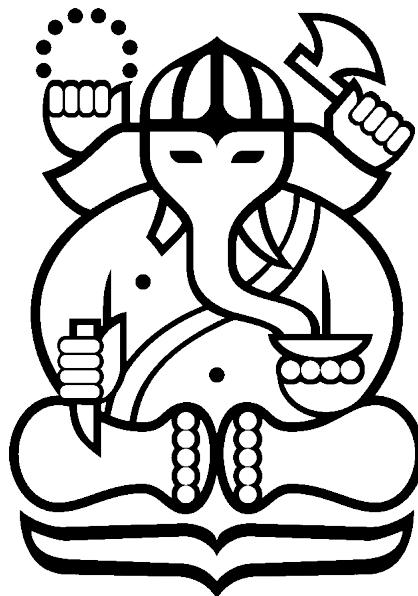


LAPORAN TUGAS KECIL II
MENCARI PASANGAN TITIK TERDEKAT DENGAN
ALGORITMA DIVIDE AND CONQUER

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 <i>Divide and Conquere</i>	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)

```
1 package io
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strconv"
8     "strings"
9 )
10
11 // Requesting input from user through stdin and do validation
12 func GetInput(text string, minRange uint32, maxRange uint32) (uint32,
13     string) {
14     fmt.Print(text, ": ")
15
16     reader := bufio.NewReader(os.Stdin)
17
18     rawInput, err := reader.ReadString('\n')
19     if err != nil {
20         return 0, fmt.Sprintf("Failed to Receive %s. Please Try Again!\n",
21             text)
22     }
23     trimInput := strings.TrimSpace(rawInput)
24
25     N, err := strconv.ParseUint(trimInput, 10, 32)
26     if err != nil || N < uint64(minRange) || N > uint64(maxRange) {
27         return 0, fmt.Sprintf("%s Must Be a Number Between %d - %d. Please Try
28             Again!\n", text, minRange, maxRange)
29     }
30     return uint32(N), ""
31 }
32
33 // Requesting input from user using InputPointsAmount function
34 func Input() (uint32, uint32) {
35     for {
36         N, errN := GetInput("Points Amount", 2, 100000)
37         dimension, errD := GetInput("Dimension", 1, 10)
38         if errN != "" || errD != "" {
39             fmt.Println()
40             fmt.Print(errN)
41             fmt.Print(errD)
42             fmt.Println()
43         } else {
44             return N, dimension
45         }
46     }
47 }
```

Source Code 1: input.go

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

```
1 package io
2
3 import (
4     "encoding/csv"
```

```

5   "fmt"
6   "os"
7
8   "github.com/sozyGithub/project/tucil_2/src/point"
9 )
10
11 // Writing array of points to CSV file for visualization
12 func WritePointsToCSV(points []point.Point) {
13     dataset := make([][]string, len(points)+1)
14     var header = []string{
15         "X", "Y", "Z",
16     }
17     dataset[0] = append(dataset[0], header...)
18     for i := 1; i <= len(points); i++ {
19         for j := 1; j <= int(points[i-1].GetDimension()); j++ {
20             dataset[i] = append(dataset[i], fmt.Sprintf("%v", points[i-1].
21                 GetCoor(j)))
22         }
23     }
24
25     csvFile, err := os.Create("../src/plot/points.csv")
26     if err != nil {
27         fmt.Println("\n!!Failed to export points.!!")
28     }
29
30     csvWriter := csv.NewWriter(csvFile)
31     for _, row := range dataset {
32         _ = csvWriter.Write(row)
33     }
34     csvWriter.Flush()
35     csvFile.Close()
36 }
37
38 // Writing array of closest pair of points to CSV file for visualization
39 func WritePPointsToCSV(sdArray [][]point.Point) {
40     dataset := make([][]string, len(sdArray)*2+1)
41     var header = []string{
42         "X", "Y", "Z",
43     }
44     dataset[0] = append(dataset[0], header...)
45     for i := 1; i <= len(sdArray); i++ {
46         for j := 1; j <= int(sdArray[i-1][0].GetDimension()); j++ {
47             dataset[i] = append(dataset[i], fmt.Sprintf("%v", sdArray[i-1][0].
48                 GetCoor(j)))
49         }
50         for j := 1; j <= int(sdArray[i-1][1].GetDimension()); j++ {
51             dataset[i+1] = append(dataset[i+1], fmt.Sprintf("%v", sdArray[i-
52                 1][1].GetCoor(j)))
53         }
54     }
55
56     csvFile, err := os.Create("../src/plot/ppoints.csv")
57     if err != nil {
58         fmt.Println("\n!!Failed to export ppoints.!!")
59     }
60
61     csvWriter := csv.NewWriter(csvFile)
62     for _, row := range dataset {
63         _ = csvWriter.Write(row)
64     }
65     csvWriter.Flush()

```

```
63     csvFile.Close()  
64 }
```

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.

```
1 package io  
2  
3 import "fmt"  
4  
5 // Initial screen display  
6 func InitScreen() {  
7     welcomeString := "Closest Pair N-D Points"  
8     fmt.Printf("%s %*s\n", "+++", len(welcomeString)+4, "+++)")  
9     fmt.Printf("+++ %s +++\n", welcomeString)  
10    fmt.Printf("%s %*s\n\n", "+++", len(welcomeString)+4, "+++)")  
11 }
```

Source Code 3: screen.go

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

```
1 package point  
2  
3 import (  
4     "fmt"  
5     "math"  
6     "math/rand"  
7     "time"  
8 )  
9  
10 const float64EqualityTracehold = 1e-12  
11  
12 // Definition of point object  
13 type Point struct {  
14     dimension uint32  
15     coordinate []float64  
16 }  
17  
18 // Creating a new point instance  
19 func CreatePoint(coordinate []float64, dimension uint32) Point {  
20     return Point{  
21         coordinate: coordinate,  
22         dimension: dimension,  
23     }  
24 }  
25  
26 // Getting coordinates of a point in string format  
27 func (p *Point) GetCoors() string {  
28     var sCoor string = "("  
29     for i := 0; i < int(p.dimension); i++ {  
30         if i != 0 {  
31             sCoor += ","  
32         }  
33         sCoor += fmt.Sprintf("%0.3f", p.coordinate[i])  
34     }  
35     sCoor += ")"  
36     return sCoor  
37 }  
38 }
```

```

39 // Getter for coordinate
40 func (p *Point) GetCoor(index int) float64 {
41     return p.coordinate[index-1]
42 }
43
44 // Getter for dimension
45 func (p *Point) GetDimension() uint32 {
46     return p.dimension
47 }
48
49 // Generating randoms point
50 func GeneratePoints(N uint32, dimension uint32) []Point {
51     var points []Point
52     rand.Seed(time.Now().UnixNano())
53
54     for i := 0; i < int(N); i++ {
55         coordinate := make([]float64, dimension)
56         for j := 0; j < int(dimension); j++ {
57             sign := rand.Intn(2)
58             coordinate[j] = float64(rand.Float64() * 1000 * math.Pow(-1, float64(sign)))
59         }
60         points = append(points, CreatePoint(coordinate, dimension))
61     }
62
63     pointsX := make([]Point, len(points))
64     copy(pointsX, points)
65
66     sortedPoints := QuickSortPoints(pointsX)
67
68     return sortedPoints
69 }
70
71 // Sorting []Point using Quick Sort Algorithm
72 func QuickSortPoints(points []Point) []Point {
73     if len(points) < 2 {
74         return points
75     }
76
77     left, right := 0, len(points)-1
78     pivot := rand.Int() % len(points)
79
80     points[pivot], points[right] = points[right], points[pivot]
81     for i := range points {
82         if points[i].GetCoor(1) < points[right].GetCoor(1) {
83             points[left], points[i] = points[i], points[left]
84             left++
85         }
86     }
87
88     points[left], points[right] = points[right], points[left]
89     QuickSortPoints(points[:left])
90     QuickSortPoints(points[left+1:])
91
92     return points
93 }
94
95 // Calculate the Euclidian Distance between point p1 and p2
96 func CalculateDistance(p1, p2 Point) float64 {
97     var sum float64 = 0
98     for i := 1; i <= int(p1.GetDimension()); i++ {

```

```

99     sum += math.Pow(p2.GetCoor(i)-p1.GetCoor(i), 2)
100 }
101 return math.Sqrt(sum)
102 }
103
104 // Check if two floating point number is equal
105 func IsEqual(a, b float64) bool {
106     return math.Abs(a-b) <= float64EqualityTracehold
107 }
108
109 // Print formated [][]Point from slice of shortest distance array
110 func PrintFormattedPoint(sdArray [][]Point) {
111     for i := 0; i < len(sdArray); i++ {
112         fmt.Printf("%*s %s %s\n", len(sdArray[i][0].GetCoors())+22, sdArray[i]
113             ][0].GetCoors(), "<->", sdArray[i][1].GetCoors())
114     }

```

Source Code 4: point.go

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

```

1 package algorithm
2
3 import (
4     "math"
5
6     "github.com/sozyGithub/project/tucil_2/src/point"
7 )
8
9 // Brute Force Algorithm implementation
10 func DoBruteForce(points []point.Point, totalOpt *int) (float64, [][]point
11 .Point) {
12     var shortestDistance float64 = math.MaxFloat64
13     var sdArray [][]point.Point
14     for i := 0; i < len(points); i++ {
15         for j := i + 1; j < len(points); j++ {
16             *totalOpt++
17             distance := point.CalculateDistance(points[i], points[j])
18             if distance < shortestDistance {
19                 shortestDistance = distance
20                 sdArray = nil
21                 sdArray = append(sdArray, []point.Point{points[i], points[j]})}
22             } else if point.AreEqual(distance, shortestDistance) {
23                 sdArray = append(sdArray, []point.Point{points[i], points[j]})}
24             }
25     }
26
27     return shortestDistance, sdArray
28 }

```

Source Code 5: bruteforce.go

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

```

1 package algorithm
2
3 import (
4     "math"
5

```

```

6   "github.com/sozyGithub/project/tucil_2/src/point"
7 )
8
9 // Divide and Conquer Algorithm implementation
10 func DoDNC(pointsX []point.Point, totalOpt *int) (float64, [][]point.Point
11 ) {
12     n := len(pointsX)
13     dimension := pointsX[0].GetDimension()
14
15     var sPoint1 point.Point
16     var sPoint2 point.Point
17     var shortestDistance float64
18     var sdArray = make([][]point.Point, 0)
19
20     // Base cases
21     if n == 2 {
22         *totalOpt++
23         var pPoint = []point.Point{pointsX[0], pointsX[1]}
24         sdArray = append(sdArray, pPoint)
25         return point.CalculateDistance(pointsX[0], pointsX[1]), sdArray
26     }
27     if n == 3 {
28         var pPoint = []point.Point{sPoint1, sPoint2}
29         sdArray = append(sdArray, pPoint)
30         shortestDistance, sdArray = DoBruteForce(pointsX, totalOpt)
31         return shortestDistance, sdArray
32     }
33
34     // Divide
35     midPoint := pointsX[n/2]
36     deltaLeft, sdArrayL := DoDNC(pointsX[0:n/2], totalOpt)
37     deltaRight, sdArrayR := DoDNC(pointsX[n/2:n], totalOpt)
38     if deltaLeft < deltaRight {
39         shortestDistance = deltaLeft
40         temp := make([][]point.Point, len(sdArrayL))
41         copy(temp, sdArrayL)
42         sdArray = append(sdArray, temp...)
43     } else if deltaLeft > deltaRight {
44         shortestDistance = deltaRight
45         temp := make([][]point.Point, len(sdArrayR))
46         copy(temp, sdArrayR)
47         sdArray = append(sdArray, temp...)
48     } else {
49         sdArrayL = append(sdArrayL, sdArrayR...)
50         sdArray = append(sdArray, sdArrayL...)
51     }
52
53     leftPoints := pointsX[0 : n/2]
54     rightPoints := pointsX[n/2 : n]
55
56     idxL := 0
57     idxR := 0
58     inLeftPoints := make([]point.Point, n/2+1)
59     inRightPoints := make([]point.Point, n/2+1)
60     for _, pointL := range leftPoints {
61         if pointL.GetCoor(1) >= midPoint.GetCoor(1)-shortestDistance {
62             inLeftPoints[idxL] = pointL
63             idxL++
64         }
65     }
66     for _, pointR := range rightPoints {

```

```

66     if pointR.GetCoor(1) <= midPoint.GetCoor(1)+shortestDistance {
67         inRightPoints[idxR] = pointR
68         idxR++
69     }
70 }
71
72 // Conquer
73 for i := 0; i < idxL; i++ {
74     for j := 0; j < idxR; j++ {
75         skip := false
76         for k := 1; k <= int(dimension); k++ {
77             if math.Abs(inLeftPoints[i].GetCoor(k)-inRightPoints[j].GetCoor(k))
78             ) > shortestDistance {
79                 skip = true
80                 break
81             }
82         if skip {
83             continue
84         }
85         *totalOpt++
86         pointDist := point.CalculateDistance(inLeftPoints[i], inRightPoints[
87 j])
88         if pointDist < shortestDistance {
89             shortestDistance = pointDist
90             sdArray = nil
91             sdArray = append(sdArray, []point.Point{inLeftPoints[i],
92 inRightPoints[j]})
93         } else if point.Equal(pointDist, shortestDistance) {
94             sdArray = append(sdArray, []point.Point{inLeftPoints[i],
95 inRightPoints[j]})
96         }
97     }
98 }
99
100 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*.

```

1 package main
2
3 import (
4     "fmt"
5     "time"
6
7     "github.com/klauspost/cpuid/v2"
8     "github.com/matishsiao/goInfo"
9     "github.com/pbnjay/memory"
10    "github.com/sozyGithub/project/tucil_2/src/algorithm"
11    "github.com/sozyGithub/project/tucil_2/src/io"
12    "github.com/sozyGithub/project/tucil_2/src/point"
13 )
14
15 func main() {
16     io.InitScreen()
17
18     N, dimension := io.Input()
19     pointsX := point.GeneratePoints(N, dimension)

```

```

20     gi, _ := goInfo.GetInfo()
21
22     fmt.Println("\n---\n\n")
23
24     // Brute Force
25     startTimeBF := time.Now()
26     totalOptBF := 0
27     shortestDistanceBF, sdArrayBF := algorithm.DoBruteForce(pointsX, &
28         totalOptBF)
28     elapsedTimeBF := time.Since(startTimeBF).Seconds()
29
30     // Divide and Conquer
31     startTimeDNC := time.Now()
32     totalOptDNC := 0
33     shortestDistanceDNC, sdArrayDNC := algorithm.DoDNC(pointsX, &totalOptDNC
34         )
34     elapsedTimeDNC := time.Since(startTimeDNC).Seconds()
35
36     // Brute Force Data
37     fmt.Println("== BRUTE FORCE ==")
38     fmt.Printf("%-20s: %f\n", "Shortest Distance", shortestDistanceBF)
39     fmt.Printf("%-20s: \n", "Points")
40     point.PrintFormattedPoint(sdArrayBF)
41     fmt.Printf("%-20s: %d\n", "Total Operations", totalOptBF)
42     fmt.Printf("%-20s: %v s\n", "Total Time", elapsedTimeBF)
43     fmt.Println("\n---\n\n")
44
45     // Divide and Conquer Data
46     fmt.Println("== DIVIDE AND CONQUER ==")
47     fmt.Printf("%-20s: %f\n", "Shortest Distance", shortestDistanceDNC)
48     fmt.Printf("%-20s: \n", "Points")
49     point.PrintFormattedPoint(sdArrayDNC)
50     fmt.Printf("%-20s: %d\n", "Total Operations", totalOptDNC)
51     fmt.Printf("%-20s: %v s\n", "Total Time", elapsedTimeDNC)
52     fmt.Println("\n---\n\n")
53
54     // Computer Specifications
55     fmt.Println("== COMPUTER SPECIFICATIONS ==")
56     fmt.Printf("%-20s: %v\n", "Operating System", gi.OS)
57     fmt.Printf("%-20s: %v GB\n", "Memory", float64(memory.TotalMemory())/
58         float64(1000000000))
59     fmt.Printf("%-20s: %s\n", "CPU Name", cpuid.CPU.BrandName)
60     fmt.Printf("%-20s: %d\n", "Physical Cores", cpuid.CPU.PhysicalCores)
61     fmt.Printf("%-20s: %d\n", "Logical Cores", cpuid.CPU.LogicalCores)
62     if dimension == 3 {
63         fmt.Println("\n---")
64         fmt.Println("Visualisasi titik yang dihasilkan dapat dilihat dengan
65             menjalankan perintah 'make run-visualizer' pada terminal")
66         fmt.Println("---")
67         io.WritePointsToCSV(pointsX)
68         io.WritePPointsToCSV(sdArrayDNC)
69     }
68 }
```

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.

```

1 import matplotlib.pyplot as plt
2 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 - ($N = 16, d = 3$)

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

Points Amount: 10000000
Dimension: -1

Points Amount Must Be a Number Between 2 - 100000. Please Try Again!
Dimension Must Be a Number Between 1 - 10. Please Try Again!

Points Amount: -1
Dimension: 3

Points Amount Must Be a Number Between 2 - 100000. Please Try Again!

Points Amount: tsuki ga kirei desune
Dimension: iie

Points Amount Must Be a Number Between 2 - 100000. Please Try Again!
Dimension Must Be a Number Between 1 - 10. Please Try Again!

Points Amount: 100
Dimension: yah:(

Dimension Must Be a Number Between 1 - 10. Please Try Again!

Points Amount: 16
Dimension: 3
```

Gambar 1: Test Case 1 - Input dan validasi input

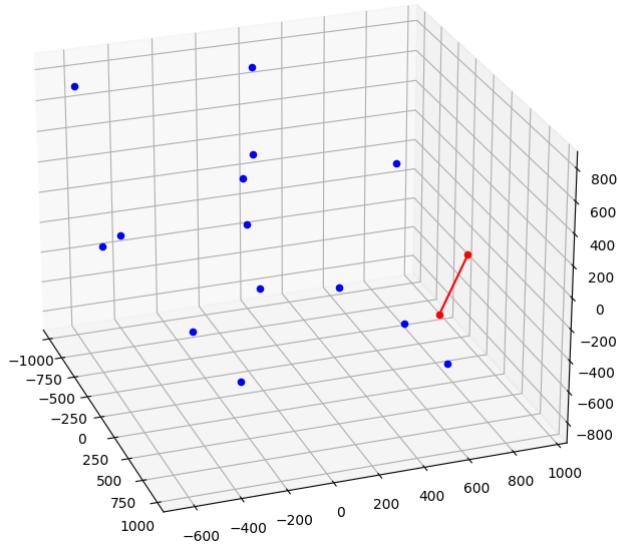
```
—
= BRUTE FORCE =
Shortest Distance   : 363.931372
Points              :
                    (38.231,929.146,-216.495) ↔ (139.054,767.987,-526.831)
Total Operations    : 120
Total Time          : 2.3778e-05 s
—

—
= DIVIDE AND CONQUER =
Shortest Distance   : 363.931372
Points              :
                    (38.231,929.146,-216.495) ↔ (139.054,767.987,-526.831)
Total Operations    : 15
Total Time          : 1.5302e-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 2: Test Case 1 - Output



Gambar 3: Test Case 1 - Visualisasi

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

```
+++
+++ Closest Pair N-D Points ***
+++
Points Amount: 16
Dimension: 10
```

Gambar 4: Test Case 2 - Input

```
=====
= BRUTE FORCE =
Shortest Distance : 1460.606589
Points :
(-140.414,325.367,434.052,-71.483,228.573,-333.501,987.819,767.010,-467.737,894.017) ↔ (454.040,892.631,526.321,216.300,
345.977,-845.845,868.976,-127.471,-933.058,649.371)
Total Operations : 120
Total Time : 5.9301e-05 s

=====
= DIVIDE AND CONQUER =
Shortest Distance : 1460.606589
Points :
(-140.414,325.367,434.052,-71.483,228.573,-333.501,987.819,767.010,-467.737,894.017) ↔ (454.040,892.631,526.321,216.300,
345.977,-845.845,868.976,-127.471,-933.058,649.371)
Total Operations : 67
Total Time : 4.5462e-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
```

Gambar 5: Test Case 2 - Output

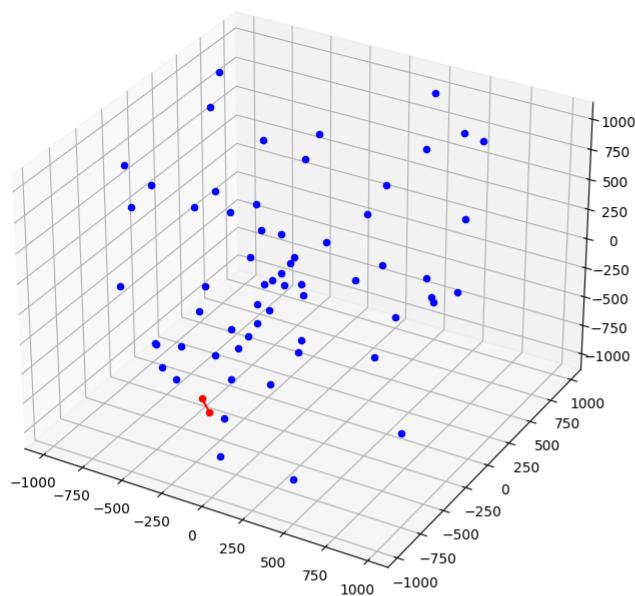
3.3 Test Case 3 - ($N = 64, d = 3$)

```
+++
+++ 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 - ($N = 64, d = 10$)

```
+++
+++ Closest Pair N-D Points ***
+++
Points Amount: 64
Dimension: 10
```

Gambar 9: Test Case 4 - Input

```
—
= BRUTE FORCE =
Shortest Distance : 1014.621245
Points :
        ((66.951,573.516,-142.219,-600.467,520.060,719.012,-448.668,278.702,-701.887,23.874) ↔ (612.011,623.378,192.093,-970.390,
303.060,855.229,-242.864,-84.276,-222.243,-83.185)
Total Operations : 2016
Total Time : 0.000977924 s
—

= DIVIDE AND CONQUER =
Shortest Distance : 1014.621245
Points :
        ((66.951,573.516,-142.219,-600.467,520.060,719.012,-448.668,278.702,-701.887,23.874) ↔ (612.011,623.378,192.093,-970.390,
303.060,855.229,-242.864,-84.276,-222.243,-83.185)
Total Operations : 322
Total Time : 0.000237644 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
```

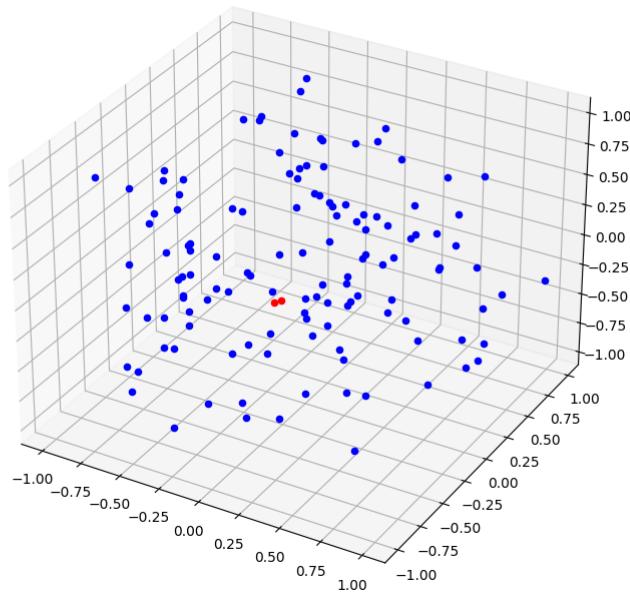
Gambar 10: Test Case 4 - Output

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

```
+++
+++ Closest Pair N-D Points ***
+++
Points Amount: 128
Dimension: 3
```

Gambar 11: Test Case 5 - Input

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

```

—
= BRUTE FORCE =
Shortest Distance : 251102.862343
Points :
(16813.547,537490.139,-161311.299,165369.398,-612610.247,101607.365) ↔ (76462.778,705387.394,-277293.822,79727.212,-5131
50.716,126635.360)
Total Operations : 8128
Total Time : 0.002218933 s

—
= DIVIDE AND CONQUER =
Shortest Distance : 251102.862343
Points :
(16813.547,537490.139,-161311.299,165369.398,-612610.247,101607.365) ↔ (76462.778,705387.394,-277293.822,79727.212,-5131
50.716,126635.360)
Total Operations : 229
Total Time : 0.000184863 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

```

Gambar 15: Test Case 6 - Output

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

```

+++                               +++
+++ Closest Pair N-D Points ++++
+++                               +++
Points Amount: 1000
Dimension: 3

```

Gambar 16: Test Case 7 - Input

```

—
= BRUTE FORCE =
Shortest Distance : 13337.923071
Points :
(-763273.227,-242873.117,-984800.612) ↔ (-751061.657,-242587.372,-990157.487)
Total Operations : 499500
Total Time : 0.070817064 s

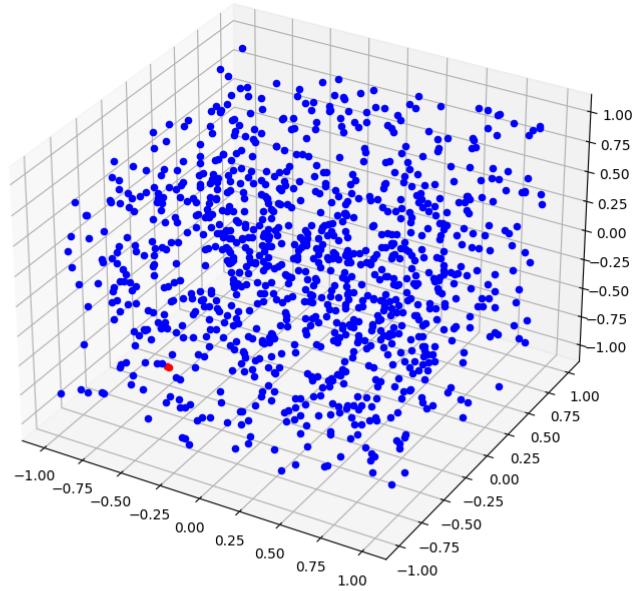
—
= DIVIDE AND CONQUER =
Shortest Distance : 13337.923071
Points :
(-763273.227,-242873.117,-984800.612) ↔ (-751061.657,-242587.372,-990157.487)
Total Operations : 1023
Total Time : 0.0006666536 s

—
= 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 17: Test Case 7 - Output



Gambar 18: Test Case 7 - Visualisasi

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

```
+++
+++ Closest Pair N-D Points ***
+++
Points Amount: 1000
Dimension: 5
```

Gambar 19: Test Case 8 - Input

```
—
= BRUTE FORCE =
Shortest Distance : 120951.511578
Points : (347399.629,304464.375,536405.115,-951165.858,-277124.040) ↔ (430595.834,300967.928,596794.324,-988418.077,-328707.468)
Total Operations : 499500
Total Time : 0.124396604 s

—
= DIVIDE AND CONQUER =
Shortest Distance : 120951.511578
Points : (347399.629,304464.375,536405.115,-951165.858,-277124.040) ↔ (430595.834,300967.928,596794.324,-988418.077,-328707.468)
Total Operations : 1436
Total Time : 0.002071646 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
```

Gambar 20: Test Case 8 - Output

4 Repository

https://github.com/sozyGithub/Tucil2_13521150

5 Checklist

Poin	Ya	Tidak
Program berhasil dikompilasi tanpa kesalahan	✓	
Program berhasil <i>running</i>	✓	
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](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](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Divide-and-Conquer-(2021)-Bagian2.pdf)