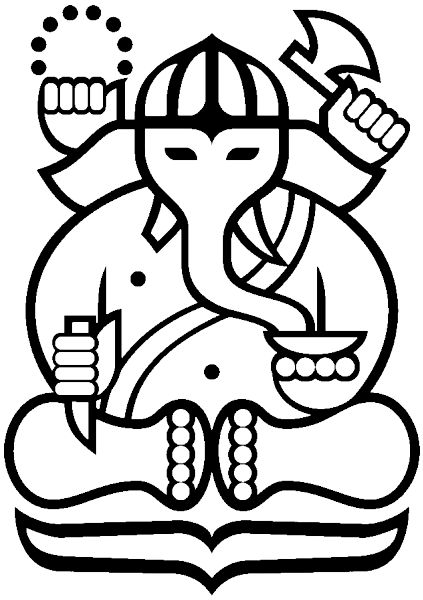
Laporan Tugas Kecil II

Mencari Pasangan Titik Terdekat 3D dengan Algoritma Divide and Conquer

Laporan dibuat untuk memenuhi salah satu tugas mata kuliah
IF2211 Strategi Algoritma



Disusun oleh:

I Putu Bakta Hari Sudewa 13521150

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2023

Daftar Isi

1 Algoritma *Divide and Conquere* **3**

2 Source Program **4**

3 Input dan Output **13**
3.1 Test Case 1 - $(N = 16, d = 3)$ 13
3.2 Test Case 2 - $(N = 16, d = 10)$ 14
3.3 Test Case 3 - $(N = 64, d = 3)$ 15
3.4 Test Case 4 - $(N = 64, d = 10)$ 16
3.5 Test Case 5 - $(N = 128, d = 3)$ 16
3.6 Test Case 6 - $(N = 128, d = 6)$ 17
3.7 Test Case 7 - $(N = 1000, d = 3)$ 18
3.8 Test Case 8 - $(N = 1000, d = 5)$ 19

4 Repositori **20**

5 Checklist **20**

6 Referensi **20**

1 Algoritma *Divide and Conquere*

Algoritma *Divide and Conquere*, terdiri dari kata *Divide* yang berarti membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula, namun berukuran lebih kecil (idealnya berukuran hampir sama) dan *Conquer (solve)* berarti menyelesaikan masing-masing upa-persoalan (secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar). Algoritma ini cukup efisien jika digunakan untuk mencari pasangan titik terdekat di antara n titik di suatu dimensi d.

Persoalan mencari pasangan titik terdekat dalam suatu dimensi d jika diselesaikan dengan algoritma *Divide and Conquer* langkah-langkahnya adalah sebagai berikut,

1. Urutkan suatu himpunan yang berisi semua titik uji berdasarkan nilai absisnya mulai dari yang terkecil ($O(n \log n)$ dengan algoritma *Quicksort*)
2. Bagi himpunan titik ke dalam dua bagian S_1 dan S_2 sama rata
3. Untuk masing-masing bagian, lakukan langkah 2 secara rekursif hingga mencapai basis, yaitu jumlah titik pada bagian tersebut ≤ 3
4. Jika jumlah titik pada suatu bagian ≤ 3, maka akan dihitung jarak terdekat antara pasangan titik pada bagian tersebut
5. Setelah diperoleh jarak terdekat pada masing-masing bagian, sebut saja δ_L dan δ_R, hitung nilai minimum antara keduanya, yaitu $min(\delta_L, \delta_R) = \delta$
6. Untuk setiap titik pada masing-masing bagian, kombinasikan setiap titik pada bagian yang satu dengan bagian yang lain, lalu cek selisih posisi tiap pasangan titik pada setiap sumbu apakah $\leq \delta$.
7. Hitung jarak *Euclidian* pasangan titik yang di setiap sumbunya memiliki nilai selisih posisi $\leq$ delta.
8. Maka akan diperoleh jarak terdekat pasangan titik, yaitu $min(\delta, \text{jarak pasangan titik}$ untuk tiap pasangan titik pada bagian 7)

Diperoleh kompleksitas algoritma *Divide and Conquer* untuk menyelesaikan persoalan ini adalah $T(n) = 2T(n/2) + O(n \log n) + O(n) = O(n \log n)$

2 Source Program

Program dibuat dengan menggunakan bahasa pemrograman Golang (implementasi algoritma) dan Python (visualisasi)

```go
package io

import (
  "bufio"
  "fmt"
  "os"
  "strconv"
  "strings"
)

// Requesting input from user through stdin and do validation
func GetInput(text string, minRange uint32, maxRange uint32) (uint32, string) {
  fmt.Print(text, ": ")

  reader := bufio.NewReader(os.Stdin)

  rawInput, err := reader.ReadString('\n')
  if err != nil {
    return 0, fmt.Sprintf("Failed to Receive %s. Please Try Again!\n", text)
  }
  trimInput := strings.TrimSpace(rawInput)

  N, err := strconv.ParseUint(trimInput, 10, 32)
  if err != nil || N < uint64(minRange) || N > uint64(maxRange) {
    return 0, fmt.Sprintf("%s Must Be a Number Between %d - %d. Please Try Again!\n", text, minRange, maxRange)
  }
  return uint32(N), ""
}

// Requesting input from user using InputPointsAmount function
func Input() (uint32, uint32) {
  for {
    N, errN := GetInput("Points Amount", 2, 100000)
    dimension, errD := GetInput("Dimension", 1, 10)
    if errN != "" || errD != "" {
      fmt.Println()
      fmt.Print(errN)
      fmt.Print(errD)
      fmt.Println()
    } else {
      return N, dimension
    }
  }
}
```

Source Code 1: input.go

Pada Source Code 1 dilakukan implementasi seluruh fungsi yang bertugas untuk melakukan proses input data dan validasinya.

```go
package io

import (
  "encoding/csv"
```

```
  "fmt"
  "os"

  "github.com/sozyGithub/project/tucil_2/src/point"
)

// Writing array of points to CSV file for visualization
func WritePointsToCSV(points []point.Point) {
  dataset := make([][]string, len(points)+1)
  var header = []string{
    "X", "Y", "Z",
  }
  dataset[0] = append(dataset[0], header...)
  for i := 1; i <= len(points); i++ {
    for j := 1; j <= int(points[i-1].GetDimension()); j++ {
      dataset[i] = append(dataset[i], fmt.Sprintf("%v", points[i-1].
   GetCoor(j)))
    }
  }

  csvFile, err := os.Create("../src/plot/points.csv")
  if err != nil {
    fmt.Println("\n!!Failed to export points.!!")
  }

  csvWriter := csv.NewWriter(csvFile)
  for _, row := range dataset {
    _ = csvWriter.Write(row)
  }
  csvWriter.Flush()
  csvFile.Close()
}

// Writing array of closest pair of points to CSV file for visualization
func WritePPointsToCSV(sdArray [][]point.Point) {
  dataset := make([][]string, len(sdArray)*2+1)
  var header = []string{
    "X", "Y", "Z",
  }
  dataset[0] = append(dataset[0], header...)
  for i := 1; i <= len(sdArray); i++ {
    for j := 1; j <= int(sdArray[i-1][0].GetDimension()); j++ {
      dataset[i] = append(dataset[i], fmt.Sprintf("%v", sdArray[i-1][0].
   GetCoor(j)))
    }
    for j := 1; j <= int(sdArray[i-1][1].GetDimension()); j++ {
      dataset[i+1] = append(dataset[i+1], fmt.Sprintf("%v", sdArray[i
   -1][1].GetCoor(j)))
    }
  }

  csvFile, err := os.Create("../src/plot/ppoints.csv")
  if err != nil {
    fmt.Println("\n!!Failed to export ppoints.!!")
  }

  csvWriter := csv.NewWriter(csvFile)
  for _, row := range dataset {
    _ = csvWriter.Write(row)
  }
  csvWriter.Flush()
```

```
  csvFile.Close()
}
```

Source Code 2: output.go

Proses penulisan seluruh *point* hasil pembangkitan secara acak dan *point* yang memiliki jarak terdekat dituliskan dengan menggunakan fungsi yang terdapat pada Source Code 2.

```
package io

import "fmt"

// Initial screen display
func InitScreen() {
  welcomeString := "Closest Pair N-D Points"
  fmt.Printf("%s %*s\n", "+++", len(welcomeString)+4, "+++")
  fmt.Printf("+++ %s +++\n", welcomeString)
  fmt.Printf("%s %*s\n\n", "+++", len(welcomeString)+4, "+++")
}
```

Source Code 3: screen.go

Welcome screen awal program merupakan tanggung jawab dari fungsi yang terdapat pada Source Code 3.

```
package point

import (
  "fmt"
  "math"
  "math/rand"
  "time"
)

const float64EqualityTracehold = 1e-12

// Definition of point object
type Point struct {
  dimension  uint32
  coordinate []float64
}

// Creating a new point instance
func CreatePoint(coordinate []float64, dimension uint32) Point {
  return Point{
    coordinate: coordinate,
    dimension:  dimension,
  }
}

// Getting coordinates of a point in string format
func (p *Point) GetCoors() string {
  var sCoor string = "("
  for i := 0; i < int(p.dimension); i++ {
    if i != 0 {
      sCoor += ","
    }
    sCoor += fmt.Sprintf("%0.3f", p.coordinate[i])
  }
  sCoor += ")"
  return sCoor
}

```

```go
// Getter for coordinate
func (p *Point) GetCoor(index int) float64 {
  return p.coordinate[index-1]
}

// Getter for dimension
func (p *Point) GetDimension() uint32 {
  return p.dimension
}

// Generating randoms point
func GeneratePoints(N uint32, dimension uint32) []Point {
  var points []Point
  rand.Seed(time.Now().UnixNano())

  for i := 0; i < int(N); i++ {
    coordinate := make([]float64, dimension)
    for j := 0; j < int(dimension); j++ {
      sign := rand.Intn(2)
      coordinate[j] = float64(rand.Float64() * 1000 * math.Pow(-1, float64
   (sign)))
    }
    points = append(points, CreatePoint(coordinate, dimension))
  }

  pointsX := make([]Point, len(points))
  copy(pointsX, points)

  sortedPoints := QuickSortPoints(pointsX)

  return sortedPoints
}

// Sorting []Point using Quick Sort Algorithm
func QuickSortPoints(points []Point) []Point {
  if len(points) < 2 {
    return points
  }

  left, right := 0, len(points)-1
  pivot := rand.Int() % len(points)

  points[pivot], points[right] = points[right], points[pivot]
  for i := range points {
    if points[i].GetCoor(1) < points[right].GetCoor(1) {
      points[left], points[i] = points[i], points[left]
      left++
    }
  }

  points[left], points[right] = points[right], points[left]
  QuickSortPoints(points[:left])
  QuickSortPoints(points[left+1:])

  return points
}

// Calculate the Euclidian Distance between point p1 and p2
func CalculateDistance(p1, p2 Point) float64 {
  var sum float64 = 0
  for i := 1; i <= int(p1.GetDimension()); i++ {
```

```
    sum += math.Pow(p2.GetCoor(i)-p1.GetCoor(i), 2)
  }
  return math.Sqrt(sum)
}

// Check if two floating point number is equal
func IsEqual(a, b float64) bool {
  return math.Abs(a-b) <= float64EqualityTracehold
}

// Print formated [][]Point from slice of shortest distance array
func PrintFormattedPoint(sdArray [][]Point) {
  for i := 0; i < len(sdArray); i++ {
    fmt.Printf("%*s %s %s\n", len(sdArray[i][0].GetCoors())+22, sdArray[i
    ][0].GetCoors(), "<->", sdArray[i][1].GetCoors())
  }
}
```

Source Code 4: point.go

Pada Source Code 4 dilakukan implementasi seluruh atribut dan *method* yang berhubungan dengan *instance point*.

```
package algorithm

import (
  "math"

  "github.com/sozyGithub/project/tucil_2/src/point"
)

// Brute Force Algorithm implementation
func DoBruteForce(points []point.Point, totalOpt *int) (float64, [][]point
    .Point) {
  var shortestDistance float64 = math.MaxFloat64
  var sdArray [][]point.Point
  for i := 0; i < len(points); i++ {
    for j := i + 1; j < len(points); j++ {
      *totalOpt++
      distance := point.CalculateDistance(points[i], points[j])
      if distance < shortestDistance {
        shortestDistance = distance
        sdArray = nil
        sdArray = append(sdArray, []point.Point{points[i], points[j]})
      } else if point.IsEqual(distance, shortestDistance) {
        sdArray = append(sdArray, []point.Point{points[i], points[j]})
      }
    }
  }

  return shortestDistance, sdArray
}
```

Source Code 5: bruteforce.go

Pada Source Code 5 dilakukan implementasi algoritma *Brute Force* yang nantinya digunakan sebagai pembanding algoritma *Divide and Conquer*

```
package algorithm

import (
  "math"

```

```
  "github.com/sozyGithub/project/tucil_2/src/point"
)

// Divide and Conquer Algorithm implementation
func DoDNC(pointsX []point.Point, totalOpt *int) (float64, [][]point.Point
    ) {
  n := len(pointsX)
  dimension := pointsX[0].GetDimension()

  var sPoint1 point.Point
  var sPoint2 point.Point
  var shortestDistance float64
  var sdArray = make([][]point.Point, 0)

  // Base cases
  if n == 2 {
    *totalOpt++
    var pPoint = []point.Point{pointsX[0], pointsX[1]}
    sdArray = append(sdArray, pPoint)
    return point.CalculateDistance(pointsX[0], pointsX[1]), sdArray
  }
  if n == 3 {
    var pPoint = []point.Point{sPoint1, sPoint2}
    sdArray = append(sdArray, pPoint)
    shortestDistance, sdArray = DoBruteForce(pointsX, totalOpt)
    return shortestDistance, sdArray
  }

  // Divide
  midPoint := pointsX[n/2]
  deltaLeft, sdArrayL := DoDNC(pointsX[0:n/2], totalOpt)
  deltaRight, sdArrayR := DoDNC(pointsX[n/2:n], totalOpt)
  if deltaLeft < deltaRight {
    shortestDistance = deltaLeft
    temp := make([][]point.Point, len(sdArrayL))
    copy(temp, sdArrayL)
    sdArray = append(sdArray, temp...)
  } else if deltaLeft > deltaRight {
    shortestDistance = deltaRight
    temp := make([][]point.Point, len(sdArrayR))
    copy(temp, sdArrayR)
    sdArray = append(sdArray, temp...)
  } else {
    sdArrayL = append(sdArrayL, sdArrayR...)
    sdArray = append(sdArray, sdArrayL...)
  }

  leftPoints := pointsX[0 : n/2]
  rightPoints := pointsX[n/2 : n]

  indxL := 0
  indxR := 0
  inLeftPoints := make([]point.Point, n/2+1)
  inRightPoints := make([]point.Point, n/2+1)
  for _, pointL := range leftPoints {
    if pointL.GetCoor(1) >= midPoint.GetCoor(1)-shortestDistance {
      inLeftPoints[indxL] = pointL
      indxL++
    }
  }
  for _, pointR := range rightPoints {
```

```
    if pointR.GetCoor(1) <= midPoint.GetCoor(1)+shortestDistance {
      inRightPoints[indxR] = pointR
      indxR++
    }
  }

  // Conquer
  for i := 0; i < indxL; i++ {
    for j := 0; j < indxR; j++ {
      skip := false
      for k := 1; k <= int(dimension); k++ {
        if math.Abs(inLeftPoints[i].GetCoor(k)-inRightPoints[j].GetCoor(k)
   ) > shortestDistance {
          skip = true
          break
        }
      }
      if skip {
        continue
      }
      *totalOpt++
      pointDist := point.CalculateDistance(inLeftPoints[i], inRightPoints[
   j])
      if pointDist < shortestDistance {
        shortestDistance = pointDist
        sdArray = nil
        sdArray = append(sdArray, []point.Point{inLeftPoints[i],
   inRightPoints[j]})
      } else if point.IsEqual(pointDist, shortestDistance) {
        sdArray = append(sdArray, []point.Point{inLeftPoints[i],
   inRightPoints[j]})
      }
    }
  }

  return shortestDistance, sdArray
}
```

Source Code 6: dnc.go

Seperti namanya, pada Source Code 6 terdapat fungsi yang menyelesaikan persoalan dengan algoritma utama program, yaitu *Divide and Conquer*.

```
package main

import (
  "fmt"
  "time"

  "github.com/klauspost/cpuid/v2"
  "github.com/matishsiao/goInfo"
  "github.com/pbnjay/memory"
  "github.com/sozyGithub/project/tucil_2/src/algorithm"
  "github.com/sozyGithub/project/tucil_2/src/io"
  "github.com/sozyGithub/project/tucil_2/src/point"
)

func main() {
  io.InitScreen()

  N, dimension := io.Input()
  pointsX := point.GeneratePoints(N, dimension)
```

```
  gi, _ := goInfo.GetInfo()

  fmt.Print("\n---\n\n")

  // Brute Force
  startTimeBF := time.Now()
  totalOptBF := 0
  shortestDistanceBF, sdArrayBF := algorithm.DoBruteForce(pointsX, &
   totalOptBF)
  elapsedTimeBF := time.Since(startTimeBF).Seconds()

  // Divide and Conquer
  startTimeDNC := time.Now()
  totalOptDNC := 0
  shortestDistanceDNC, sdArrayDNC := algorithm.DoDNC(pointsX, &totalOptDNC
   )
  elapsedTimeDNC := time.Since(startTimeDNC).Seconds()

  // Brute Force Data
  fmt.Println("==  BRUTE FORCE ==")
  fmt.Printf("%-20s: %f\n", "Shortest Distance", shortestDistanceBF)
  fmt.Printf("%-20s: \n", "Points")
  point.PrintFormattedPoint(sdArrayBF)
  fmt.Printf("%-20s: %d\n", "Total Operations", totalOptBF)
  fmt.Printf("%-20s: %v s\n", "Total Time", elapsedTimeBF)
  fmt.Print("\n---\n\n")

  // Divide and Conquer Data
  fmt.Println("==  DIVIDE AND CONQUER ==")
  fmt.Printf("%-20s: %f\n", "Shortest Distance", shortestDistanceDNC)
  fmt.Printf("%-20s: \n", "Points")
  point.PrintFormattedPoint(sdArrayDNC)
  fmt.Printf("%-20s: %d\n", "Total Operations", totalOptDNC)
  fmt.Printf("%-20s: %v s\n", "Total Time", elapsedTimeDNC)
  fmt.Print("\n---\n\n")

  // Computer Specifications
  fmt.Println("== COMPUTER SPECIFICATIONS ==")
  fmt.Printf("%-20s: %v\n", "Operating System", gi.OS)
  fmt.Printf("%-20s: %v GB\n", "Memory", float64(memory.TotalMemory())/
   float64(1000000000))
  fmt.Printf("%-20s: %s\n", "CPU Name", cpuid.CPU.BrandName)
  fmt.Printf("%-20s: %d\n", "Physical Cores", cpuid.CPU.PhysicalCores)
  fmt.Printf("%-20s: %d\n", "Logical Cores", cpuid.CPU.LogicalCores)
  if dimension == 3 {
    fmt.Println("\n---")
    fmt.Println("Visualisasi titik yang dihasilkan dapat dilihat dengan
   menjalankan perintah 'make run-visualizer' pada terminal")
    fmt.Println("---")
    io.WritePointsToCSV(pointsX)
    io.WritePPointsToCSV(sdArrayDNC)
  }
}
```

Source Code 7: main.go

Source Code 7 merupakan program utama tempat dipanggilnya fungsi-fungsi yang memiliki perannya masing-masing. Pada bagian ini pula diimplementasikan proses mencetak data ke layar.

```
import matplotlib.pyplot as plt
import pandas as pd
```

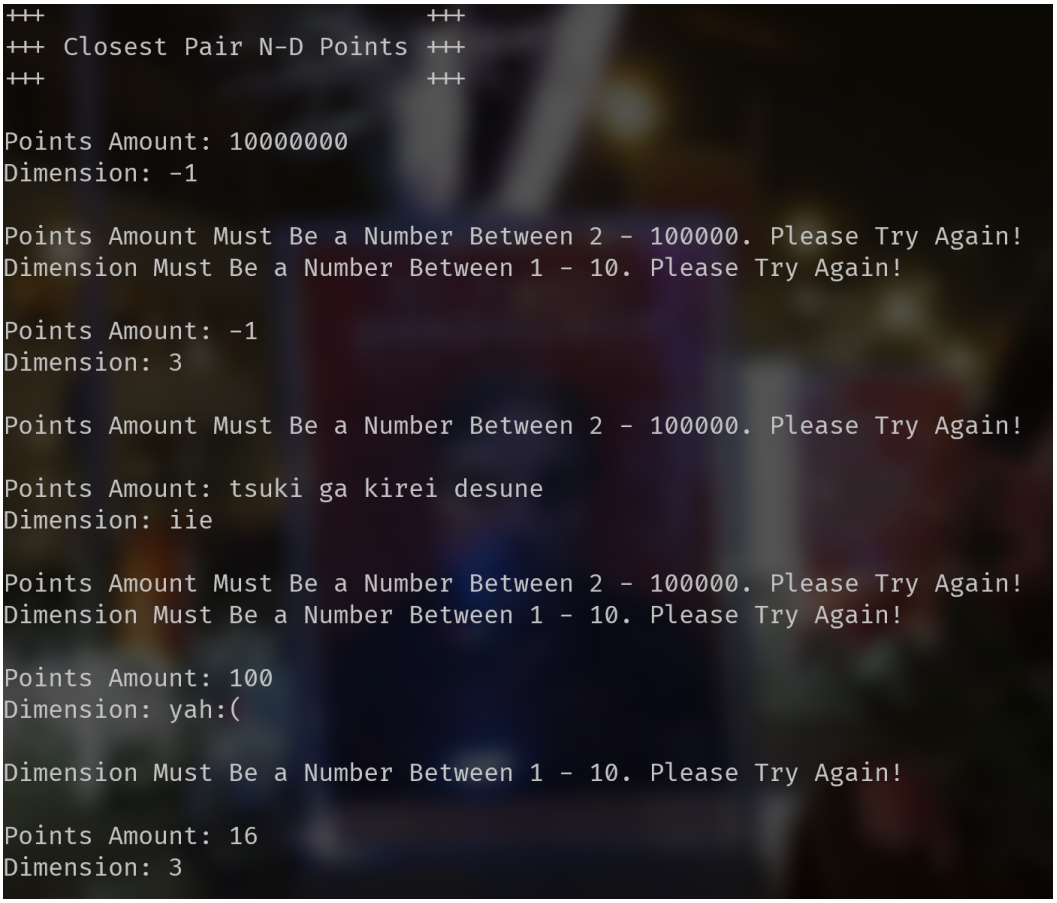
```
3
4  # Read data from CSV
5  df = pd.read_csv('points.csv')
6  df2 = pd.read_csv('ppoints.csv')
7
8  # Initializing the frame
9  plt.figure(figsize=(15, 13))
10 ax = plt.axes(projection='3d')
11
12 # Drawing point to the frame
13 for i in range(len(df.index)):
14     inside = False
15     for j in range(len(df2.index)):
16         if (str(df['X'][i]) == str(df2['X'][j]) and str(df['Y'][i]) == str
       (df2['Y'][j]) and str(df['Z'][i]) == str(df2['Z'][j])):
17             inside = True
18             break
19     if inside:
20         ax.scatter(df['X'][i], df['Y'][i], df['Z'][i], color='red')
21     else:
22         ax.scatter(df['X'][i], df['Y'][i], df['Z'][i], color='blue')
23 for i in range(1, len(df2.index), 2):
24     ax.plot(df2['X'][i-1:i+1], df2['Y'][i-1:i+1],
25             df2['Z'][i-1:i+1], color='red')
26
27 # Saving plotting result to file
28 plt.savefig("../../output/result.png")
29
30 # Showing the visualizer
31 plt.show()
```

Source Code 8: plot.py

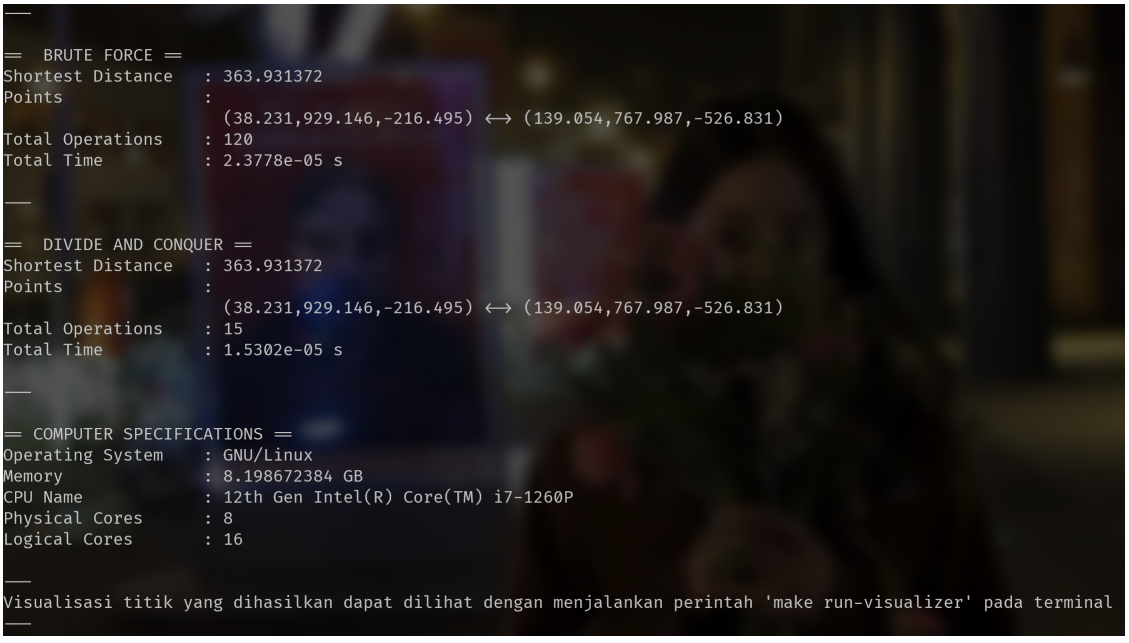
Source Code 8 bertanggung jawab untuk melakukan visualisasi dengan menggunakan kekuatan python yang sangat luar biasa dalam aspek visualisasi data.

3 Input dan Output

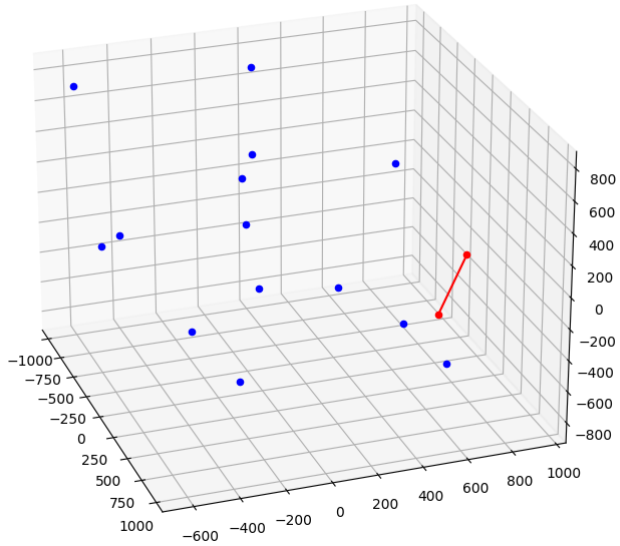
3.1 Test Case 1 - $(N = 16, d = 3)$



Gambar 1: Test Case 1 - Input dan validasi input

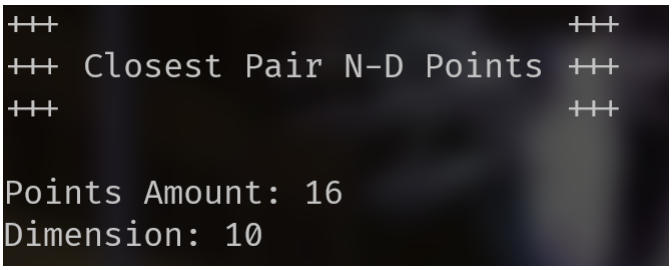


Gambar 2: Test Case 1 - Output



Gambar 3: Test Case 1 - Visualisasi

3.2 Test Case 2 - $(N = 16, d = 10)$



Gambar 4: Test Case 2 - Input

Gambar 5: Test Case 2 - Output

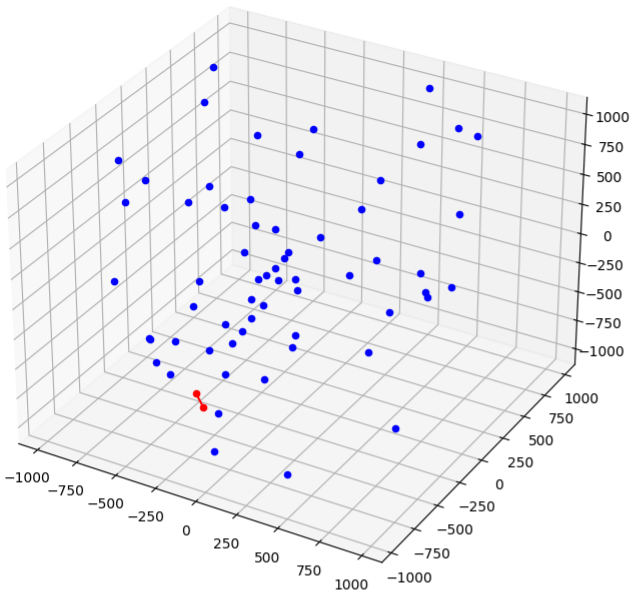
3.3 Test Case 3 - $(N = 64, d = 3)$

```
+++                                 +++
+++ Closest Pair N-D Points         +++
+++                                 +++

Points Amount: 64
Dimension: 3
```

Gambar 6: Test Case 3 - Input

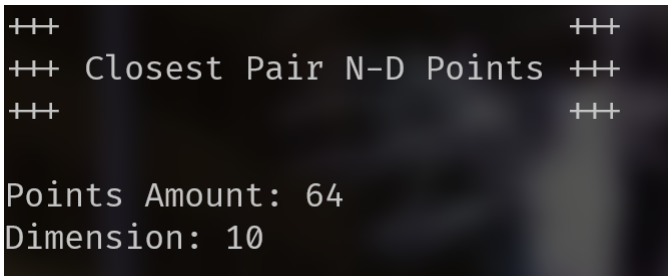
```
──

═  BRUTE FORCE ═
Shortest Distance   : 107.550721
Points              :
                      (-354.833,-513.646,-810.969) ⟷ (-306.204,-518.926,-906.753)
Total Operations    : 2016
Total Time          : 0.000339873 s

──

═  DIVIDE AND CONQUER ═
Shortest Distance   : 107.550721
Points              :
                      (-354.833,-513.646,-810.969) ⟷ (-306.204,-518.926,-906.753)
Total Operations    : 66
Total Time          : 4.2146e-05 s

──

═ COMPUTER SPECIFICATIONS ═
Operating System    : GNU/Linux
Memory              : 8.198672384 GB
CPU Name            : 12th Gen Intel(R) Core(TM) i7-1260P
Physical Cores      : 8
Logical Cores       : 16

──
Visualisasi titik yang dihasilkan dapat dilihat dengan menjalankan perintah 'make run-visualizer' pada terminal
──
```

Gambar 7: Test Case 3 - Output

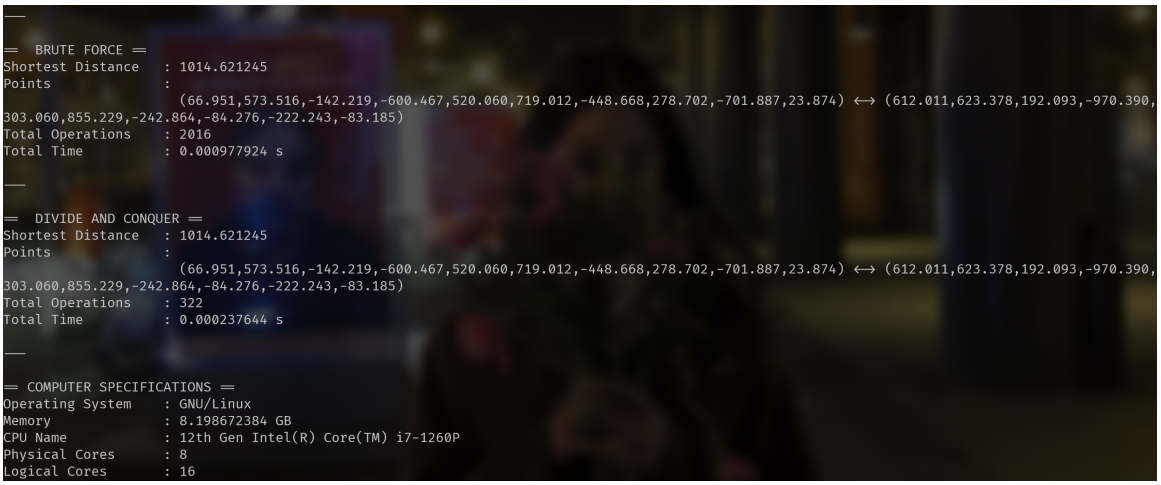


Gambar 8: Test Case 3 - Visualisasi

3.4 Test Case 4 - $(N = 64, d = 10)$

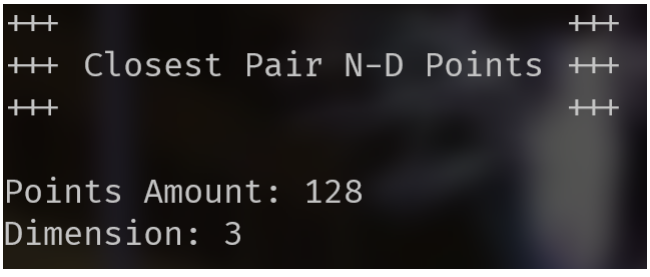


Gambar 9: Test Case 4 - Input



Gambar 10: Test Case 4 - Output

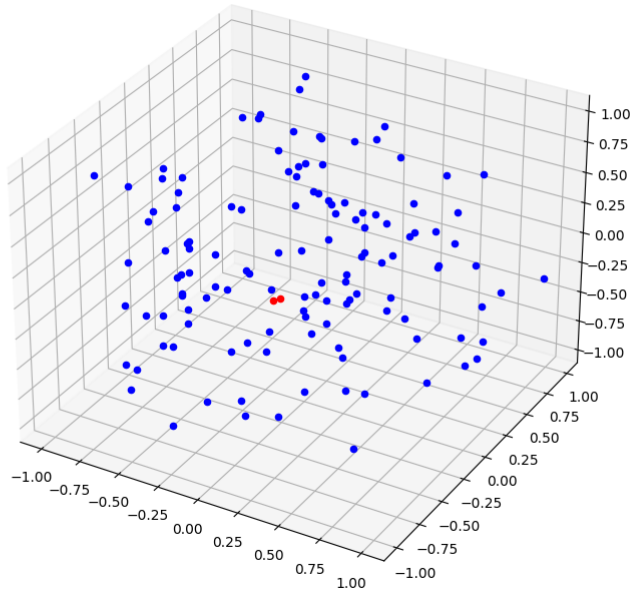
3.5 Test Case 5 - $(N = 128, d = 3)$



Gambar 11: Test Case 5 - Input

```
──

═  BRUTE FORCE ═
Shortest Distance   : 64881.329106
Points              :
                      (-416625.936,369230.151,-662997.771) ⟷ (-407815.587,429995.886,-683961.789)
Total Operations    : 8128
Total Time          : 0.001214091 s

──

═  DIVIDE AND CONQUER ═
Shortest Distance   : 64881.329106
Points              :
                      (-416625.936,369230.151,-662997.771) ⟷ (-407815.587,429995.886,-683961.789)
Total Operations    : 132
Total Time          : 0.000103886 s

──

═ COMPUTER SPECIFICATIONS ═
Operating System    : GNU/Linux
Memory              : 8.198672384 GB
CPU Name            : 12th Gen Intel(R) Core(TM) i7-1260P
Physical Cores      : 8
Logical Cores       : 16

──
Visualisasi titik yang dihasilkan dapat dilihat dengan menjalankan perintah 'make run-visualizer' pada terminal
──
```

Gambar 12: Test Case 5 - Output



Gambar 13: Test Case 5 - Visualisasi

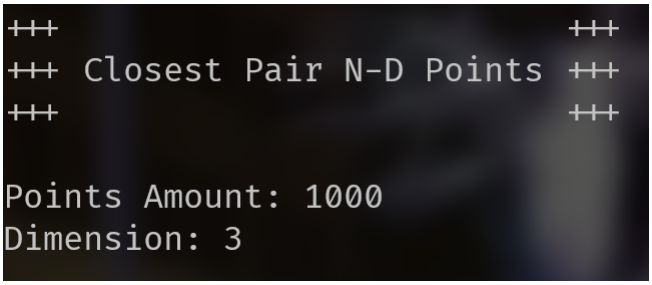
3.6 Test Case 6 - $(N = 128, d = 6)$

```
+++                                 +++
+++ Closest Pair N-D Points         +++
+++                                 +++

Points Amount: 128
Dimension: 6
```

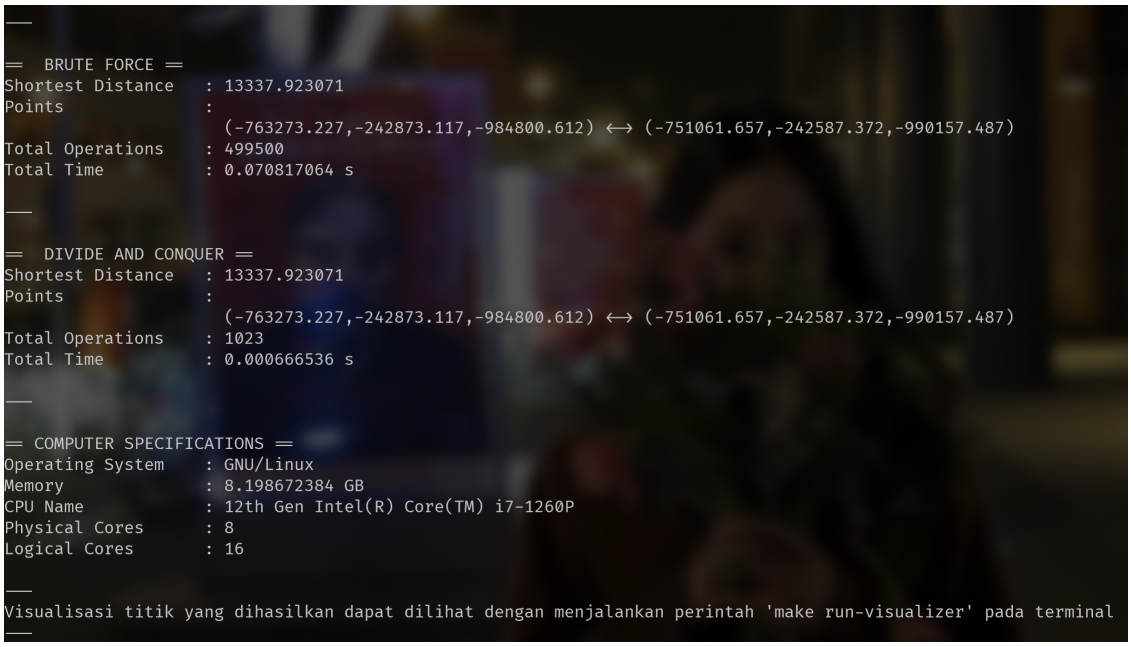
Gambar 14: Test Case 6 - Input

Gambar 15: Test Case 6 - Output

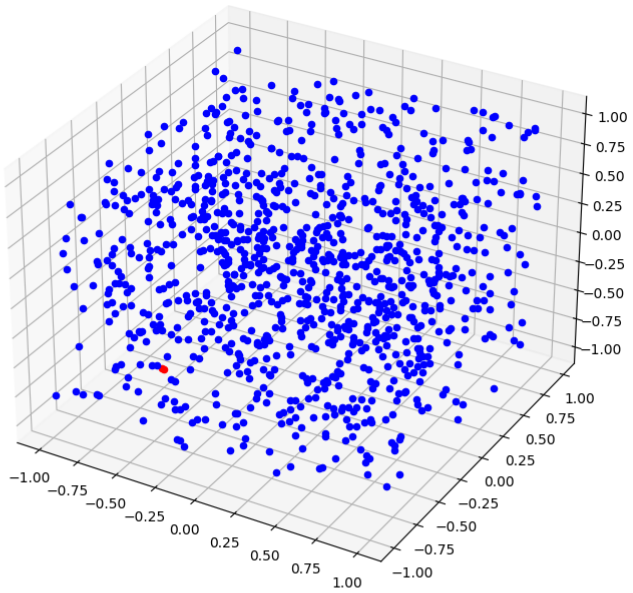
3.7 Test Case 7 - $(N = 1000, d = 3)$



Gambar 16: Test Case 7 - Input

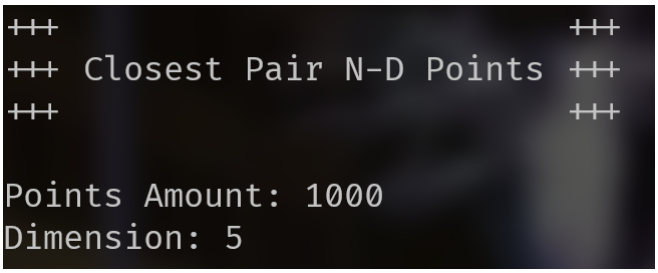


Gambar 17: Test Case 7 - Output

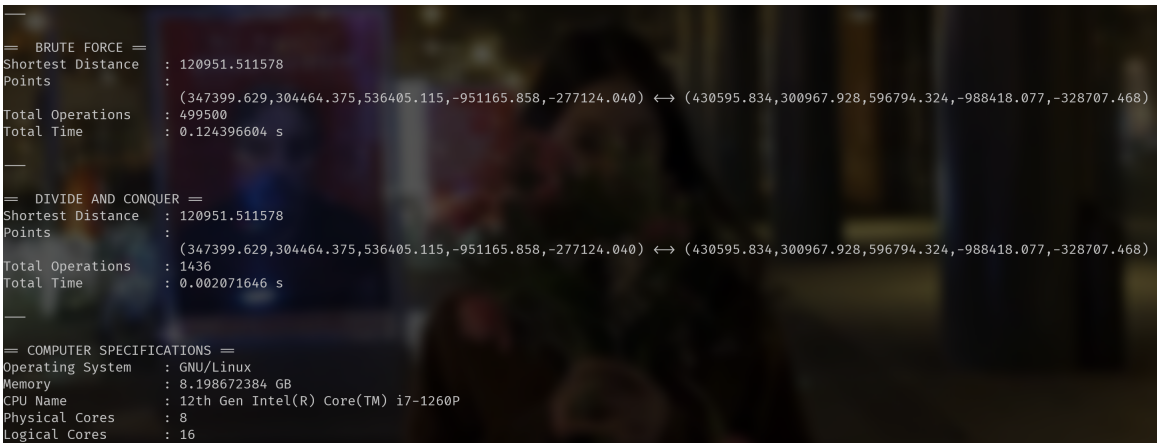


Gambar 18: Test Case 7 - Visualisai

3.8 Test Case 8 - $(N = 1000, d = 5)$



Gambar 19: Test Case 8 - Input



Gambar 20: Test Case 8 - Output

4 Repositori

https://github.com/sozyGithub/Tucil2_13521150

5 Checklist

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

6 Referensi

- R. Munir, (2023). *Algoritma Divide and Conquer (bagian 1)*[Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian1.pdf
- R. Munir, (2023). *Algoritma Divide and Conquer (bagian 2)*[Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian2.pdf