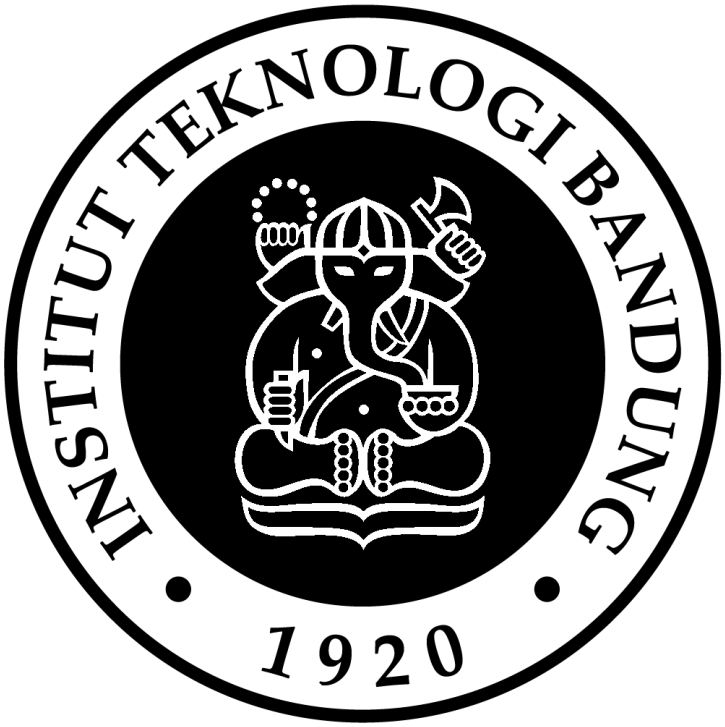
TUGAS KECIL 2 IF2211 STRATEGI ALGORITMA



Disusun Oleh:

13521131	Jeremya Dharmawan Raharjo
13521157	Hanif Muhammad Zhafran

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

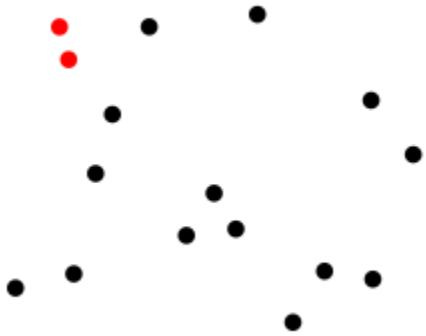
INSTITUT TEKNOLOGI BANDUNG

2023

DAFTAR ISI

BAB I
DESKRIPSI MASALAH

1.1 Deskripsi Masalah



Permasalah Sepasang Dua Titik Terdekat (*Closest Pair of Points*) merupakan permasalahan mencari sepasang titik terdekat dalam himpunan titik yang berada pada $\mathbb{R}^n$. Penyelesaian pencarian sepasang titik ini dapat dilakukan dengan membandingkan semua titik-titik yang berada pada himpunan dengan teknik *brute force*.

Pemecahan masalah tersebut dengan mencari jarak euclidian terkecil antara dua pasang titik di himpunan titik tersebut. Terdapat cara yang lebih efisien dengan menggunakan teknik *divide and conquer*, dimana memecah himpunan titik yang ada menjadi 2 buah upahimpunan titik kemudian mencari jarak minimumnya terhadap masalah yang sudah dipecah tersebut.

BAB II
METODE PENYELESAIAN

1.1 Definisi General Algoritma Divide and Conquer

Algoritma *divide and conquer* merupakan strategi pemecahan masalah dengan melakukan dekomposisi atau pemecahan masalah menjadi upa masalah yang lebih kecil. Adapun proses ini dalam bidang ilmu komputer secara umum dibagi menjadi tiga buah proses:

- Divide: membagi persoalan menjadi beberapa upa-persoalan yang memiliki kemiripan dengan persoalan semula namun berukuran lebih kecil (idealnya berukuran hampir sama),
- Conquer (solve): menyelesaikan masing-masing upa-persoalan (secara langsung jika sudah berukuran kecil atau secara rekursif jika masih berukuran besar).
- Combine: mengabungkan solusi masing-masing upa-persoalan sehingga membentuk solusi persoalan semula.

1.2 Penerapan Algoritma Divide and Conquer pada Pencarian Sepasang Titik Terdekat

Dalam kasus pencarian sepasang titik terdekat, algoritma *divide and conquer* dapat diterapkan dengan cara berikut:

1. Bagi himpunan titik menjadi dua bagian sama besar, secara vertikal menggunakan hiperbidang yang melalui titik tengah dari posisi titik terhadap sumbu-x himpunan. Pada pembagian himpunan titik pada $\mathbb{R}^2$ hiperbidangnya berupa garis yang membagi himpunan titik menjadi 2 sub-bidang, sedangkan pada $\mathbb{R}^3$ pembaginya berupa bidang yang berimpit dengan sumbu-x (hiperbidang untuk dimensi tiga). Untuk kasus $\mathbb{R}^n$, pembaginya adalah suatu hiperbidang $\mathbb{R}^{n-1}$ yang berimpit dengan sumbu-x.
2. Cari jarak terdekat pada kedua bagian himpunan titik secara rekursif. Jarak terdekat ini dapat dicari dengan algoritma yang sama secara rekursif, dengan catatan hanya menghitung jarak antara titik di setengah himpunan terdekat dengan hiperbidang pemisah, yaitu hiperbidang yang melalui titik tengah/median dari himpunan.
3. Proses pemisahan dilakukan sampai menyentuh basis, yaitu ketika himpunan titik hasil pemisahan berjumlah 2 atau 3.

4. Tentukan jarak terdekat antara himpunan titik pada area pemisahan yang sama. Untuk kasus 2 titik, jarak akan langsung dihitung antara kedua titik tersebut. Untuk kasus 3 titik, akan dilakukan pengecekan seluruh kemungkinan pasangan dari 3 titik tersebut (*brute force*). Jarak ini dapat dicari dengan menghitung jarak antara semua pasangan titik di himpunan dan memilih jarak terdekat, yang kita sebut dengan δ.
5. Cari jarak terdekat minimum antara sepasang titik yang berbeda pada dua himpunan titik yang dibagi dengan hiperbidang pada kasus sebelumnya. Jarak ini dicari dengan mempertimbangkan hanya titik-titik yang berjarak kurang dari jarak terdekat minimum antara kedua himpunan (δ).
6. Untuk optimasi pada tahap ke 5, ketika salah satu komponen jarak antar dua titik pada suatu pasangan titik lebih besar dari δ, maka pasangan titik tersebut bukan merupakan pasangan titik dengan jarak terkecil. Dengan demikian, pengecekan jarak dapat langsung dilanjutkan ke pasangan titik berikutnya.
7. Jarak terdekat antara sepasang titik terdekat adalah jarak minimum dari ketiga jarak terdekat yang telah dicari.

1.3 Perbandingan Algoritma Divide and Conquer dengan Algoritma Brute Force

Pada Algoritma Brute Force, himpunan titik yang disimpan pada larik akan dibandingkan, mulai dari indeks ke-1 dengan indeks ke-2, ke-3, hingga indeks ke-n, kemudian melakukan perbandingan kembali titik pada indeks ke-2 dengan indeks selanjutnya hingga indeks ke n, begitu seterusmya hingga indeks ke n-1. Iterasi perbandingan ini sebanyak n*(n-1)/2 , sehingga kompleksitas waktunya adalah $O(n^2)$.

Pada Algoritma Divide and Conquer, pencarian titik terdekat dengan membagi himpunan titik menjadi 2 buah upahimpunan titik yang berukuran sama besar yang dibatasi oleh hiperbidang dan mencari nilai minimum diantara keduanya. Setelah itu, proses pencarian jarak minimum pada upahimpunan dengan batas jarak sebesar minimum dari posisi pembagian himpunan titik dengan hiperbidang, apabila menemukan jarak yang lebih kecil, kita memperbarui jarak minimum tersebut (kompleksitas pencarian ini ialah O(n)). Jika kita mengasumsikan dimensi sebagai suatu konstanta maka didapatkan relasi rekursi sebagai berikut:

$$T(n) = \begin{cases} 2T(n/2) + cn & ,n > 2 \\ a & ,n = 2 \end{cases}$$

Menurut teorema Master, kompleksitas waktu T(n) = O(n log n), dimana hal tersebut merupakan pencarian titik yang lebih mangkus dibandingkan algoritma brute force yang memiliki kompleksitas $O(n^2)$.

Namun, generalisasi solusi menjadi $\mathbb{R}^n$ akan membuat dimensi menjadi suatu variabel. Sehingga, kompleksitas waktunya akan menjadi T(n) = 2T(n/2) + O(nd). Oleh karena itu, untuk dimensi yang tinggi, algoritma *brute force* akan mempunyai performa yang lebih baik dibandingkan algoritma *divide and conquer.*

BAB III

EKSPERIMEN

Test case berikut diujikan pada laptop dengan spesifikasi sebagai berikut:

Processor: AMD Ryzen 7 5800U with Radeon Graphics (16 CPU, ~1.9GHz)

RAM: 16GB

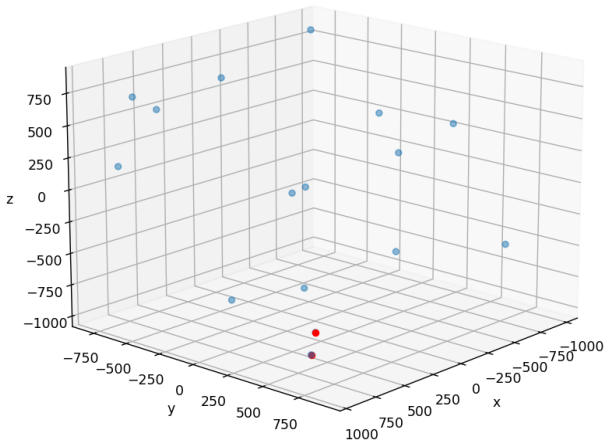
3.1 Kasus Input n = 16 dan dim = 3

```
Input the number of points ( > 1) = 16
Input dimension ( > 0) = 3

==================BRUTE FORCE==================
Point 1                    : [ 857.97476317  708.6677284  -666.67782966]
Point 2                    : [ 832.89294233  658.54516714 -862.40140964]
Minimum distance           : 203.59049251560066
Euclidean distance counter  : 120
Computation time           : 0.00046030001249164343s

===============DIVIDE AND CONQUER================
Point 1                    : [ 857.97476317  708.6677284  -666.67782966]
Point 2                    : [ 832.89294233  658.54516714 -862.40140964]
Minimum distance           : 203.59049251560066
Euclidean distance counter : 23
Computation time           : 0.0005572000227402896s
Visualize the result? (y/n)y
```

Closest Point in 3D plot



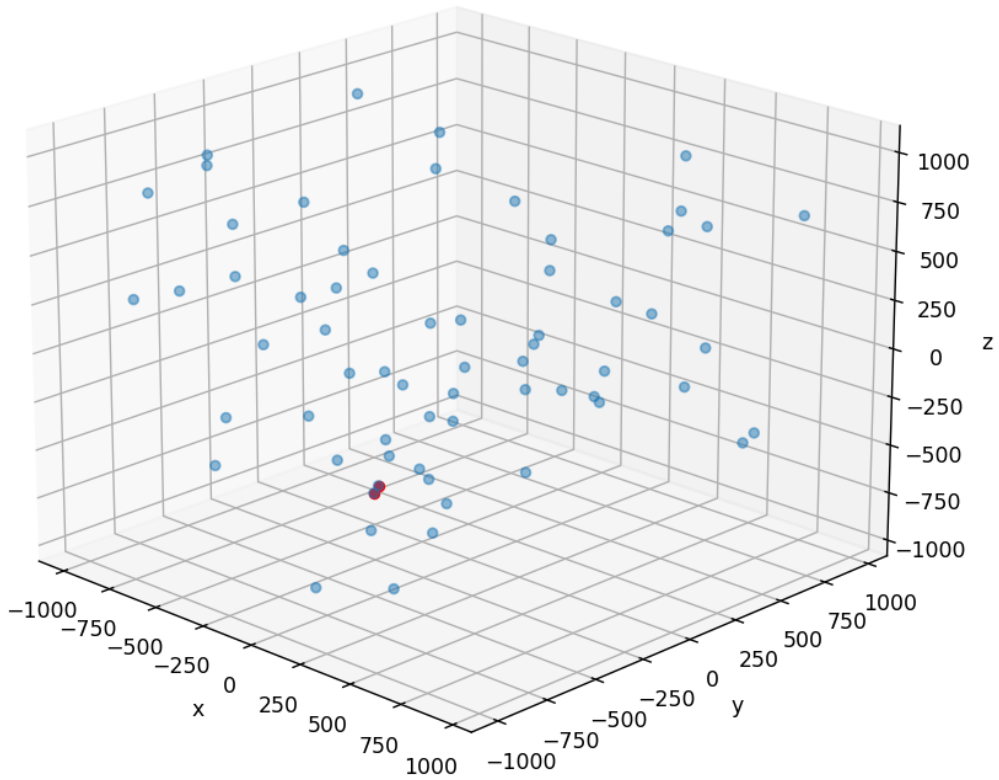
3.2 Kasus Input n = 64 dan dim = 3

```
Input the number of points ( > 1) = 64
Input dimension ( > 0) = 3

==================BRUTE FORCE==================
Point 1                    : [-353.86138247 -127.46395065 -775.10254837]
Point 2                    : [-355.89378441 -149.03888925 -806.88976658]
Minimum distance           : 38.47123438274024
Euclidean distance counter  : 2016
Computation time           : 0.006458100018789992s

==============DIVIDE AND CONQUER================
Point 1                    : [-353.86138247 -127.46395065 -775.10254837]
Point 2                    : [-355.89378441 -149.03888925 -806.88976658]
Minimum distance           : 38.47123438274024
Euclidean distance counter : 112
Computation time           : 0.0024150000244844705s
Visualize the result? (y/n)y
```

Closest Point in 3D plot



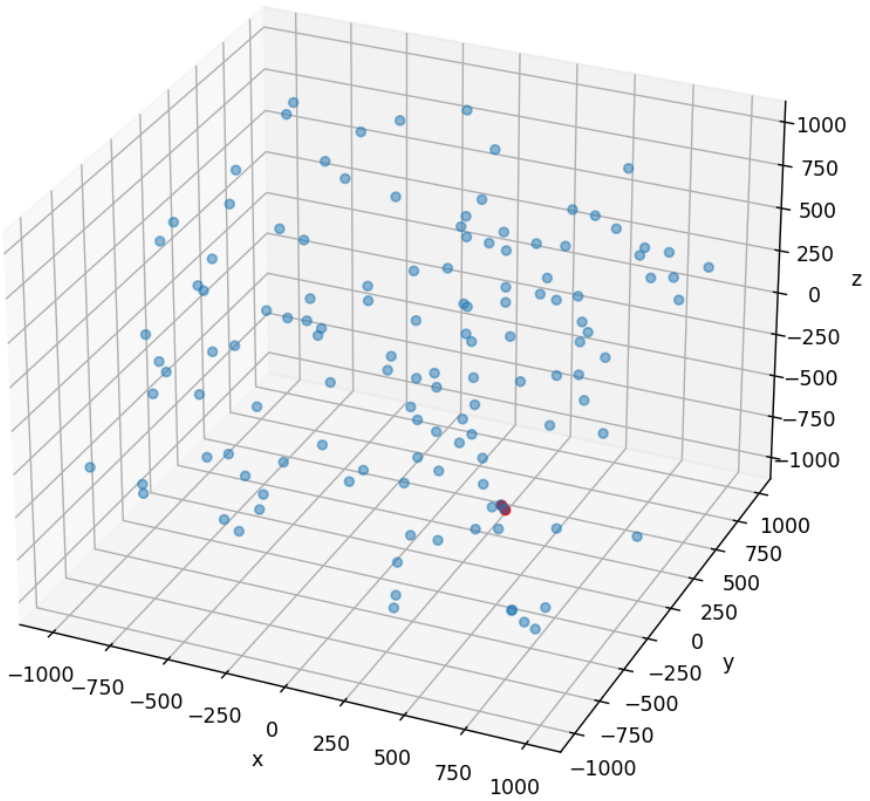
3.3 Kasus Input n = 128 dan dim = 3

```
Input the number of points ( > 1) = 128
Input dimension ( > 0) = 3

==================BRUTE FORCE==================
Point 1                    : [ 813.07328003 -963.44631499   47.02190691]
Point 2                    : [ 822.59979779 -952.67672986   18.4491332 ]
Minimum distance           : 31.986589412642736
Euclidean distance counter  : 8128
Computation time           : 0.024020300013944507s

===============DIVIDE AND CONQUER================
Point 1                    : [ 813.07328003 -963.44631499   47.02190691]
Point 2                    : [ 822.59979779 -952.67672986   18.4491332 ]
Minimum distance           : 31.986589412642736
Euclidean distance counter : 261
Computation time           : 0.004558999993605539s
Visualize the result? (y/n)y
```

Closest Point in 3D plot



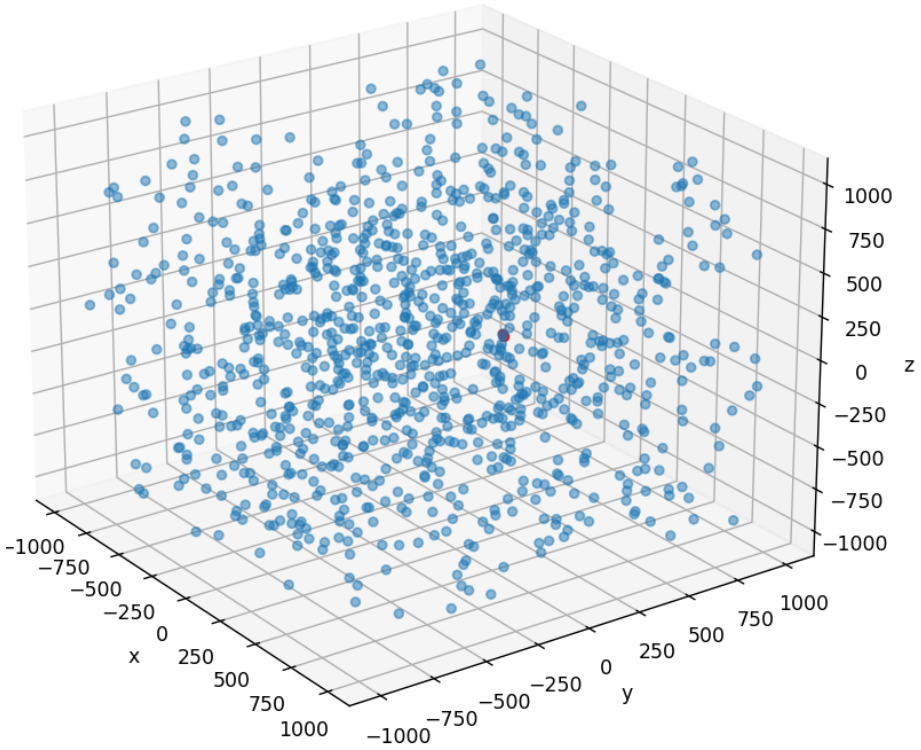
3.4 Kasus Input n = 1000 dan dim = 3

```
Input the number of points ( > 1) = 1000
Input dimension ( > 0) = 3

==================BRUTE FORCE==================
Point 1                     : [ -47.71196066  450.20680936 -163.28924379]
Point 2                     : [ -41.96087958  449.11446645 -175.57908958]
Minimum distance            : 13.61280486020232
Euclidean distance counter  : 499500
Computation time            : 1.3678385999810416s

==============DIVIDE AND CONQUER================
Point 1                     : [ -47.71196066  450.20680936 -163.28924379]
Point 2                     : [ -41.96087958  449.11446645 -175.57908958]
Minimum distance            : 13.61280486020232
Euclidean distance counter  : 2866
Computation time            : 0.050683499983279034s
Visualize the result? (y/n)y
```

Closest Point in 3D plot



3.5 Bonus 2: Kasus Input n = 64 dan dim = 8

```
Input the number of points ( > 1) = 64
Input dimension ( > 0) = 8

==================BRUTE FORCE==================
Point 1                    : [-216.41151793 -570.08176929  527.81397033  280.38969907 -370.1072103
  560.1612803  -719.3676278   908.83644244]
Point 2                    : [-496.30700631 -713.83206715  535.1253758   506.31446714 -639.69985023
  801.16839667 -710.87892351  576.12134511]
Minimum distance           : 625.8091608241319
Euclidean distance counter  : 2016
Computation time           : 0.009416200016858056s

===============DIVIDE AND CONQUER================
Point 1                    : [-216.41151793 -570.08176929  527.81397033  280.38969907 -370.1072103
  560.1612803  -719.3676278   908.83644244]
Point 2                    : [-496.30700631 -713.83206715  535.1253758   506.31446714 -639.69985023
  801.16839667 -710.87892351  576.12134511]
Minimum distance           : 625.8091608241319
Euclidean distance counter : 226
Computation time           : 0.005627999984426424s
```

3.6 Bonus 2: Kasus Input n = 1000 dan dim = 12

```
Input the number of points ( > 1) = 1000
Input dimension ( > 0) = 12

==================BRUTE FORCE==================
Point 1                    : [-746.90036158  832.2289101   194.97108779 -365.85972271  957.16159094
 -563.34720404  105.42132528 -973.771321    -118.27473788 -715.13663209
  562.88863477  305.88085955]
Point 2                    : [-820.76797011  700.33126832  553.4818023  -617.5924485   939.89970599
 -383.51243181   45.96393511 -743.2142424   -443.60134006 -448.48211966
  388.91786398  265.7407445 ]
Minimum distance           : 716.1718089083697
Euclidean distance counter  : 499500
Computation time           : 2.529876700020395s

===============DIVIDE AND CONQUER================
Point 1                    : [-746.90036158  832.2289101   194.97108779 -365.85972271  957.16159094
 -563.34720404  105.42132528 -973.771321    -118.27473788 -715.13663209
  562.88863477  305.88085955]
Point 2                    : [-820.76797011  700.33126832  553.4818023  -617.5924485   939.89970599
 -383.51243181   45.96393511 -743.2142424   -443.60134006 -448.48211966
  388.91786398  265.7407445 ]
Minimum distance           : 716.1718089083697
Euclidean distance counter : 13951
Computation time           : 0.635912500001723s
```

3.7 Bonus 2: Kasus Input n = 100 dan dim = 100

```
Input the number of points ( > 1) = 100
Input dimension ( > 0) = 100

==================BRUTE FORCE===================
Point 1                    : [ 2.59221859e+02  7.20166685e+02  4.86883694e+02  9.83657835e+02
  8.72875833e+02 -8.44769761e+02 -7.32536876e+02  6.47541416e+02
  3.35192043e+02 -2.14949439e+02 -6.16777399e+02  3.62555523e+01
  6.40206290e+01  3.30686909e+02 -6.61681919e+02  4.89021732e+02
  1.83515223e+02  2.98376342e+02 -6.30326256e+02  1.06826967e+02
 -5.18767744e+02  6.15890584e+02 -5.51077799e+02  9.77109603e+01
 -6.06057207e+02 -8.88698850e+02  3.28671601e+02 -3.84778459e+02
 -2.31668150e+02  1.13970205e+02  1.09414515e+02  7.43255933e+00
 -4.38322837e+02  7.66799803e+02  2.51956066e+02 -3.97375537e+02
  8.39492503e+02  7.05310167e+02  9.35682625e+02 -4.73823666e+02
 -5.57160298e+02  6.13362729e+02 -4.23565696e+02  9.61174486e+02
  7.98913231e+02  5.25769123e+02 -6.91462553e+02 -3.77363518e+02
 -6.87801348e+02 -8.35432796e+01  8.39159657e+02 -6.42638794e+02
 -4.70538707e+02 -5.27134828e+02  3.17265153e+02 -2.62167187e+02
 -7.18855676e+02  5.01972125e+02 -1.56520341e+02 -4.96243092e+02
 -3.45381159e+02 -9.79299717e+02  7.52133260e+02  5.50209551e+02
  6.34993964e+02 -9.72830750e+02 -9.86571085e-01  3.86881502e+02
  6.60578241e+02  8.47683831e+02  7.72713092e+02 -4.60527551e+02
  7.82826102e+02 -7.59989648e+02 -4.36859037e+02  1.32292096e+02
 -9.92454747e+01  9.97266542e+02 -2.34447405e+02 -2.52628721e+00
 -6.77156876e+02 -3.75539545e+02 -1.71087476e+02  3.46449913e+02
  7.47209581e+02 -1.45152777e+01 -7.47172655e+02 -3.57752044e+02
 -3.28833172e+02  7.37427530e+01 -1.03715664e+02  2.99352492e+02
 -2.42955735e+02  5.13016125e+01 -4.06524214e+02  5.74250020e+02
  5.12934179e+02  1.16134970e+02  9.89350934e+02 -7.52509813e+02]
Point 2                    : [ 452.05597515  916.88735615  634.19500267   37.14603091 -333.10476161
  305.03352428 -329.61222212  400.483426     -8.1915042   807.29132306
 -278.22920335 -470.80391241  980.12530292  596.04081305  125.66385141
  564.0892826   207.97710642 -599.98757692 -110.94435371 -931.10560493
 -318.95062728  324.67461951 -173.32367091 -235.92456877  532.14233218
 -463.82256229  372.79421328  239.32799817  710.16980434  748.34167751
 -669.92733137  288.7729398   588.23027465  358.54129993 -508.32709633
 -384.62730076 -529.57546733 -893.78587247  996.60493855  124.02143255
 -369.04521162  352.0258639  -293.9759622   483.06046765  599.95940258
```

```
 -369.04521162  352.0258639  -293.9759622   483.06046765  599.95940258
  113.02212802 -562.20768373  841.68770789 -309.33988945 -532.41126978
  255.35307713 -198.97586721  769.17298255 -421.48629538  193.36636143
  109.97750388  261.11885678  794.21166893  107.99588047  -65.24286517
   73.16407864  436.83801148 -556.37721776  632.39508997 -315.77385102
 -370.15950318  525.52167461  439.37501248  760.28470603  -78.42339507
  746.38368661  450.41812477  580.65085329 -358.46582267   56.77286692
 -641.94330986 -745.54739037  773.65303169  -11.9010523   -41.15981
 -810.01899907 -644.12834176 -767.38729982 -710.49784012  -35.93728721
 -874.16020488 -906.14539128  205.67802785   49.35213006 -185.21330923
 -571.46980306  105.31261565 -945.63364815  847.30759688  147.67517721
 -673.00359327   91.03446188  229.86829753   -2.1161416   -33.20010318]
Minimum distance           : 6564.544978483163
Euclidean distance counter  : 4950
Computation time           : 0.023974999989150092s

==============DIVIDE AND CONQUER===============
Point 1                    : [ 2.59221859e+02  7.20166685e+02  4.86883694e+02  9.83657835e+02
  8.72875833e+02 -8.44769761e+02 -7.32536876e+02  6.47541416e+02
  3.35192043e+02 -2.14949439e+02 -6.16777399e+02  3.62555523e+01
  6.40206290e+01  3.30686909e+02 -6.61681919e+02  4.89021732e+02
  1.83515223e+02  2.98376342e+02 -6.30326256e+02  1.06826967e+02
  6.60578241e+02  8.47683831e+02  7.72713092e+02 -4.60527551e+02
  7.82826102e+02 -7.59989648e+02 -4.36859037e+02  1.32292096e+02
 -9.92454747e+01  9.97266542e+02 -2.34447405e+02 -2.52628721e+00
 -6.77156876e+02 -3.75539545e+02 -1.71087476e+02  3.46449913e+02
  7.47209581e+02 -1.45152777e+01 -7.47172655e+02 -3.57752044e+02
 -3.28833172e+02  7.37427530e+01 -1.03715664e+02  2.99352492e+02
 -2.42955735e+02  5.13016125e+01 -4.06524214e+02  5.74250020e+02
  5.12934179e+02  1.16134970e+02  9.89350934e+02 -7.52509813e+02]
Point 2                    : [ 452.05597515  916.88735615  634.19500267   37.14603091 -333.10476161
  305.03352428 -329.61222212  400.483426     -8.1915042   807.29132306
 -278.22920335 -470.80391241  980.12530292  596.04081305  125.66385141
  564.0892826   207.97710642 -599.98757692 -110.94435371 -931.10560493
 -318.95062728  324.67461951 -173.32367091 -235.92456877  532.14233218
 -463.82256229  372.79421328  239.32799817  710.16980434  748.34167751
 -669.92733137  288.7729398   588.23027465  358.54129993 -508.32709633
 -384.62730076 -529.57546733 -893.78587247  996.60493855  124.02143255
 -369.04521162  352.0258639  -293.9759622   483.06046765  599.95940258
  113.02212802 -562.20768373  841.68770789 -309.33988945 -532.41126978
  255.35307713 -198.97586721  769.17298255 -421.48629538  193.36636143
  109.97750388  261.11885678  794.21166893  107.99588047  -65.24286517
   73.16407864  436.83801148 -556.37721776  632.39508997 -315.77385102
 -370.15950318  525.52167461  439.37501248  760.28470603  -78.42339507
```

```
  109.97750388  261.11885678  794.21166893  107.99588047  -65.24286517
   73.16407864  436.83801148 -556.37721776  632.39508997 -315.77385102
 -370.15950318  525.52167461  439.37501248  760.28470603  -78.42339507
  746.38368661  450.41812477  580.65085329 -358.46582267   56.77286692
 -641.94330986 -745.54739037  773.65303169  -11.9010523   -41.15981
 -810.01899907 -644.12834176 -767.38729982 -710.49784012  -35.93728721
 -874.16020488 -906.14539128  205.67802785   49.35213006 -185.21330923
 -571.46980306  105.31261565 -945.63364815  847.30759688  147.67517721
 -673.00359327   91.03446188  229.86829753   -2.1161416   -33.20010318]
Minimum distance           : 6564.544978483163
Euclidean distance counter : 4950
Computation time           : 0.18778859998565167s
```

BAB IV
CHECKLIST

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

DAFTAR REFERENSI

Munir, Rinaldi. *Homepage Rinaldi Munir*. Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/

Shamos dan Bentley. *DIVIDE-AND-CONQUER IN MULTIDIMENSIONAL SPACE.* Diakses dari http://euro.ecom.cmu.edu/people/faculty/mshamos/1976ShamosBentley.pdf

LAMPIRAN

i. Source Code (Python)

sort.py

```
# Partition for quick sort, only for numpy array
def partition(arr, low, high, key):
      i = low - 1
      pivot = arr[high][key]
      for j in range(low, high):
         if (arr[j][key] <= pivot):
            i += 1
            arr[[i, j]] = arr[[j, i]]

      arr[[high, i + 1]] = arr[[i + 1, high]]
      return i + 1

# Quick sort algorithm
def quick_sort(arr, low, high, key):
      if low < high:
         mid = partition(arr, low, high, key)
         quick_sort(arr, mid + 1, high, key)
         quick_sort(arr, low, mid - 1, key)
```

solve.py

```
import numpy as np
from sort import *
import math
calc_step = 0

# Solve by brute force
def brute_force(arr):
      # Set global variables for euclidean distance counter
      global calc_step

      # Points count
      n = np.shape(arr)[0]

      # Initialize minimum value and min_pair
      min = math.inf
      min_pair = None

      # Search for minimum distance by checking every pair
      for i in range(n - 1):
            for j in range(i + 1, n):
                  calc_step += 1
```

```
                distance = np.linalg.norm(arr[i]-arr[j])
                if (min > distance):
                        min = distance
                        min_pair = np.array([arr[i], arr[j]])

        return min, min_pair

def presorted_nearest_strip(strip_left, strip_right, min_distance,
min_pair):
        # Set global variables for euclidean distance counter
        global calc_step

        # Points dimension
        strip_dim = strip_left.shape[1]

        # Initialize minimum distance
        the_dist = min_distance

        # Check pair of points in the strip
        for p_left in strip_left:
                for p_right in strip_right:

                        # Initialize dim and point_valid
                        dim = 0
                        point_valid = True

                        # If a pair of points have at least one component
aaaaaaaa        with a difference larger than min_distance, skip check

                        while dim < strip_dim - 1:
                                if abs(p_left[dim] - p_right[dim]) > min_distance:
                                        point_valid = False
                                        break
                                else:
                                        dim += 1

                # Point valid only if all components of a pair of
aaaaaaaa          points have difference less than min_distance

                        if (point_valid):
                                distance = np.linalg.norm(p_left - p_right)
                                calc_step += 1
                                if distance < the_dist:
                                        the_dist = distance
                                        min_pair = np.array([p_left, p_right])

        return the_dist, min_pair

def presorted_divide_and_conquer(presortedX):
        # Set global variables for euclidean distance counter
        global calc_step
```

```
    if(presortedX.shape[0] <= 3): # Base case
        return brute_force(presortedX)

    else: # Recurrence
    # Divide
    # Split into two region with the same number of points (at
aaaaaa most 1 difference in number when the total is odd)

        arr_left = presortedX[:presortedX.shape[0] // 2]
        arr_right = presortedX[presortedX.shape[0] // 2:]

    # Get mid strip
        mid = (presortedX[presortedX.shape[0] // 2 - 1][0] +
aaaaaaaaaaaaaaaapresortedX[presortedX.shape[0] // 2][0]) / 2

    # Conquer
    # Solve for each left and right region
        delta_left, min_pair_left =
aaaaaaaaaapresorted_divide_and_conquer(arr_left)
        delta_right, min_pair_right =
aaaaaaaaaapresorted_divide_and_conquer(arr_right)

    # Combine
    # Calculate minimum distance and pair from left and right
aaaaaaa section
        delta = min(delta_left, delta_right)
        min_pair_LR = min_pair_left.copy() if delta ==
aaaaaaaaaadelta_left else min_pair_right.copy()

    # Threshold for strip region
        threshold_left = mid - delta
        threshold_right = mid + delta

    # Get points in the strip region
        arr_strip_left = presortedX[(presortedX[:, 0] >=
aaaaaaaaaathreshold_left) & (presortedX[:, 0] <= mid)]
        arr_strip_right = presortedX[(presortedX[:, 0] <=
aaaaaaaaaathreshold_right) & (presortedX[:, 0] > mid)]

    # Calculate minimum distance and pair in the strip, and
aaaaaa  combine it with the previous minimum distance and pair
        delta_strip, min_pair_strip =
aaaaaaaapresorted_nearest_strip(arr_strip_left, arr_strip_right,
aaaaaaaadelta, min_pair_LR)

        return min(delta, delta_strip), min_pair_strip
```

plot.py

```
import matplotlib.pyplot as plt
```

```
import numpy as np

def visualize3D(arr, pair):

      x = arr[:, 0]
      y = arr[:, 1]
      z = arr[:, 2]

      fig = plt.figure(figsize=(8,8))
      fig.suptitle('Closest Point in 3D plot')
      ax = plt.axes(projection="3d")

      fg = ax.scatter3D(x, y, z, alpha = 0.5)
      fg = ax.scatter3D(pair[0][0], pair[0][1], pair[0][2],
aaaaaaaa  color='r', alpha = 1)
      fg = ax.scatter3D(pair[1][0], pair[1][1], pair[1][2],
aaaaaaaa  color='r', alpha = 1)

      ax.set_xlabel("x-axis")
      ax.set_ylabel("y-axis")
      ax.set_zlabel("z-axis")

      plt.show()
```

main.py

```
from solve import *
from plot import *
import time as t
import solve


n = int(input("Input the number of points ( > 1) = "))
while (n <= 1):
      print("Number of points must be greater than 1")
      n = int(input("Input the number of points ( > 1) = "))

dim = int(input("Input dimension ( > 0) = "))
while (dim <= 0):
      print("Dimension must be greater than 0")
      dim = int(input("Input dimension ( > 0) = "))

x = np.random.uniform(-1000.0, 1000.0, (n, dim))

tic = t.perf_counter()
minima, min_pair_bf = brute_force(x)
toc = t.perf_counter()

print("\n==================BRUTE FORCE==================")
print(f"Point 1                        : {min_pair_bf[0]}")
print(f"Point 2                        : {min_pair_bf[1]}")
```

```
print(f"Minimum distance              : {minima}")
print(f"Euclidean distance counter  : {solve.calc_step}")
print(f"Computation time              : {toc-tic}s")

# Reset euclidean distance counter

solve.calc_step = 0
tic = t.perf_counter()
quick_sort(x, 0, x.shape[0] - 1, 0)
minima, min_pair_dnc = presorted_divide_and_conquer(x)
toc = t.perf_counter()
print("\n===============DIVIDE AND CONQUER===============")
print(f"Point 1                  : {min_pair_bf[0]}")
print(f"Point 2                  : {min_pair_bf[1]}")
print(f"Minimum distance         : {minima}")
print(f"Euclidean distance counter : {solve.calc_step}")
print(f"Computation time         : {toc-tic}s")

if(x.shape[1] == 3):
     vis_valid = input("Visualize the result? (y/n)")
     while vis_valid != "y" and vis_valid != "n":
          print("Input not valid, please choose between y/n")
          vis_valid = input("Visualize the result? (y/n)")

     if vis_valid == "y":
          visualize3D(x, min_pair_dnc)
```

ii. Tautan Repository

Tautan repository tugas kecil: https://github.com/jejejery/Tucil_2_IF2211