

IF2211 - Strategi Algoritma
Mencari Pasangan Titik Terdekat N-Dimensi
dengan Algoritma *Divide and Conquer*
Laporan Tugas Kecil 2



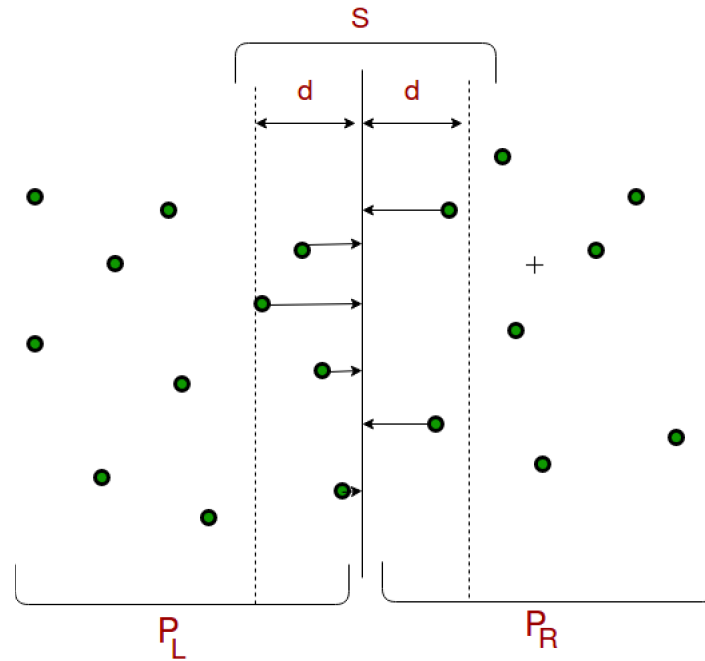
| Nama | NIM |
|---------------|----------|
| William Nixon | 13521123 |
| Nicholas Liem | 13521135 |

Institut Teknologi Bandung
Sekolah Teknik Elektro dan Informatika
Tahun Ajaran 2022/2023

Daftar Isi

| | | |
|----------|----------------------------------|-----------|
| 1 | Deskripsi Masalah | 1 |
| 2 | Pendekatan Solusi | 2 |
| 2.1 | Solusi Dimensi N | 2 |
| 2.2 | Masalah-Masalah | 2 |
| 3 | Implementasi Kode | 4 |
| 3.1 | Main Program | 4 |
| 3.2 | Kelas Point | 6 |
| 3.3 | Kelas PointManager | 8 |
| 4 | Pengujian | 14 |
| 5 | Checklist | 17 |
| 6 | Daftar Pustaka | 18 |
| 7 | Lampiran | 19 |
| 7.1 | Link Repository GitHub | 19 |

1 Deskripsi Masalah



Sumber: [Medium — Algorithms: Divide and Conquer — Closest Pair](#)

Masalah pasangan terdekat dalam ruang n -dimensi adalah sebuah masalah komputasi yang bertujuan untuk mencari dua titik terdekat di antara sekumpulan n titik dalam ruang n -dimensi. Masalah ini memiliki berbagai macam aplikasi, termasuk pengolahan citra, pengenalan pola, dan geometri komputasional. Sebagai contoh, dalam *computer vision*, masalah pasangan terdekat dapat digunakan untuk mencari wilayah yang mirip dalam citra, sedangkan dalam biologi komputasional, masalah ini dapat digunakan untuk mencari kemiripan antara molekul.

Salah satu pendekatan populer untuk menyelesaikan masalah ini adalah dengan menggunakan metode *divide and conquer*. Dalam pendekatan ini, ruang n -dimensi dibagi menjadi dua subruang yang lebih kecil secara rekursif, dan kemudian pasangan titik terdekat di setiap subruang dihitung. Setelah itu, pasangan titik terdekat di antara kedua subruang dipilih. Dalam memilih pasangan terdekat di antara kedua subruang, perlu diperhatikan jarak antara titik di subruang kanan dengan titik di subruang kiri, sehingga tidak terlewatkan pasangan titik terdekat yang melintasi kedua subruang tersebut.

Kompleksitas waktu terbaik untuk menyelesaikan masalah pasangan terdekat dalam ruang n -dimensi menggunakan metode *divide and conquer* adalah $O(n \log n)$, jika titik-titik yang diberikan memenuhi syarat sparsity (densitas tidak terlalu tinggi) sehingga perbandingan pada saat combine menuju konstan. Hal ini lebih baik dari metode lain seperti Brute Force $O(n^2)$ karena harus membandingkan seluruh titik. Oleh karena itu, metode divide and conquer dianggap sebagai metode terbaik untuk menyelesaikan masalah pasangan terdekat dalam ruang n -dimensi.

2 Pendekatan Solusi

2.1 Solusi Dimensi N

Solusi serupa di dalam seluruh dimensi. Pendekatan solusi yang dipakai adalah sebagai berikut.

1. Lakukan sorting membesar pada setiap titik berdasarkan nilai sumbu pertama, sumbu x . Jika nilai sumbu pertama sama, lakukan sorting berdasarkan nilai sumbu selanjutnya (sumbu y) jika ada. Jika masih sama, lanjutkan ke sumbu-sumbu di dimensi selanjutnya.

Divide step adalah sebagai berikut:

2. Bagi bidang menjadi dua sisi (kiri dan kanan) berdasarkan axis pertama (sumbu x), kemudian partisi bidang tersebut hingga mencapai basis yakni satu atau dua titik.
3. Terdapat dua basis. Titik yang hanya berjumlah 1 (sendiri) akan memiliki jarak minimum tak hingga, dan titik yang berjumlah 2 akan memiliki jarak minimum berupa jarak euclidean antara kedua titik tersebut.

Conquer step adalah sebagai berikut:

4. Untuk menyatukan solusinya, kita bandingkan nilai minimum antara jarak titik di bagian partisi kiri dan kanan. Kita mengambil nilai terkecil dari kedua partisi tersebut dan menjadikan nilai tersebut sebagai δ .
5. Carilah titik-titik dalam ruang yang dipisahkan oleh *pivot* dengan lebar δ pada sumbu partisi (sumbu pertama).
6. Bandingkan kumpulan titik yang dekat dengan sumbu partisi pada divisi kiri dan kanan. Sebelum melakukan pemanggilan jarak euclidean antara dua titik, pastikan bahwa seluruh posisi dimensi pada titik-titik tersebut berada di dalam nilai δ . Jika jarak titik berdasarkan semua sumbu lebih kecil dari δ , bandingkan jarak antara titik-titik pada subruang tersebut.
7. Jika ditemukan jarak antar titik kiri dan kanan yang lebih kecil dari δ , maka *update* nilai δ semula. Lakukan secara rekursif sampai seluruh titik awal dibandingkan kembali (langkah combine telah selesai).

2.2 Masalah-Masalah

Ada beberapa masalah atau kasus khusus yang mungkin dapat menyebabkan algoritma ini tidak berjalan dengan maksimal.

1. Salah satu kasusnya adalah ketika titik-titik berimpitan pada *strip pivot* yang jumlahnya sangat banyak atau mendekati N . Hal ini akan menurunkan performa algoritma *divide and conquer* sebab pada tahap conquer tidak akan ada titik yang tereliminasi karena seluruhnya berada di dalam strip δ . Sebagai

akibatnya, syarat perbandingan bersifat $O(1)$ pada tahap merge tidak terjadi. Jika titik tersebar secara normal dan tidak terlalu tinggi densitasnya, maka program seharusnya berjalan dengan lancar.

2. Kasus lain adalah seiring bertambah tingginya dimensi, maka performa algoritma akan semakin buruk. Hal ini karena nilai minimum distance akan menjadi sangat tinggi, jauh lebih tinggi dari jarak pada salah satu sumbu. Mirip pada kasus sebelumnya, tingginya nilai δ ini akan menyebabkan seluruh titik untuk berada di dalam strip tersebut, sehingga perbandingan antar seluruh titik harus tetap dilakukan. Pada kasus ekstrem, performa algoritma *divide and conquer* mungkin menyerupai brute force.

3 Implementasi Kode

3.1 Main Program

```
1 from Class.Point import Point
2 from Class.PointManager import PointManager
3 import time
4 import sys
5
6 def main():
7     pm = PointManager()
8     inputFile = input("Apakah ingin membaca poin dari file? (Y/N)\n")
9     if(inputFile == "Y" or inputFile == "y" or inputFile == "N" or
10        inputFile == "n"):
11         fileName = input("Input nama file (co: tes.txt): ")
12         path = sys.path[0] + "\\Input\\" + fileName
13         pm.readPoints(path)
14     else:
15         dim = int(input("Insert the number of dimensions (dim): "))
16         n = int(input("Insert the number of points (n): "))
17         pm.generateRandomPoints(n, dim)
18
19     pm.mergeSort(pm.getPoints())
20
21     dnc_start_time = time.time()
22     dnc_shortestPairDistance = pm.divideAndConquerSolution()
23     dnc_solPointOne = pm.getDNCsSolPointOne()
24     dnc_solPointTwo = pm.getDNCsSolPointTwo()
25
26     dnc_end_time = time.time()
27     dnc_elapsed_time = dnc_end_time - dnc_start_time
28     print(
29         "
30         ===== "
31     )
32     print(
33         f"[Divide and Conquer] Shortest Pair Distance is {
34         dnc_shortestPairDistance:.2f} of point {dnc_solPointOne.
35         printSelf()} and {dnc_solPointTwo.printSelf()}"
36     )
37     print(
38         f"[Divide and Conquer] Euclidean Distance Calculation Count
39         (Divide and Conquer): {pm.getEuclideanDistanceCount()}"
40     )
41     print(f"[Divide and Conquer] Elapsed time: {dnc_elapsed_time:.6
42         f} seconds")
43
44     print(
45         "
46         ===== "
47     )
48     # reset Euclidean Count
49     pm.resetEuclideanCount()
```

```

44
45     bf_start_time = time.time()
46     bf_shortestPairDistance = pm.bruteForceSolution()
47     bf_solPointOne = pm.getBFSolPointOne()
48     bf_solPointTwo = pm.getBFSolPointTwo()
49     bf_end_time = time.time()
50     bf_elapsed_time = bf_end_time - bf_start_time
51     print(
52         "
===== "
53     )
54
55     print(
56         f"[Brute Force] Shortest Pair Distance is {
bf_shortestPairDistance:.2f} of point {bf_solPointOne.printSelf
()} and {bf_solPointTwo.printSelf()}"
57     )
58     print(
59         f"[Brute Force] Euclidean Distance Calculation Count: {pm.
getEuclideanDistanceCount()}"
60     )
61     print(
62         f"[Brute Force] Elapsed time: {bf_elapsed_time:.6f} seconds
"
63     )
64     print(
65         "
===== "
66     )
67     time.sleep(2)
68
69     answer = ''
70     flag = False
71     while(not flag):
72         answer = input("Do you want to plot the points? Y/N\n")
73         if (answer == "Y" or answer == "y" or answer == "N" or
answer == "n"):
74             flag = not flag
75
76     if (answer == "Y" or answer == "y"):
77         pm.plot()
78     else:
79         print("Program Exit")
80         exit(0)
81
82 if __name__ == "__main__":
83     main()

```

3.2 Kelas Point

```
1 import math
2
3 class Point:
4
5     def __init__(self, *args):
6         self.coords = args[0]
7         self.dimension = len(self.coords)
8         self.solution = False
9
10    def __str__(self):
11        return str(self.coords)
12
13    def __repr__(self):
14        return str(self.coords)
15
16    def setSolution(self):
17        self.solution = True
18
19    def getSolution(self):
20        return self.solution
21
22    def scanNear(self, other, minDist):
23        for ind in range(len(self.coords)):
24            if abs(self.getCoords(ind) - other.getCoords(ind)) >
minDist:
25                return False
26            return True
27
28    def printSelf(self):
29        return f"Point: {self.coords}"
30
31    def getCoords(self, index) -> float:
32        return self.coords[index]
33
34    def getDimension(self) -> int:
35        return self.dimension
36
37    def distanceTo(self, otherPoint) -> float:
38        if self.dimension != otherPoint.getDimension():
39            raise ValueError("Dimension has to be same")
40        else:
41            distance = 0
42            for i in range(self.dimension):
43                distance += (self.getCoords(i) - otherPoint.
getCoords(i)) ** 2
44            return math.sqrt(distance)
45
46    def lessThan(self, otherPoint) -> bool:
47        ax_ind = 0
48        while ax_ind < self.dimension and self.getCoords(
49            ax_ind
50        ) == otherPoint.getCoords(ax_ind):
51            ax_ind += 1
52
```

```
53         if ax_ind < self.dimension and self.getCoords(ax_ind) <
otherPoint.getCoords(
54             ax_ind
55         ):
56             return True
57         else:
58             return False
59
60     def nearPivot(self, pivot, minDist):
61         if abs(self.getCoords(0) - pivot.getCoords(0)) >= minDist:
62             return False
63         return True
64
65     def average(self, other):
66         coordinates = []
67         for i in range(len(self.coords)):
68             coordinates.append((self.coords[i] + other.coords[i]) /
2)
69         return Point(coordinates)
```

3.3 Kelas PointManager

```
1 from Class.Point import Point
2 import matplotlib.pyplot as plt
3 from mpl_toolkits.mplot3d import Axes3D
4 import random
5 import math
6 import ast
7
8 class PointManager:
9     def __init__(self):
10         self.euclideanDistanceCount = 0
11         self.bf_solPointOne = None
12         self.bf_solPointTwo = None
13         self.dnc_solPointOne = None
14         self.dnc_solPointTwo = None
15         self.points = []
16         self.pivot = None
17         self.distance = None
18
19     def readPoints(self, path):
20         self.points = []
21         with open(path, "r") as f:
22             lines = [line.strip("\n") for line in f]
23             for line in lines:
24                 pointz = ast.literal_eval(line[7:])
25                 pointbaru = Point(pointz)
26                 self.points.append(pointbaru)
27
28     def setPoints(self, array):
29         self.points = array
30
31     def getPoints(self):
32         return self.points
33
34     def addPoint(self, newPoint: Point) -> None:
35         self.points.append(newPoint)
36
37     def removePoint(self, point: Point) -> None:
38         self.points.remove(point)
39
40     def listPoints(self) -> None:
41         print("List of Points: ")
42         for p in self.points:
43             print(p.printSelf())
44
45     def getEuclideanDistanceCount(self) -> int:
46         return self.euclideanDistanceCount
47
48     def getDistance(self, pointOne: Point, pointTwo: Point) ->
float:
49         self.euclideanDistanceCount += 1
50         return pointOne.distanceTo(pointTwo)
51
52     def getBFSolPointOne(self) -> Point:
53         return self.bf_solPointOne
```

```

54
55     def getBFSolPointTwo(self) -> Point:
56         return self.bf_solPointTwo
57
58     def getDNCSolPointOne(self) -> Point:
59         return self.dnc_solPointOne
60
61     def getDNCSolPointTwo(self) -> Point:
62         return self.dnc_solPointTwo
63
64     def resetEuclideanCount(self) -> None:
65         self.euclideanDistanceCount = 0
66
67     def bruteForceSolution(self) -> float:
68         if len(self.points) == 1 or len(self.points) == 0:
69             self.distance = float("inf")
70             return
71         shortestDistance = float("inf")
72         for i in range(len(self.points)):
73             for j in range(i + 1, len(self.points)):
74                 distance = self.getDistance(self.points[i], self.
points[j])
75                 if distance < shortestDistance:
76                     self.bf_solPointOne = self.points[i]
77                     self.bf_solPointTwo = self.points[j]
78                     shortestDistance = distance
79                 self.bf_solPointOne.setSolution()
80                 self.bf_solPointTwo.setSolution()
81                 self.distance = shortestDistance
82             return shortestDistance
83
84     def divideAndConquerSolution(self) -> float:
85         if len(self.points) == 1:
86             self.distance = float("inf")
87             return
88         elif len(self.points) == 2:
89             self.distance = self.getDistance(self.points[0], self.
points[1])
90             self.dnc_solPointOne = self.points[0]
91             self.dnc_solPointTwo = self.points[1]
92             return
93         leftPM, rightPM = self.splitPoints()
94         leftPM.divideAndConquerSolution()
95         rightPM.divideAndConquerSolution()
96         self.conquer(leftPM, rightPM)
97         return self.distance
98
99     def conquer(self, leftPM, rightPM):
100         leftDistance = leftPM.distance
101         rightDistance = rightPM.distance
102         if leftDistance < rightDistance:
103             self.distance = leftDistance
104             self.dnc_solPointOne = leftPM.dnc_solPointOne
105             self.dnc_solPointTwo = leftPM.dnc_solPointTwo
106         else:
107             self.distance = rightDistance

```

```

108         self.dnc_solPointOne = rightPM.dnc_solPointOne
109         self.dnc_solPointTwo = rightPM.dnc_solPointTwo
110
111         minDist = min(leftDistance, rightDistance)
112         pivot = self.pivot
113         pointsLeft = leftPM.getDelta(pivot, minDist)
114         pointsRight = rightPM.getDelta(pivot, minDist)
115         distance, sol1, sol2 = self.compare(pointsLeft, pointsRight
, minDist)
116
117         self.euclideanDistanceCount += (
118             leftPM.euclideanDistanceCount + rightPM.
euclideanDistanceCount
119         )
120
121         if distance < minDist:
122             self.distance = distance
123             self.dnc_solPointOne = sol1
124             self.dnc_solPointTwo = sol2
125
126     def compare(self, pointsLeft, pointsRight, minDist):
127         distance = float("inf")
128         sol1 = None
129         sol2 = None
130         for pointL in pointsLeft:
131             for pointR in pointsRight:
132                 if not pointL.scanNear(pointR, minDist):
133                     continue
134                 dist = self.getDistance(pointL, pointR)
135                 if dist < distance:
136                     distance = dist
137                     sol1 = pointL
138                     sol2 = pointR
139             return distance, sol1, sol2
140
141     def getDelta(self, pivot, minDist):
142         points = []
143         for point in self.points:
144             if point.nearPivot(pivot, minDist):
145                 points.append(point)
146         return points
147
148     def splitPoints(self):
149         midPoint = math.floor(len(self.points) / 2)
150         left = PointManager()
151         left.setPoints(self.points[:midPoint])
152         right = PointManager()
153         right.setPoints(self.points[midPoint:])
154         self.pivot = self.points[midPoint - 1].average(self.points[
midPoint])
155         return left, right
156
157     def mergeSort(self, pointArray):
158         # Divide
159         if len(pointArray) > 1:
160             mid = len(pointArray) // 2

```

```

161         left = pointArray[:mid]
162         right = pointArray[mid:]
163
164         # Conquer
165         self.mergeSort(left)
166         self.mergeSort(right)
167
168         i = j = k = 0
169         # Merge
170         # Proses mengisi ulang pointArray
171         while (i < len(left)) and (j < len(right)):
172             if left[i].lessThan(right[j]):
173                 pointArray[k] = left[i]
174                 i += 1
175             else:
176                 pointArray[k] = right[j]
177                 j += 1
178             k += 1
179
180         # Kasus array sisa
181         while i < len(left):
182             pointArray[k] = left[i]
183             i += 1
184             k += 1
185
186         while j < len(right):
187             pointArray[k] = right[j]
188             j += 1
189             k += 1
190
191     def generateRandomPoints(self, n, dim):
192         # TODO: Generate untuk n dimension
193         for i in range(n):
194             points = []
195             for elem in range(dim):
196                 points.append(random.uniform(-1e9, 1e9))
197             self.addPoint(Point(points))
198
199     def plot(self) -> None:
200         dim = self.points[0].getDimension()
201         if self.points[0].getDimension() > 3:
202             print("Sorry, dimension too high to visualize!")
203         if dim == 3:
204             self.plot3D()
205         elif dim == 2:
206             self.plot2D()
207         elif dim == 1:
208             self.plot1D()
209
210     def plot3D(self) -> None:
211         fig = plt.figure()
212         ax = fig.add_subplot(111, projection="3d")
213         for p in self.points:
214             if p.getSolution():
215                 ax.scatter(
216                     [p.getCoords(0)],

```

```

217         [p.getCoords(1)],
218         [p.getCoords(2)],
219         c="red",
220         marker="o",
221         s=90,
222     )
223     else:
224         ax.scatter(
225             [p.getCoords(0)],
226             [p.getCoords(1)],
227             [p.getCoords(2)],
228             c="blue",
229             marker="o",
230             s=90,
231         )
232     ax.set_xlabel("X")
233     ax.set_ylabel("Y")
234     ax.set_zlabel("Z")
235     ax.set_aspect("auto", adjustable="box")
236     plt.show()
237
238 def plot2D(self) -> None:
239     fig = plt.figure()
240     ax = fig.add_subplot(111)
241     for p in self.points:
242         if p.getSolution():
243             ax.scatter(
244                 [p.getCoords(0)],
245                 [p.getCoords(1)],
246                 c="red",
247                 marker="o",
248                 s=90,
249             )
250         else:
251             ax.scatter(
252                 [p.getCoords(0)],
253                 [p.getCoords(1)],
254                 c="blue",
255                 marker="o",
256                 s=90,
257             )
258     ax.set_xlabel("X")
259     ax.set_ylabel("Y")
260     ax.set_aspect("equal", adjustable="box")
261     plt.show()
262
263 def plot1D(self) -> None:
264     fig = plt.figure()
265     ax = fig.add_subplot(111)
266     for p in self.points:
267         if p.getSolution():
268             ax.scatter(
269                 [p.getCoords(0)],
270                 [0],
271                 c="red",
272                 marker="o",

```

```
273         s=90,
274     )
275     else:
276         ax.scatter(
277             [p.getCoords(0)],
278             [0],
279             c="blue",
280             marker="o",
281             s=90,
282         )
283     ax.set_xlabel("X")
284     ax.set_ylabel("Y")
285     plt.show()
```

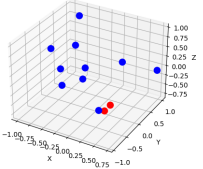
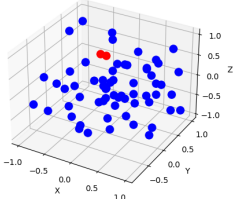
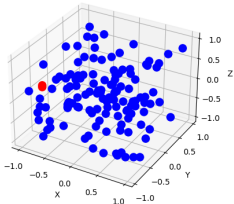
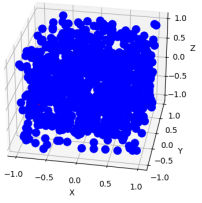
4 Pengujian

Program diuji dengan menggunakan sebuah laptop dengan spesifikasi:

| No | Jenis | Spesifikasi |
|----|--------------|---|
| 1 | Jenis Laptop | Asus Vivobook A409U |
| 2 | RAM | 4GB |
| 3 | Prosesor | Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz 2.30 GHz |

Berikut merupakan hasil pengujian dari program kami. Titik-titik uji merupakan float dengan range $(-1e9, 1e9)$. Hasil pengujian akan dilakukan pada dimensi 3 dan 5.

Hasil uji pada dimensi 3:

| Poin | Jumlah Titik | Output Hasil Uji |
|------|--------------|--|
| 1 | 16 | <pre> Program Plot PS C:\Users\MS0\Desktop\Tuc12_13021123_13021130> python _main_.py Apakah ingin membaca poin dari file? (Y/N) n Insert the number of dimensions (dim) (<= 1): 3 Insert the number of points (n): (<= 2): 16 [Divide and Conquer] Shortest Pair Distance is 2.81291777 of point Point: [601155212.4515319, -420544672.4171362, -918078077.1558712] and Point: [8781449.4768078, 4736870, 111767, -86542879.432121] [Divide and Conquer] Euclidean Distance Calculation Count (Divide and Conquer): 18 [Divide and Conquer] Elapsed time: 1.880997 seconds [Brute Force] Shortest Pair Distance is 2.81291777 of point Point: [601155212.4515319, -420544672.4171362, -918078077.1558712] and Point: [8781449.4768078, 4736870, 111767, -86542879.432121] [Brute Force] Euclidean Distance Calculation Count: 138 [Brute Force] Elapsed time: 8.800889 seconds Do you want to plot the points? Y/N y </pre>  |
| 2 | 64 | <pre> PS C:\Users\MS0\Desktop\Tuc12_13021123_13021130> python _main_.py Apakah ingin membaca poin dari file? (Y if yes, whatever if no) n Insert the number of dimensions (dim): 3 Insert the number of points (n): 64 [Divide and Conquer] Shortest Pair Distance is 111126531.68 of point Point: [155103857.88812054, -421668826.1614212, 576364445.5389572] and Point: [136020248.68821385, -36988361.3827855, 97727734.5817679] [Divide and Conquer] Euclidean Distance Calculation Count (Divide and Conquer): 74 [Divide and Conquer] Elapsed time: 0.883996 seconds [Brute Force] Shortest Pair Distance is 111126531.68 of point Point: [155103857.88812054, -421668826.1614212, 576364445.5389572] and Point: [136020248.68821385, -36988361.3827855, 97727734.5817679] [Brute Force] Euclidean Distance Calculation Count: 2018 [Brute Force] Elapsed time: 8.823488 seconds Do you want to plot the points? Y/N y </pre>  |
| 3 | 128 | <pre> PS C:\Users\MS0\Desktop\Tuc12_13021123_13021130> python _main_.py Apakah ingin membaca poin dari file? (Y if yes, whatever if no) n Insert the number of dimensions (dim): 3 Insert the number of points (n): 128 [Divide and Conquer] Shortest Pair Distance is 41882142.19 of point Point: [-951847051.8287647, -452214171.43888724, 548873969.86818887] and Point: [-945273889.4483576, -454188881.4511717, 96842526.9139545] [Divide and Conquer] Euclidean Distance Calculation Count (Divide and Conquer): 147 [Divide and Conquer] Elapsed time: 0.890884 seconds [Brute Force] Shortest Pair Distance is 41882142.19 of point Point: [-951847051.8287647, -452214171.43888724, 548873969.86818887] and Point: [-945273889.4483576, -454188881.4511717, 96842526.9139545] [Brute Force] Euclidean Distance Calculation Count: 8128 [Brute Force] Elapsed time: 8.122881 seconds Do you want to plot the points? Y/N y </pre>  |
| 4 | 1000 | <pre> PS C:\Users\MS0\Desktop\Tuc12_13021123_13021130> python _main_.py Apakah ingin membaca poin dari file? (Y/N) n Insert the number of dimensions (dim) (<= 1): 3 Insert the number of points (n): (<= 2): 1000 [Divide and Conquer] Shortest Pair Distance is 7561148.18 of point Point: [58623658.93117118, 975418171.1751281, -481387867.11248881] and Point: [59881397.7816176, 97044751.886476, -41118785.462688] [Divide and Conquer] Euclidean Distance Calculation Count (Divide and Conquer): 1226 [Divide and Conquer] Elapsed time: 0.478999 seconds [Brute Force] Shortest Pair Distance is 7561148.18 of point Point: [58623658.93117118, 975418171.1751281, -481387867.11248881] and Point: [59881397.7816176, 97044751.886476, -41118785.462688] [Brute Force] Euclidean Distance Calculation Count: 497408 [Brute Force] Elapsed time: 2.244459 seconds </pre>  |

Hasil uji pada dimensi 5:

| Poin | Jumlah Titik | Output Hasil Uji |
|------|--------------|---|
| 1 | 16 | <pre> PS C:\Users\AGUS\Desktop\fac12_13c21123_13c21135> python _main_.py Apakah ingin membaca poin dari file? (Y if yes, whatever if no) Y Insert the number of dimensions (dim): 5 Insert the number of points (N): 16 ===== [Divide and Conquer] Shortest Pair Distance is 75208052.11 of point Point: [-441361 986.0001138, -5096487.486706, 10647086.17095513, 76751060.833266, -267709360.53 80542] and Point: [-37176608.1388423, -967180639.15472, 139188529.1202136, 57318221.3519457, -93408735.451384] [Divide and Conquer] Euclidean Distance Calculation Count: [Divide and Conquer]: 21 [Divide and Conquer] Elapsed time: 0.863068 seconds ===== [Brute Force] Shortest Pair Distance is 75208052.11 of point Point: [-441361000.8041218, -810954417.4463766, 106647086.17095513, 76751060.833266, -267709360.5380542] and Point: [- 37176608.1388423, -967180639.15472, 139188529.1202136, 57318221.3519457, -93408735.451384] [Brute Force] Euclidean Distance Calculation Count: 128 [Brute Force] Elapsed time: 0.862081 seconds ===== Do you want to plot the points? Y/N Y Sorry, dimension too high to visualize! </pre> |
| 2 | 64 | <pre> PS C:\Users\AGUS\Desktop\fac12_13c21123_13c21135> python _main_.py Apakah ingin membaca poin dari file? (Y if yes, whatever if no) Y Insert the number of dimensions (dim): 5 Insert the number of points (N): 64 ===== [Divide and Conquer] Shortest Pair Distance is 32377867.46 of point Point: [763155832.4483619, 522088345.5923723, 42124769.33182764, 356952186.355541, 984413054.4918363] and Poi nt: [79521232.889645, -40763028.335381, 71113835.1645083, 22898512.69748105, 945777643.6345136] [Divide and Conquer] Euclidean Distance Calculation Count: [Divide and Conquer]: 128 [Divide and Conquer] Elapsed time: 0.461184 seconds ===== [Brute Force] Shortest Pair Distance is 32377867.46 of point Point: [763155832.4483619, 522088345.5923723, 42124769.33182764, 356952186.355541, 984413054.4918363] and Point: [795 21232.889645, -40763028.335381, 71113835.1645083, 22898512.69748105, 945777643.6345136] [Brute Force] Euclidean Distance Calculation Count: 3848 [Brute Force] Elapsed time: 0.855887 seconds ===== Do you want to plot the points? Y/N Y Sorry, dimension too high to visualize! </pre> |
| 3 | 128 | <pre> PS C:\Users\AGUS\Desktop\fac12_13c21123_13c21135> python _main_.py Apakah ingin membaca poin dari file? (Y if yes, whatever if no) Y Insert the number of dimensions (dim): 5 Insert the number of points (N): 128 ===== [Divide and Conquer] Shortest Pair Distance is 35209757.54 of point Point: [666769888.6280224, 669994639.6847719, 557767520.5117111, 159395849.85981274, -337252811.173954] and Poi nt: [79683938.367377, 66981369.248867, 409566902.8887468, 5617488.1549703, -367689434.5778812] [Divide and Conquer] Euclidean Distance Calculation Count: [Divide and Conquer]: 232 [Divide and Conquer] Elapsed time: 0.493193 seconds ===== [Brute Force] Shortest Pair Distance is 35209757.54 of point Point: [666769888.6280224, 669994639.6847719, 557767520.5117111, 159395849.85981274, -337252811.173954] and Point: [796 83938.367377, 66981369.248867, 409566902.8887468, 5617488.1549703, -367689434.5778812] [Brute Force] Euclidean Distance Calculation Count: 8128 [Brute Force] Elapsed time: 0.806607 seconds ===== Do you want to plot the points? Y/N Y Sorry, dimension too high to visualize! PS C:\Users\AGUS\Desktop\fac12_13c21123_13c21135> </pre> |
| 4 | 1000 | <pre> PS C:\Users\AGUS\Desktop\fac12_13c21123_13c21135> python _main_.py Apakah ingin membaca poin dari file? (Y if yes, whatever if no) Y Insert the number of dimensions (dim): 5 Insert the number of points (N): 1000 ===== [Divide and Conquer] Shortest Pair Distance is 388361264.52 of point Point: [255618880.8448948, -576827538.4324881, -313488448.1365844, -515026783.4846707, 788418096.5511956] and P oint: [180452885.590483, -511699728.9226595, -386378212.81844154, -589925544.83881516, 78795205.945745] [Divide and Conquer] Euclidean Distance Calculation Count: [Divide and Conquer]: 1748 [Divide and Conquer] Elapsed time: 0.225558 seconds ===== [Brute Force] Shortest Pair Distance is 388361264.52 of point Point: [255618880.8448948, -576827538.4324881, -313488448.1365844, -515026783.4846707, 788418096.5511956] and Point: [1 80452885.590483, -511699728.9226595, -386378212.81844154, -589925544.83881516, 78795205.945745] [Brute Force] Euclidean Distance Calculation Count: 499508 [Brute Force] Elapsed time: 3.565698 seconds ===== Do you want to plot the points? Y/N Y Sorry, dimension too high to visualize! </pre> |

5 Checklist

| Poin | Judul Fitur | Ya | Tidak |
|------|--|----|-------|
| 1 | Program berhasil dikompilasi tanpa 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 | ✓ | |

6 Daftar Pustaka

- Indyk, Piotr. *Closest Pair*. 2023. URL: <https://people.csail.mit.edu/indyk/6.838-old/handouts/lec17.pdf> (visited on 02/28/2023).
- *Closest Pair Problems in Very High Dimensions*. 2023. URL: https://www.researchgate.net/publication/220897600_Closest_Pair_Problems_in_Very_High_Dimensions (visited on 02/28/2023).
- Munir, Rinaldi. *Algoritma Divide and Conquer (Bagian 1)*. 2023. URL: [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) (visited on 02/28/2023).
- *Algoritma Divide and Conquer (Bagian 2)*. 2023. URL: [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) (visited on 02/28/2023).
- *Algoritma Divide and Conquer (Bagian 3)*. 2023. URL: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-\(2022\)-Bag3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf) (visited on 02/28/2023).
- *Algoritma Divide and Conquer (Bagian 4)*. 2023. URL: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-\(2022\)-Bagian4.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Divide-and-Conquer-(2022)-Bagian4.pdf) (visited on 02/28/2023).
- Nayyar, Varun. *Computational Geometry, Lectures 3,4 Closest Pair Problem*. 2023. URL: <https://www.cse.iitd.ac.in/~ssen/cs852/scribe/scribe2/lec.pdf> (visited on 02/28/2023).

7 Lampiran

7.1 Link Repository GitHub

[Repo GitHub \[Click Me!\]](#) or https://github.com/NicholasLiem/Tucil2_13521123_13521135