

            xLine = [point1[0], point2[0]]
            yLine = [point1[1], point2[1]]
            zLine = [point1[2], point2[2]]
            ax.plot(xLine, yLine, zLine, c='red')
            ax.set_title(title)
            ax.set_xlabel('X')
            ax.set_ylabel('Y')
            ax.set_zlabel('Z')

        # Show or save plot
        if saveConfig is None:
            plt.show()
        else:
            plt.savefig(saveConfig)
```

```
    else:
        # Show or save plot not visualizable
        if saveConfig is None:
            print(WHITE + "Not visualizable!" + RESET)
        else:
            img = Image.new('RGB', (640, 480), color = 'white')

            draw = ImageDraw.Draw(img)
            text = "Not visualizable!"
            font = ImageFont.truetype('arial.ttf', size=40)
            text_width, text_height = draw.textsize(text, font)
            x = (img.width - text_width) / 2
            y = (img.height - text_height) / 2
            draw.text((x, y), text, font=font, fill='black')

            img.save(saveConfig)
```

7. splash.py
    a. Setup

```
import time
from colors import *
```

    b. splash()

```
# Splash Screen
def splash():
    print(LIGHT_GREEN)
    print("    /$$      /$$/$$$$$$$$/$$       /$$$$$$  /$$$$$$ /$$      /$$/$$$$$$$$/$$")
    print("   | $$  /$ | $| $$_____| $$      /$$__  $$/$$__  $| $$$    /$$| $$_____| $$")
    print("   | $$ /$$$| $| $$     | $$     | $$  \\__| $$  \\ $| $$$$  /$$$| $$     | $$")
    print("   | $$/$$ $$ $| $$$$$  | $$     | $$     | $$  | $| $$ $$/$$ $| $$$$$  | $$")
    print("   | $$$$_  $$$| $$__/  | $$     | $$     | $$  | $| $$  $$$| $| $$__/  |__/")
    print("   | $$$/ \\  $$| $$     | $$     | $$    $| $$  | $| $$\\  $ | $| $$        ")
    print("   | $$/   \\  $| $$$$$$$| $$$$$$$|  $$$$$$|  $$$$$$| $$ \\/  | $| $$$$$$$/$$")
    print("   |__/     \\__|________|_______/\\_____/ \\_____/|__/     |__|_______|__/")

    print(WHITE)
    print("W", end="", flush=True)
    time.sleep(0.05)
    print("e", end="", flush=True)
    time.sleep(0.05)
    print("l", end="", flush=True)
    time.sleep(0.05)
    print("c", end="", flush=True)
    time.sleep(0.05)
    print("o", end="", flush=True)
    time.sleep(0.05)
    print("m", end="", flush=True)
    time.sleep(0.05)
    print("e", end="", flush=True)
    time.sleep(0.05)
    print(" ", end="", flush=True)
    time.sleep(0.05)
    print("t", end="", flush=True)
    time.sleep(0.05)
    print("o", end="", flush=True)
    time.sleep(0.05)
    print(" ", end="", flush=True)
    time.sleep(0.05)
    print("S", end="", flush=True)
    time.sleep(0.05)
    print("h", end="", flush=True)
    time.sleep(0.05)
    print("o", end="", flush=True)
    time.sleep(0.05)
    print("r", end="", flush=True)
    time.sleep(0.05)
    print("t", end="", flush=True)
    time.sleep(0.05)
    print("e", end="", flush=True)
    time.sleep(0.05)
```

```
    print("s", end="", flush=True)
    time.sleep(0.05)
    print("t", end="", flush=True)
    time.sleep(0.05)
    print(" ", end="", flush=True)
    time.sleep(0.05)
    print("D", end="", flush=True)
    time.sleep(0.05)
    print("i", end="", flush=True)
    time.sleep(0.05)
    print("s", end="", flush=True)
    time.sleep(0.05)
    print("t", end="", flush=True)
    time.sleep(0.05)
    print("a", end="", flush=True)
    time.sleep(0.05)
    print("n", end="", flush=True)
    time.sleep(0.05)
    print("c", end="", flush=True)
    time.sleep(0.05)
    print("e", end="", flush=True)
    time.sleep(0.05)
    print(" ", end="", flush=True)
    time.sleep(0.05)
    print("S", end="", flush=True)
    time.sleep(0.05)
    print("o", end="", flush=True)
    time.sleep(0.05)
    print("l", end="", flush=True)
    time.sleep(0.05)
    print("v", end="", flush=True)
    time.sleep(0.05)
    print("e", end="", flush=True)
    time.sleep(0.05)
    print("r")
```

```
    print(YELLOW)
    print("A " + CROSSED + "Group" + RESET + YELLOW + " Solo Project")
    print("Made By " + UNDERLINE + "Ghazi Akmal Fauzan (13521058)" + RESET)

    print(LIGHT_RED)
    print("Loading", end="", flush=True)
    for i in range(3):
        print(".", end="", flush=True)
        time.sleep(1)

    print(WHITE)
    print("Solver Loaded!")
    print(RESET)
```

8. utility.py
    a. Setup

```
import math
```

b. distance(point1, point2)

```python
# Calculate the distance between two points
def distance(point1, point2):
    dis = 0
    for i in range(len(point1)):
        dis += (point1[i] - point2[i]) ** 2
    return math.sqrt(dis)
```

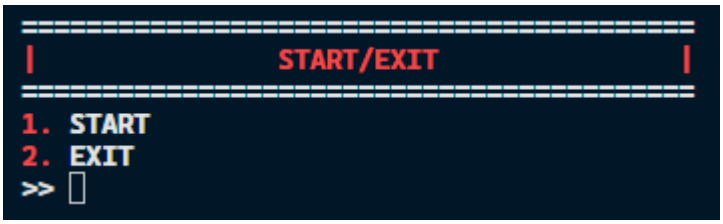
c. sort(points)

```python
# Sort the points
def sort(points):
    if len(points) <= 1:
        return points
    else:
        pivot = points[0]
        less = []
        greater = []
        for i in range(1, len(points)):
            if points[i] < pivot:
                less.append(points[i])
            else:
                greater.append(points[i])
        return sort(less) + [pivot] + sort(greater)
```

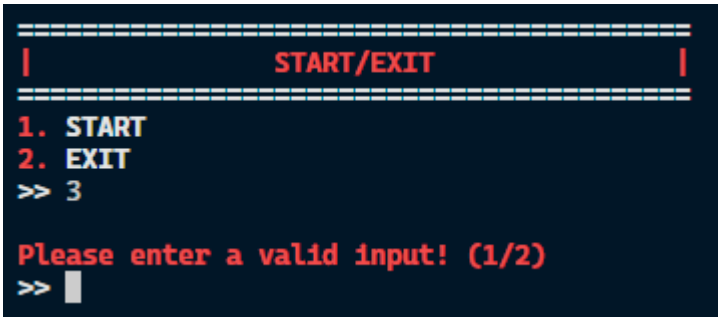
D. Input dan Output

1. Splash Screen

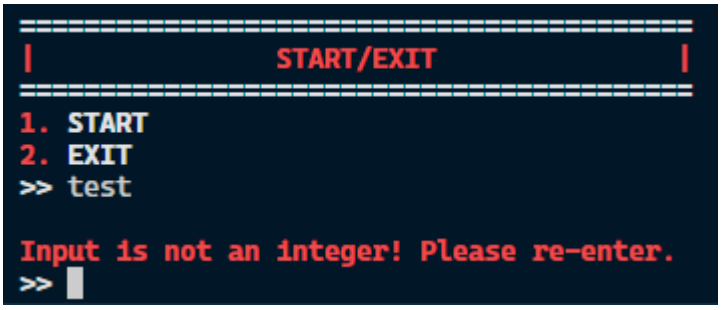
2. Start/Exit Command
    a. Tampilan awal



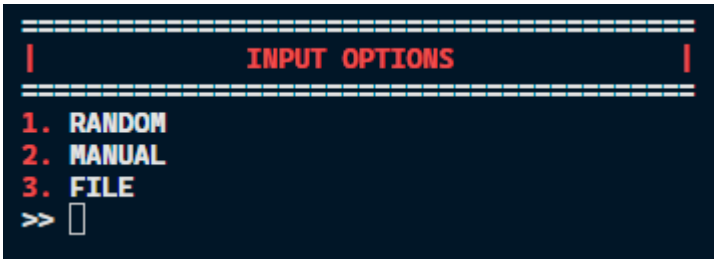
    b. Wrong input (integer)



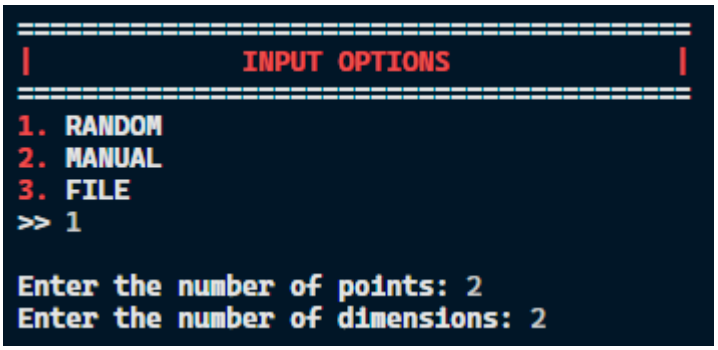
    c. Wrong input (not integer)



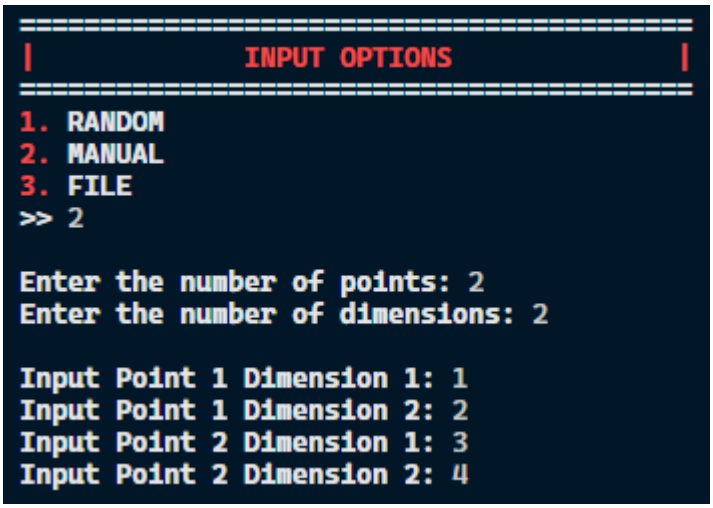
3. Input Options Command
    a. Tampilan awal



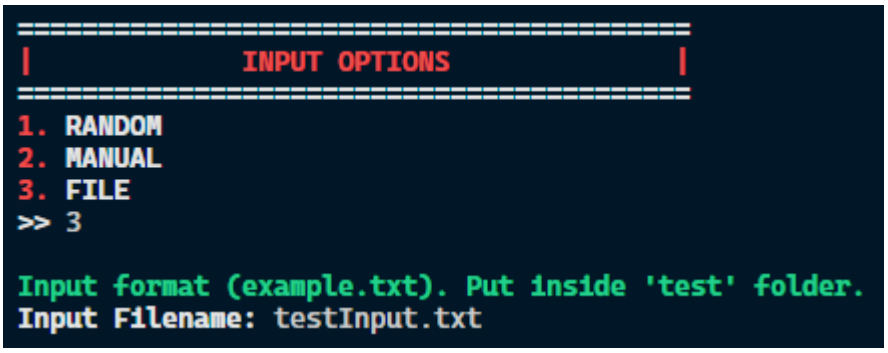
    b. Random



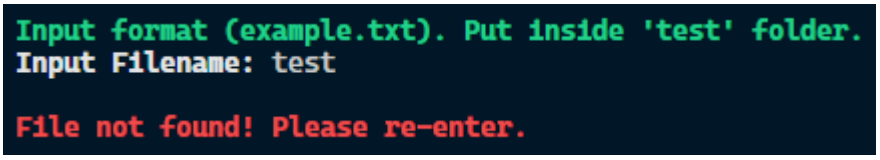
c. Manual



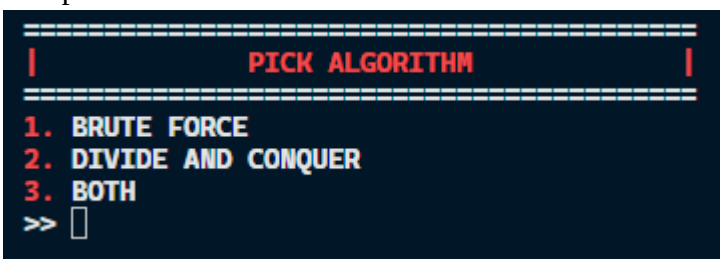
d. File



e. Wrong input (file)

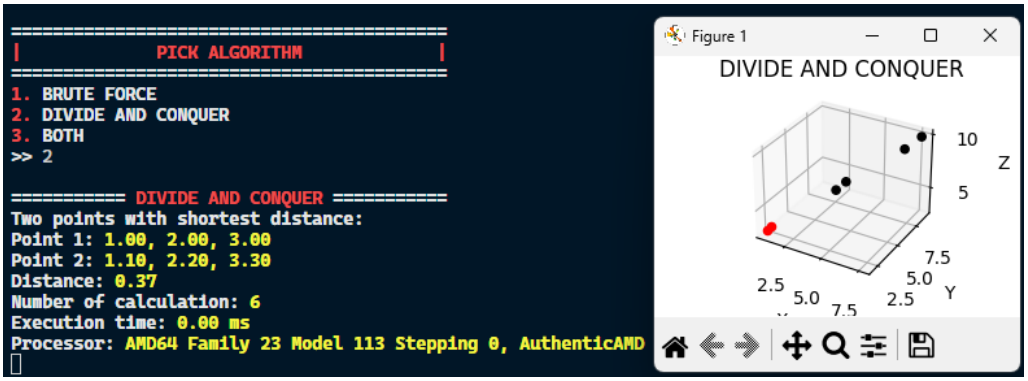


4. Pick Algorithm Command
   a. Tampilan awal

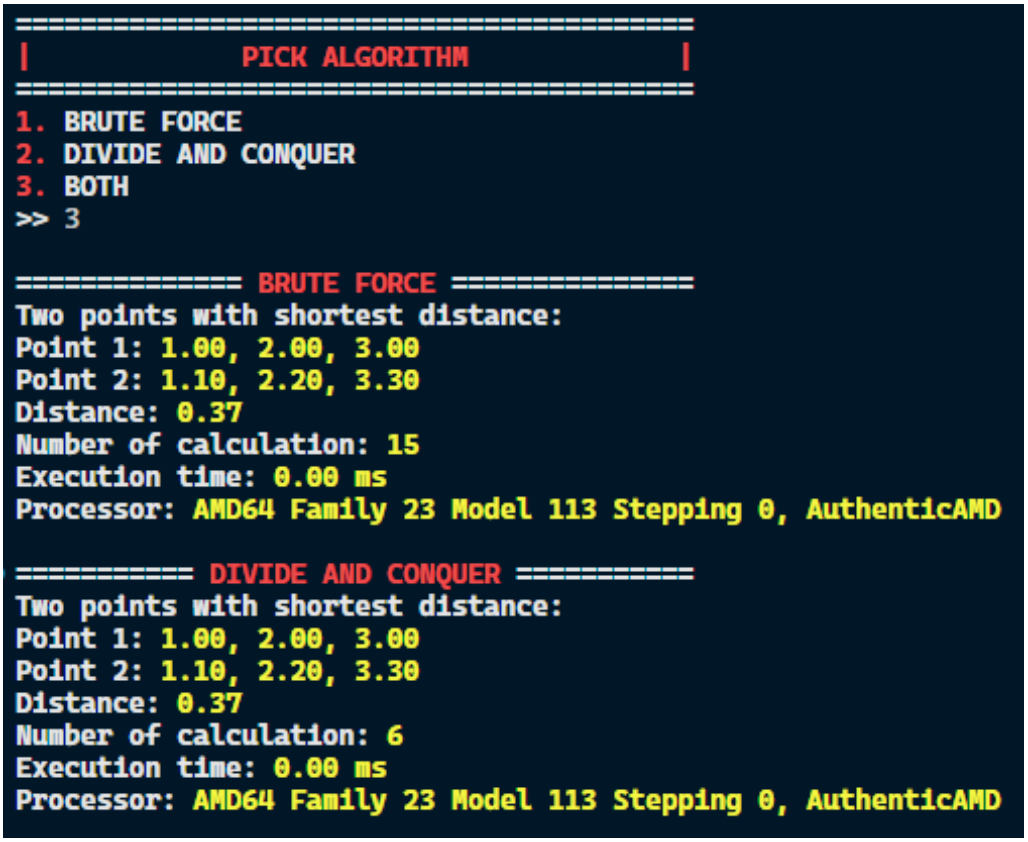


b. Brute Force

c. Divide and Conquer



d. Both



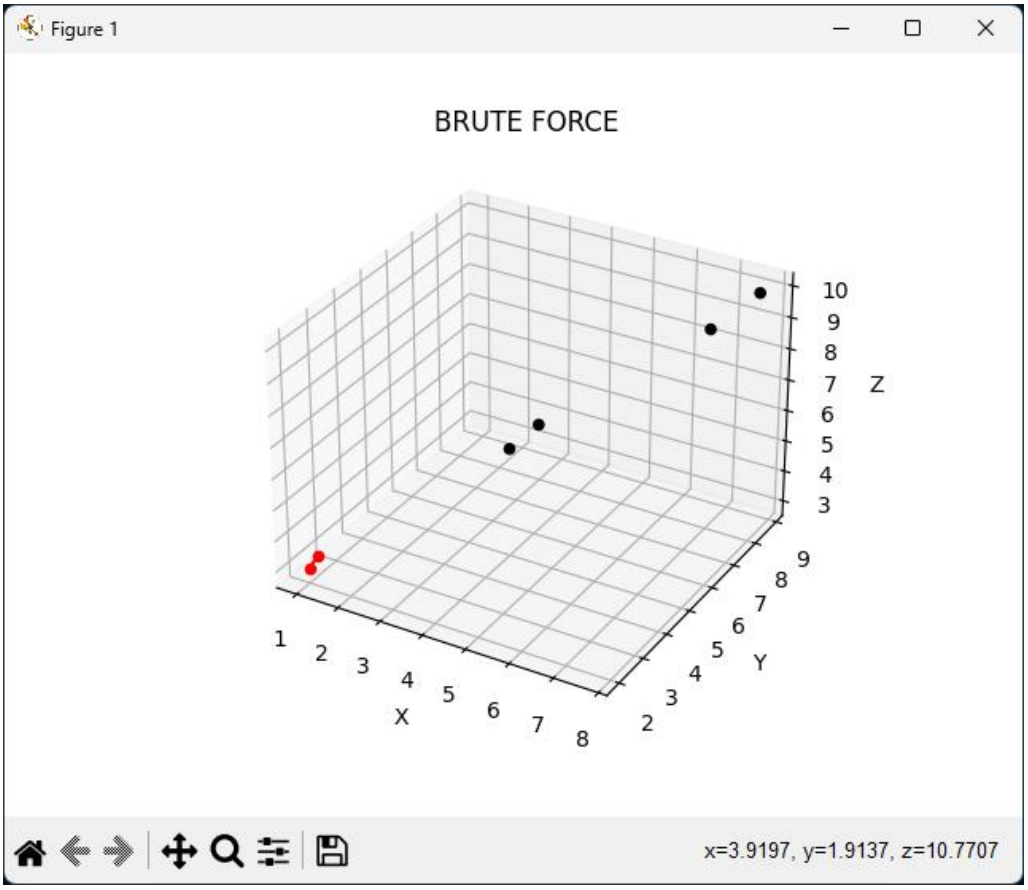


Figure 1
BRUTE FORCE
X
Y
Z
x=3.9197, y=1.9137, z=10.7707

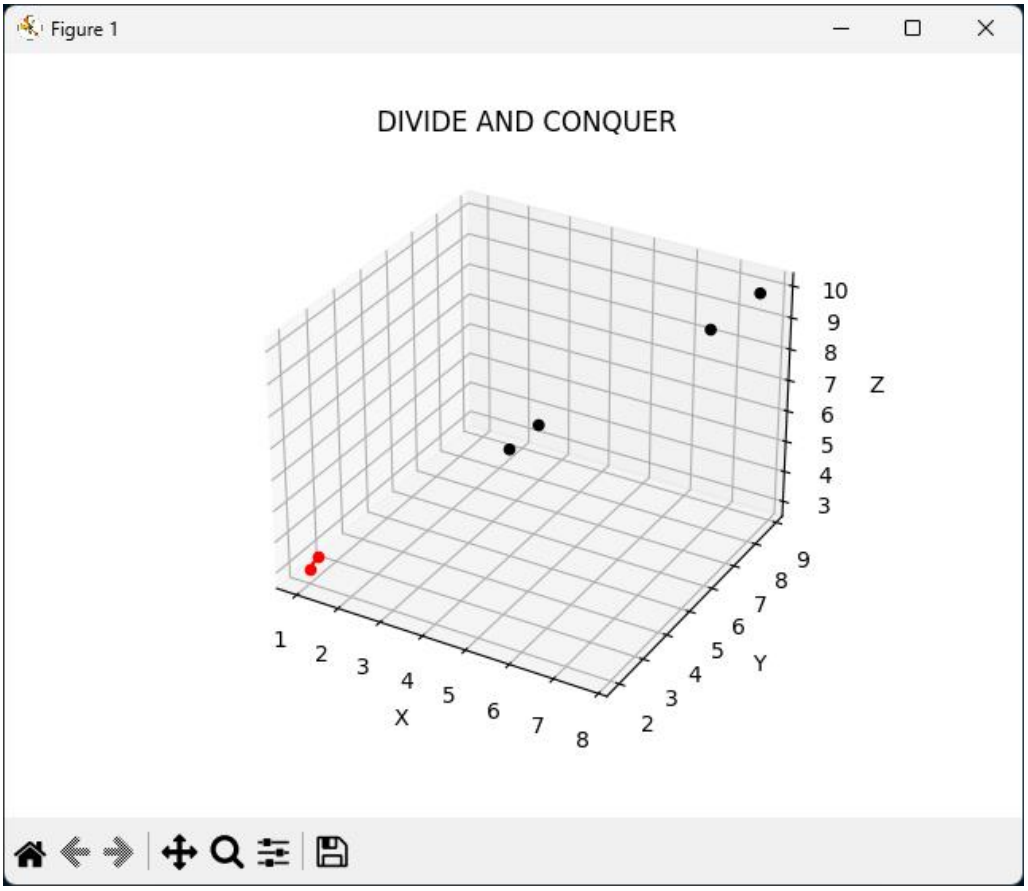


Figure 1
DIVIDE AND CONQUER
X
Y
Z

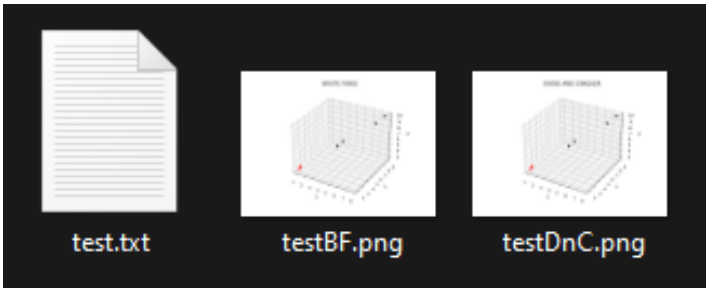
5. Save Command
   a. Tampilan awal

```
==========================================
|              SAVE SOLUTION?            |
==========================================
1. YES
2. NO
>>
```

   b. Save file (file saved)

```
==========================================
|              SAVE SOLUTION?            |
==========================================
1. YES
2. NO
>> 1

Input Filename: test

Additional Information Added into txt File
File saved!
```

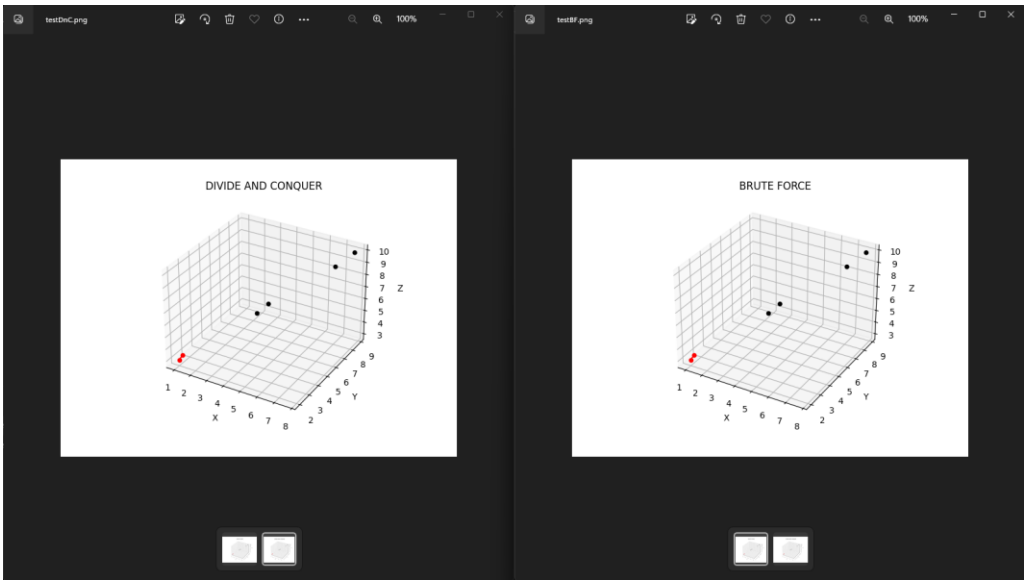
   c. Tampilan folder output (opsi algoritma ‘both’)



   d. Tampilan file txt

```
test.txt - Notepad
File  Edit  View

Points:
(1.00, 2.00, 3.00)
(4.00, 5.00, 6.00)
(7.00, 8.00, 9.00)
(1.10, 2.20, 3.30)
(4.40, 5.50, 6.60)
(7.70, 8.80, 9.90)

============== BRUTE FORCE ===============
Two points with shortest distance:
Point 1: 1.00, 2.00, 3.00
Point 2: 1.10, 2.20, 3.30
Distance: 0.37
Number of calculation: 15
Execution time: 0.00 ms
Processor: AMD64 Family 23 Model 113 Stepping 0, AuthenticAMD

=========== DIVIDE AND CONQUER ===========
Two points with shortest distance:
Point 1: 1.00, 2.00, 3.00
Point 2: 1.10, 2.20, 3.30
Distance: 0.37
Number of calculation: 6
Execution time: 0.00 ms
Processor: AMD64 Family 23 Model 113 Stepping 0, AuthenticAMD

Ln 1, Col 1   100%   Windows (CRLF)   UTF-8
```

e. Tampilan file png

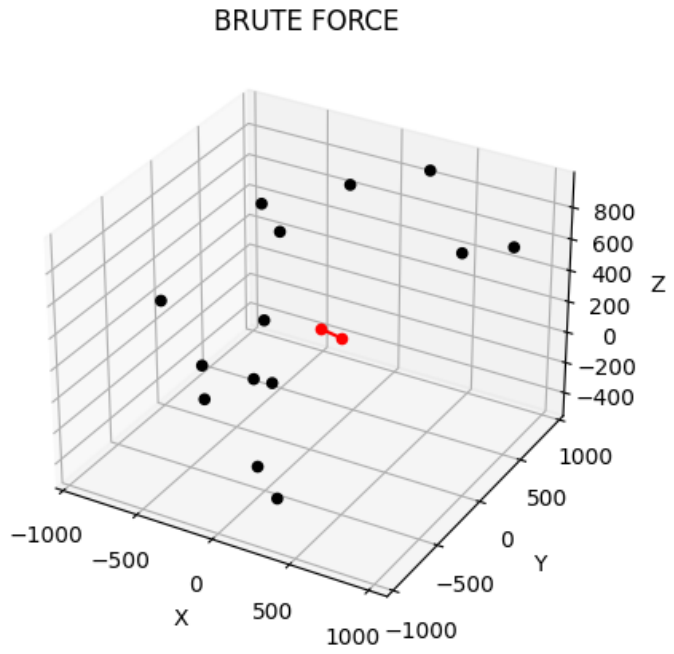


6. Test Case

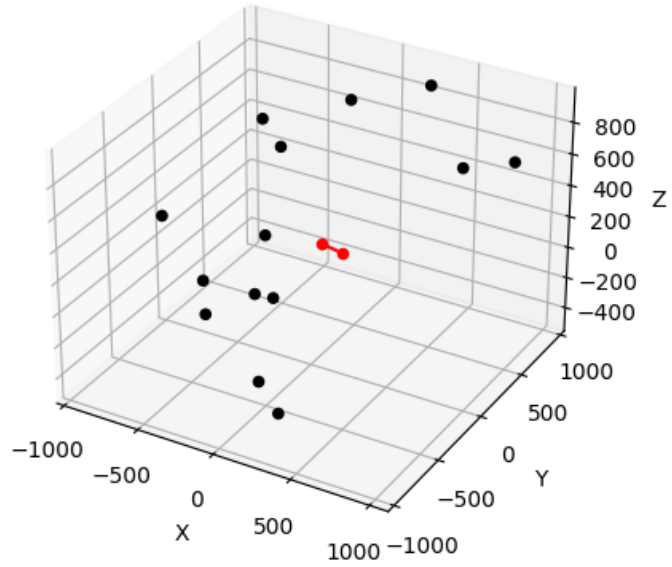
Untuk test case ini dilakukan pada Python versi 3.11.0 dan PC dengan spesifikasi di bawah ini. Angka random yang digunakan dibatasi dari -1000 sampai 1000.

a. Test Case 1
Random dan algoritma ‘both’
Banyak titik = 16
Dimensi = 3

```
============== BRUTE FORCE ===============
Two points with shortest distance:
Point 1: 238.96, -289.63, 443.73
Point 2: 421.13, -365.26, 471.44
Distance: 199.18
Number of calculation: 120
Execution time: 1.00 ms
Processor: AMD64 Family 23 Model 113 Stepping 0, AuthenticAMD

=========== DIVIDE AND CONQUER ===========
Two points with shortest distance:
Point 1: 238.96, -289.63, 443.73
Point 2: 421.13, -365.26, 471.44
Distance: 199.18
Number of calculation: 17
Execution time: 0.00 ms
Processor: AMD64 Family 23 Model 113 Stepping 0, AuthenticAMD
```



DIVIDE AND CONQUER



TestCase1.txt - Notepad

File Edit View

```
Points:
(-729.06, 879.11, 246.01)
(-8.81, 467.12, 918.42)
(-912.01, 276.30, -407.41)
(-272.92, -169.36, -84.60)
(-575.57, -223.65, -277.29)
(218.60, -929.29, -97.19)
(297.95, -855.14, -316.39)
(264.14, 916.41, 877.18)
(238.96, -289.63, 443.73)
(-845.25, -244.40, 297.67)
(521.94, 845.98, 449.96)
(-486.18, 380.85, -456.13)
(-604.49, 471.89, 646.44)
(978.15, 598.99, 722.83)
(421.13, -365.26, 471.44)
(-152.66, -244.61, 363.97)

============== BRUTE FORCE ===============
Two points with shortest distance:
Point 1: 238.96, -289.63, 443.73
Point 2: 421.13, -365.26, 471.44
Distance: 199.18
Number of calculation: 120
Execution time: 1.00 ms
Processor: AMD64 Family 23 Model 113 Stepping 0, AuthenticAMD

=========== DIVIDE AND CONQUER ===========
Two points with shortest distance:
Point 1: 238.96, -289.63, 443.73
Point 2: 421.13, -365.26, 471.44
Distance: 199.18
Number of calculation: 17
Execution time: 0.00 ms
Processor: AMD64 Family 23 Model 113 Stepping 0, AuthenticAMD
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

b. Test Case 2

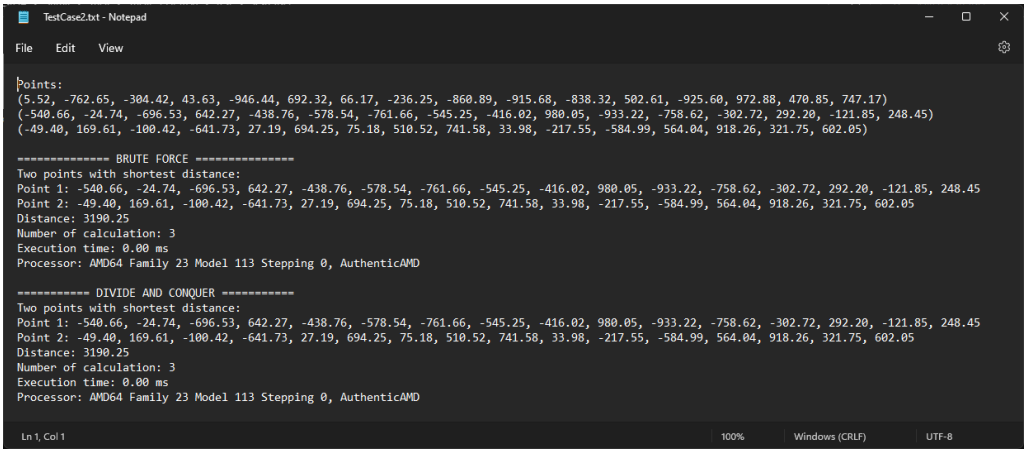
Random dan algoritma ‘both’

Banyak titik = 3

Dimensi = 16

```
============= BRUTE FORCE =============
Two points with shortest distance:
Point 1: -540.66, -24.74, -696.53, 642.27, -438.76, -578.54, -761.66, -545.25, -416.02, 980.05, -933.22, -758.62, -302.72, 292.20, -121.85, 248.45
Point 2: -49.40, 169.61, -100.42, -641.73, 27.19, 694.25, 75.18, 510.52, 741.58, 33.98, -217.55, -584.99, 564.04, 918.26, 321.75, 602.05
Distance: 3190.25
Number of calculation: 3
Execution time: 0.00 ms
Processor: AMD64 Family 23 Model 113 Stepping 0, AuthenticAMD
Not visualizable!

========== DIVIDE AND CONQUER ==========
Two points with shortest distance:
Point 1: -540.66, -24.74, -696.53, 642.27, -438.76, -578.54, -761.66, -545.25, -416.02, 980.05, -933.22, -758.62, -302.72, 292.20, -121.85, 248.45
Point 2: -49.40, 169.61, -100.42, -641.73, 27.19, 694.25, 75.18, 510.52, 741.58, 33.98, -217.55, -584.99, 564.04, 918.26, 321.75, 602.05
Distance: 3190.25
Number of calculation: 3
Execution time: 0.00 ms
Processor: AMD64 Family 23 Model 113 Stepping 0, AuthenticAMD
Not visualizable!
```

Gambar tidak dapat divisualisasikan.



c. Test Case 3

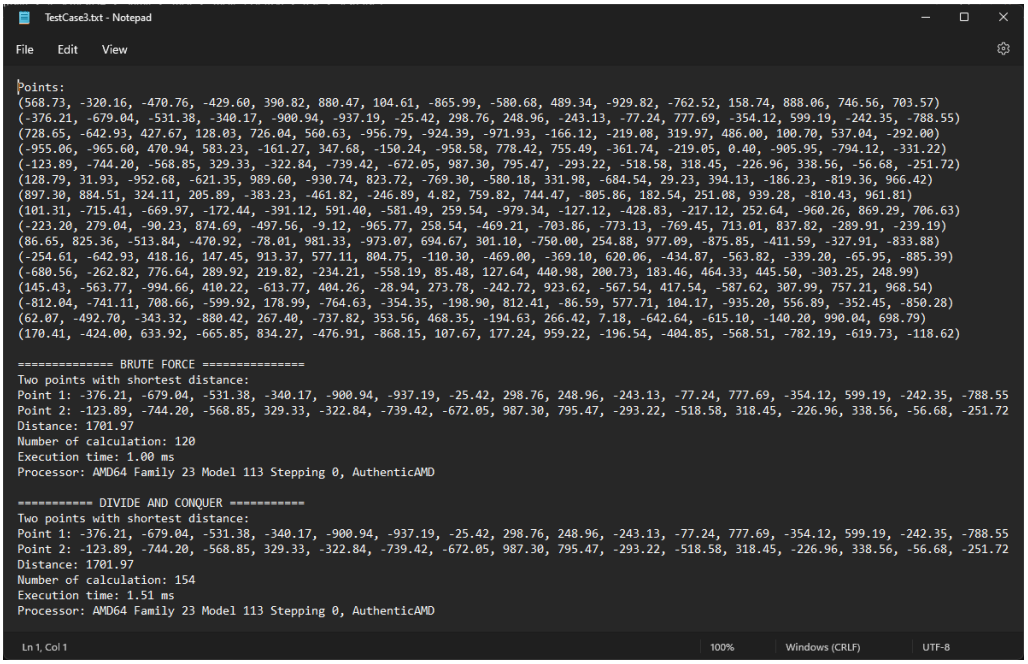
Random dan algoritma ‘both’

Banyak titik = 16

Dimensi = 16

```
============= BRUTE FORCE =============
Two points with shortest distance:
Point 1: -376.21, -679.04, -531.38, -340.17, -900.94, -937.19, -25.42, 298.76, 248.96, -243.13, -77.24, 777.69, -354.12, 599.19, -242.35, -788.55
Point 2: -123.89, -744.20, -568.85, 329.33, -322.84, -739.42, -672.05, 987.30, 795.47, -293.22, -518.58, 318.45, -226.96, 338.56, -56.68, -251.72
Distance: 1701.97
Number of calculation: 120
Execution time: 1.00 ms
Processor: AMD64 Family 23 Model 113 Stepping 0, AuthenticAMD
Not visualizable!

========== DIVIDE AND CONQUER ==========
Two points with shortest distance:
Point 1: -376.21, -679.04, -531.38, -340.17, -900.94, -937.19, -25.42, 298.76, 248.96, -243.13, -77.24, 777.69, -354.12, 599.19, -242.35, -788.55
Point 2: -123.89, -744.20, -568.85, 329.33, -322.84, -739.42, -672.05, 987.30, 795.47, -293.22, -518.58, 318.45, -226.96, 338.56, -56.68, -251.72
Distance: 1701.97
Number of calculation: 154
Execution time: 1.51 ms
Processor: AMD64 Family 23 Model 113 Stepping 0, AuthenticAMD
Not visualizable!
```

Gambar tidak dapat divisualisasikan.

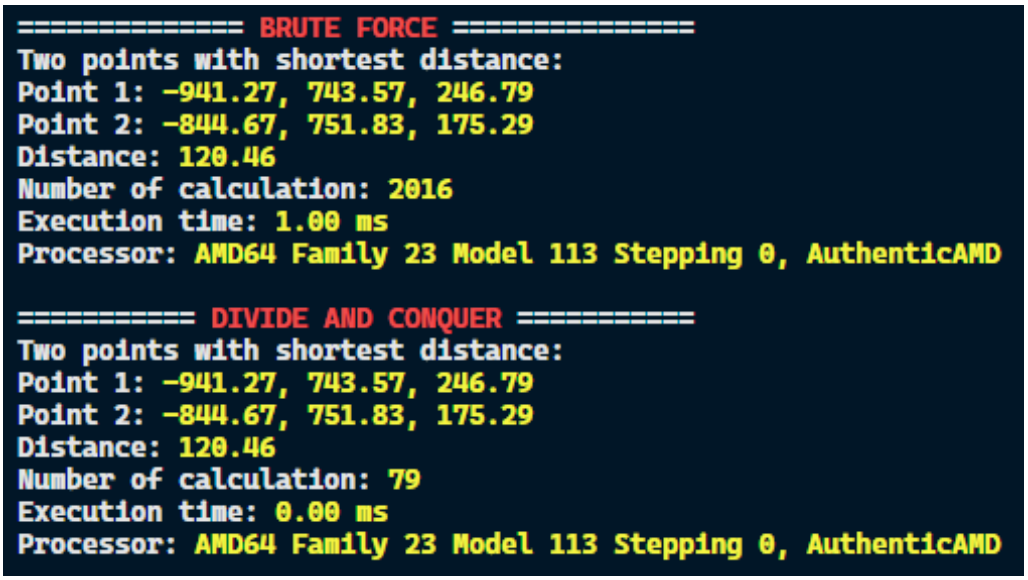


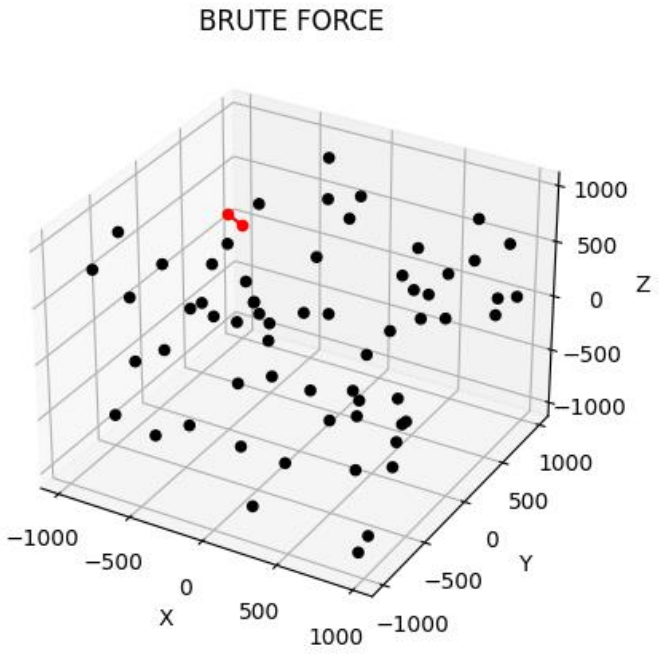
d. Test Case 4

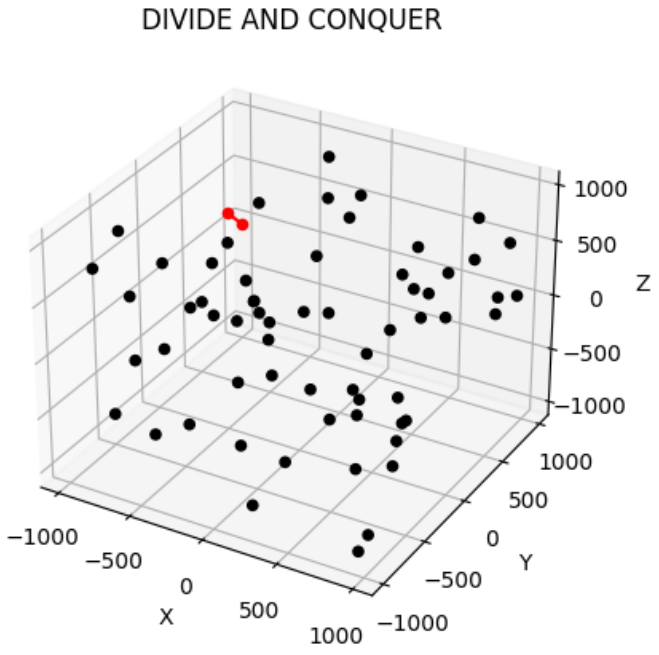
Random dan algoritma ‘both’

Banyak titik = 64

Dimensi = 3







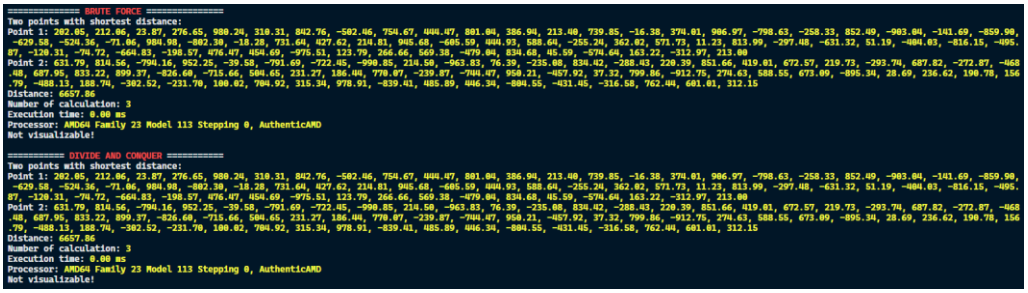
File txt tidak dapat ditampilkan karena terlalu banyak titik yang harus di-*screenshot*.

e. Test Case 5

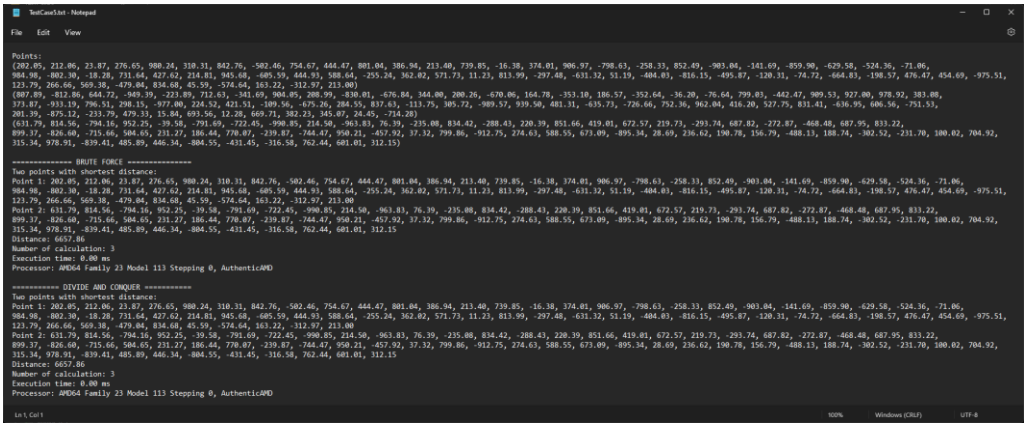
Random dan algoritma ‘both’

Banyak titik = 3

Dimensi = 64



Gambar tidak dapat divisualisasikan.

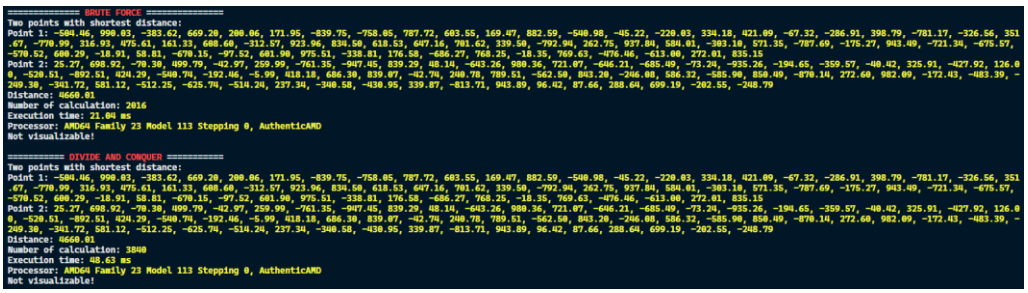


f. Test Case 6

Random dan algoritma ‘both’

Banyak titik = 64

Dimensi = 64



Gambar tidak dapat divisualisasikan dan file txt tidak dapat ditampilkan karena terlalu banyak titik yang harus di-*screenshot*.

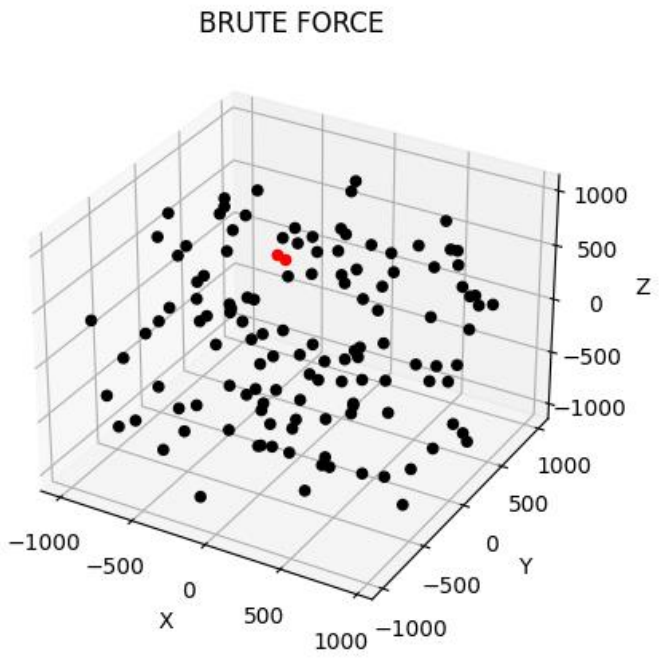
g. Test Case 7

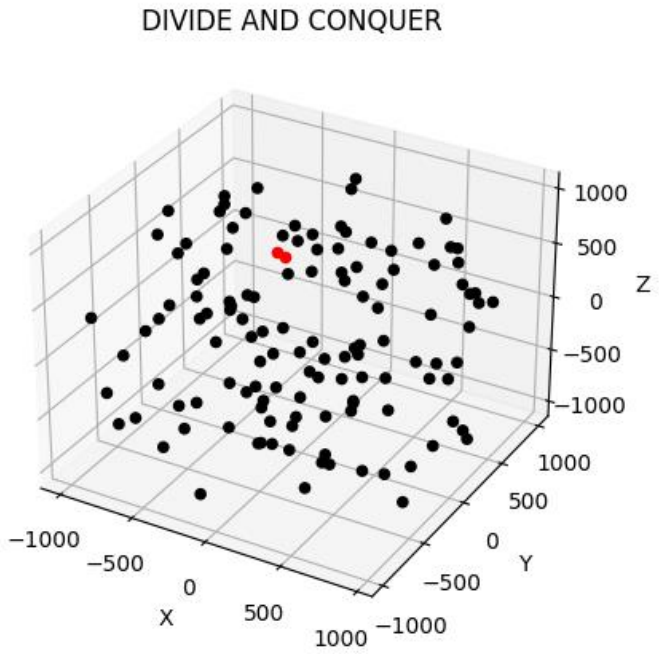
Random dan algoritma ‘both’

Banyak titik = 128

Dimensi = 3

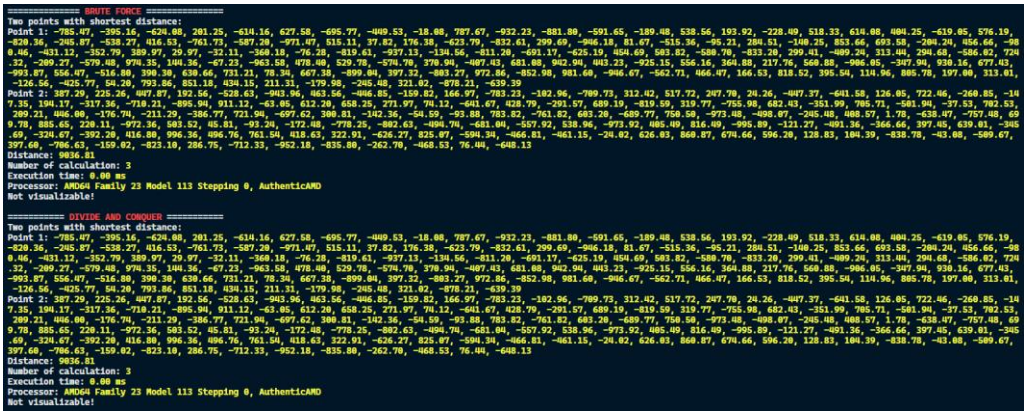
```
============== BRUTE FORCE ===============
Two points with shortest distance:
Point 1: -607.45, 752.07, 20.56
Point 2: -543.67, 752.88, 7.15
Distance: 65.18
Number of calculation: 8128
Execution time: 6.52 ms
Processor: AMD64 Family 23 Model 113 Stepping 0, AuthenticAMD

=========== DIVIDE AND CONQUER ===========
Two points with shortest distance:
Point 1: -607.45, 752.07, 20.56
Point 2: -543.67, 752.88, 7.15
Distance: 65.18
Number of calculation: 165
Execution time: 0.97 ms
Processor: AMD64 Family 23 Model 113 Stepping 0, AuthenticAMD
```





File txt tidak dapat ditampilkan karena terlalu banyak titik yang harus di-*screenshot*.

h. Test Case 8
Random dan algoritma ‘both’
Banyak titik = 3
Dimensi = 128



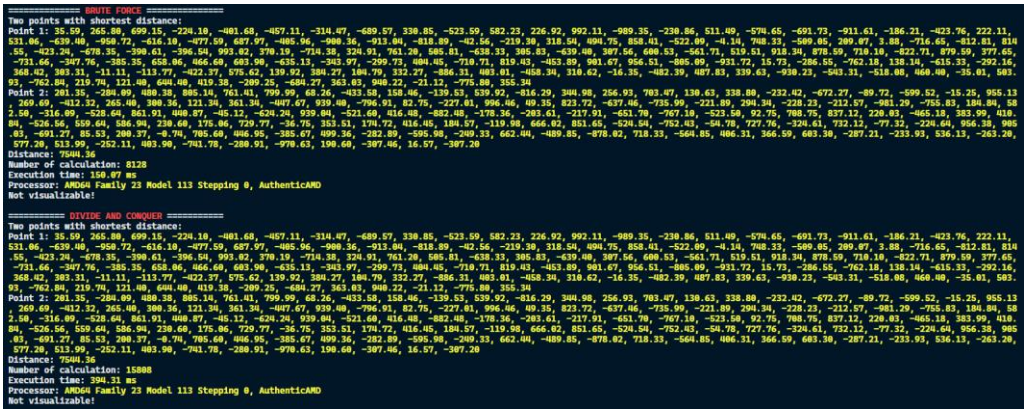
Gambar tidak dapat divisualisasikan dan file txt tidak dapat ditampilkan karena terlalu banyak titik yang harus di-*screenshot*.

i. Test Case 9

Random dan algoritma ‘both’

Banyak titik = 128

Dimensi = 128



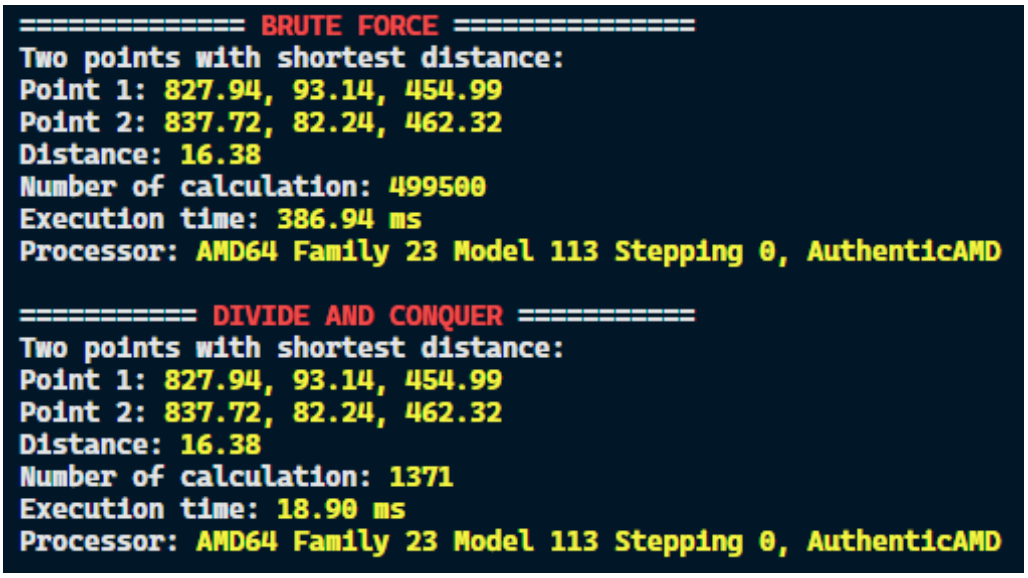
Gambar tidak dapat divisualisasikan dan file txt tidak dapat ditampilkan karena terlalu banyak titik yang harus di-*screenshot*.

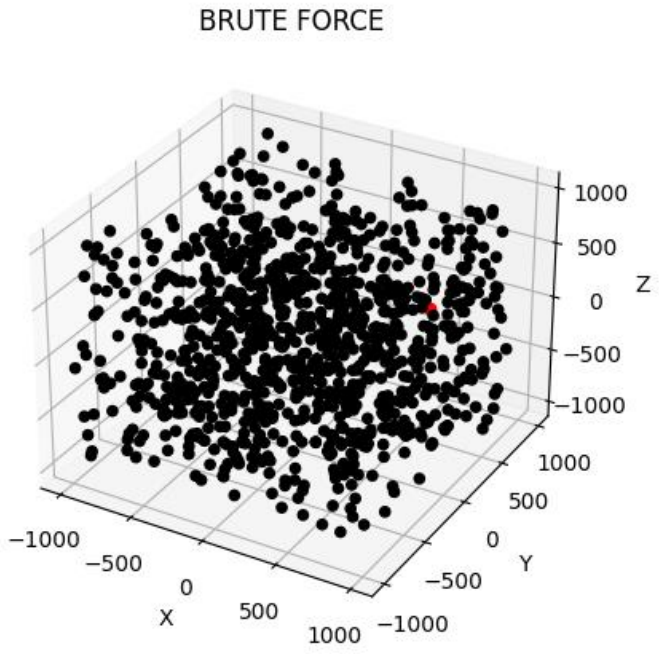
j. Test Case 10

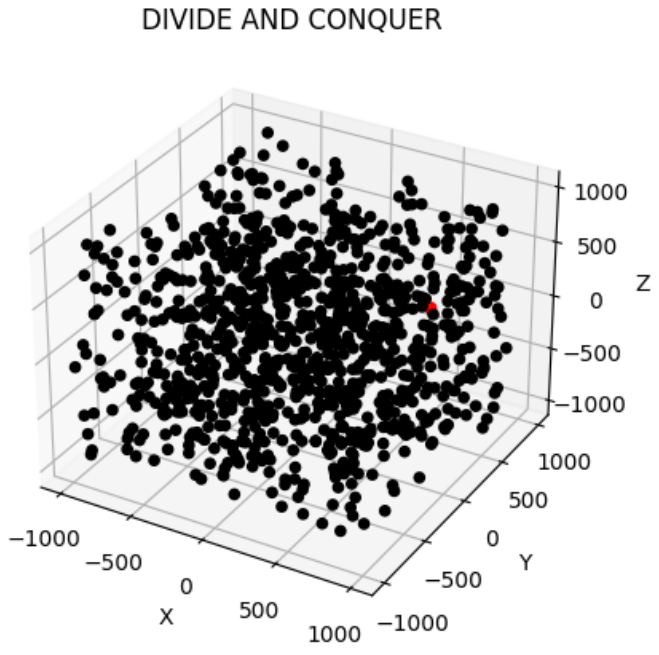
Random dan algoritma ‘both’

Banyak titik = 1000

Dimensi = 3







File txt tidak dapat ditampilkan karena terlalu banyak titik yang harus di-*screenshot*.

k. Test Case 11
Random dan algoritma ‘both’
Banyak titik = 3
Dimensi = 1000
(Dapat dilihat pada folder ‘test/TestCase11’ karena terlalu banyak titik sehingga sulit untuk ditampilkan/*screenshot*)

l. Test Case 12
Random dan algoritma ‘both’
Banyak titik = 1000
Dimensi = 1000
(Dapat dilihat pada folder ‘test/TestCase12’ karena terlalu banyak titik sehingga sulit untuk ditampilkan/*screenshot*)

Semua hasil test case dapat dilihat pada folder ‘test’ repository.

E. *Link to Repository*

https://github.com/ghaziakmalf/Tucil2_13521058

F. *Check List Table*

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