Laporan Tugas Kecil 2 IF2211 Strategi Algoritma

Mencari Pasangan Titik Terdekat 3D dengan Algoritma Divide and Conquer

Dibuat oleh:

Margaretha Olivia Haryono	13521071
Michael Utama	13521137

I. Spesifikasi Program

Tautan Repository Github : https://github.com/margarethaolivia/Tucil2_13521071_13521137

Bahasa yang digunakan : Python

Tabel Keberjalanan Program

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

II. Alur Algoritma

1. Alternatif pertama (simple combine)

Algoritma *closest pair* dengan divide and conquer alternatif pertama diimplementasikan pada file dnc.py. Berikut adalah langkah-langkah algoritma *closest pair*.

a. Solve: Jika n = 2, jarak titik dihitung langsung dengan rumus jarak euclidean

b. Divide: Urutkan titik berdasarkan proyeksinya pada e_1. Bagi kumpulan titik ke dalam dua bagian berdasarkan hasil pengurutan.

   Setelah langkah ini, terdapat garis atau bidang maya L yang membagi ruang menjadi dua partisi sehingga seluruh titik pada masing-masing bagian menempati partisi yang sama dan kedua bagian menempati partisi yang berbeda.

c. Conquer: secara rekursif, terapkan algoritma divide and conquer pada masing-masing bagian untuk mencari titik terdekat.

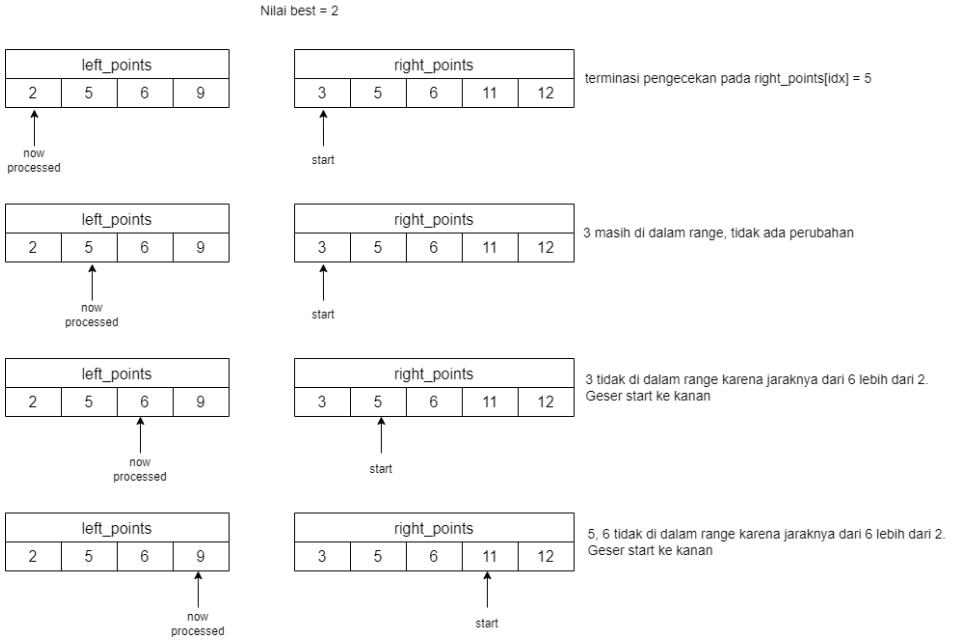
d. Combine:

i. Cari jarak terdekat dari pemanggilan conquer, beri nama *best*.

ii. Kumpulkan seluruh titik dengan jarak *best* dari bidang L. Pisahkan berdasar hasil pembagian pada langkah Divide. Beri nama kumpulan titik *left_points* dan *right_points*.

iii. Urutkan titik berdasarkan besar proyeksinya pada e_2.

iv. Ukur jarak dari seluruh titik pada *left_points* ke seluruh titik pada *right_points* yang memiliki jarak pada e_2 kurang dari *best*. Untuk menghindari iterasi seluruh *right_points* berulang kali, catat dan geser posisi awal yang harus diperiksa pada *right_points* seperti pada gambar di bawah.



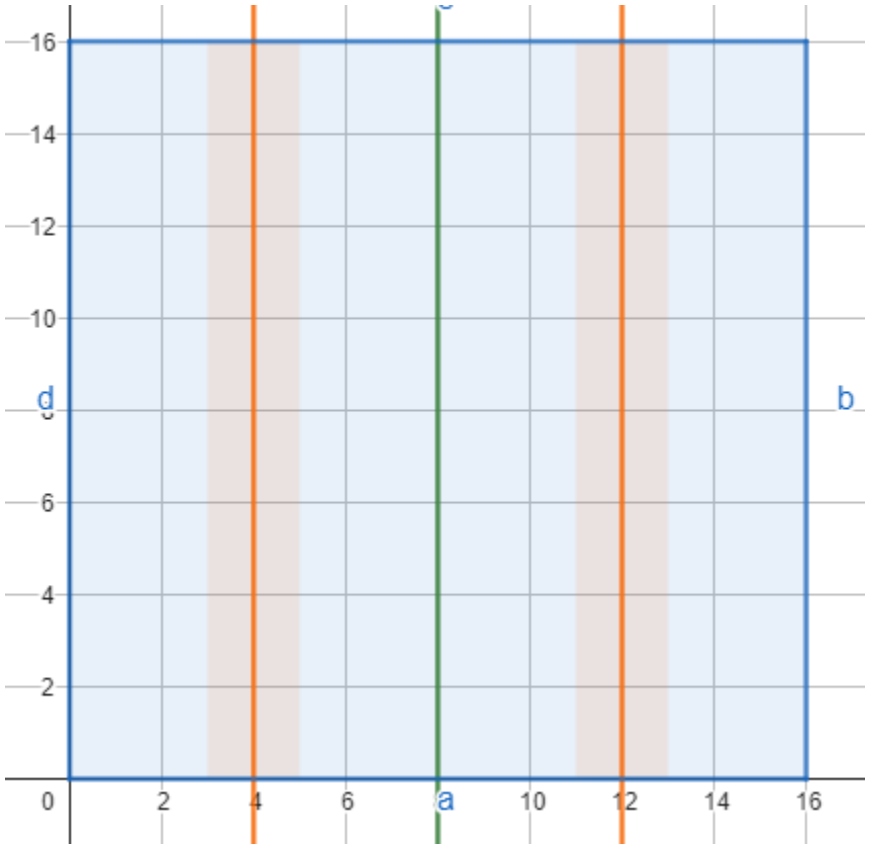
Gambar pencatatan posisi awal dengan *best* = 2

Dengan cara di atas, program hanya perlu melakukan iterasi sebanyak jumlah pasangan titik berjarak kurang dari *best* ditambah ukuran right_points (untuk menggeser start) dan ukuran left_points.

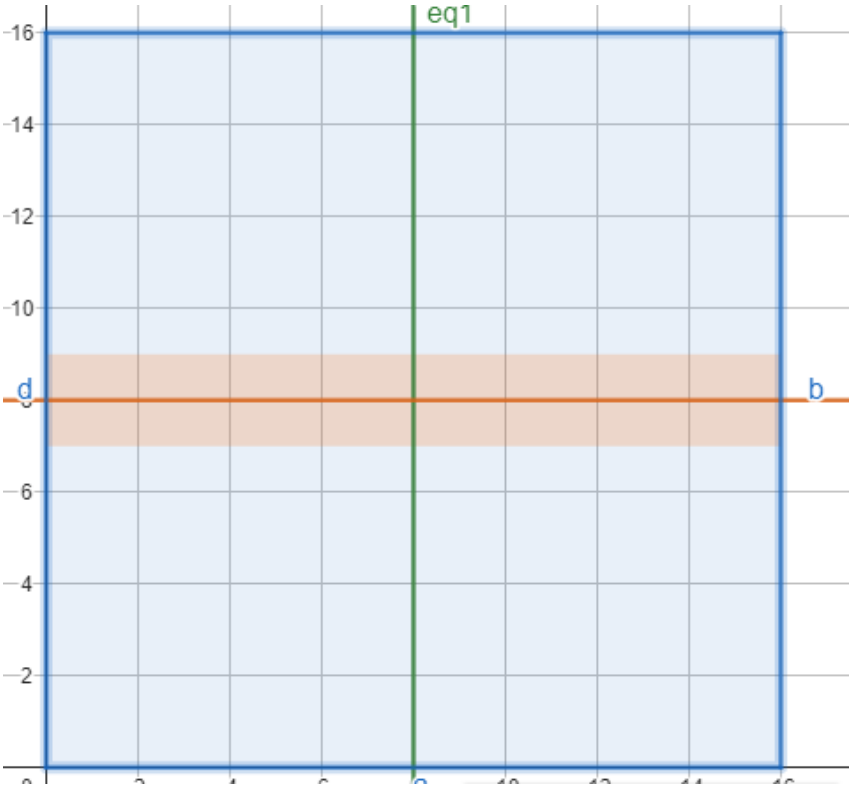
v. Update nilai *best* jika jarak pada langkah (iv) kurang dari *best*.

Dengan asumsi titik tersebar *uniformly at random*, terdapat optimasi yang dapat dilakukan untuk mengurangi jumlah pemanggilan fungsi distance, sekaligus mengurangi waktu jalannya program. Penggunaan dimensi yang berbeda pada langkah divide dapat

meningkatkan rata-rata panjang *range* titik yang dibagi, sehingga rasio $\frac{best}{range}$ semakin kecil. Mengecilnya rasio tersebut berarti jumlah titik *random* yang berada di daerah *combine* berkurang.



Gambar a menunjukkan contoh pembagian dengan dimensi sama



Gambar b menunjukkan contoh pembagian dengan dimensi berbeda

Daerah yang diberi warna jingga menandakan area dengan jarak tertentu terhadap garis pembagi kedua. Pada gambar a, luas area berwarna jingga lebih besar daripada gambar b, sehingga jumlah pasangan titik yang harus dihitung pada proses combine semakin besar.

Algoritma *closest pair* alternatif pertama ini memiliki kelemahan pada kasus tertentu, yaitu ketika banyak titik dengan jarak sama terletak pada e_1 dan e_2 yang sama, tetapi pada e_3 yang berbeda. Kompleksitas jika hanya dihitung dari banyak pemanggilan fungsi *distance* pada fungsi rekursi pertama adalah

$$T(n) = (n/2) \times (n/2)$$

$$T(n) = O(n^2)$$

Untuk mengatasi masalah tersebut, diperlukan teknik tambahan untuk melakukan kombinasi yang akan dibahas pada alternatif berikutnya.

2. Alternatif kedua (hard combine)

Algoritma *closest pair* untuk alternatif kedua tidak diimplementasikan. Hal ini disebabkan kesulitan dalam implementasi BBST (balanced binary search tree) pada bahasa python tanpa menggunakan library khusus, tidak seperti std::set atau std::map pada c++. Algoritma *closest pair* alternatif kedua pada R^3 hampir sama seperti pada alternatif pertama, hanya berbeda pada tahap combine, dengan langkah

a. Cari jarak terdekat dari pemanggilan conquer, beri nama *best*.
b. Kumpulkan seluruh titik dengan jarak *best* dari bidang L. Pisahkan berdasar hasil pembagian pada langkah Divide. Beri nama kumpulan titik *left_points* dan *right_points*.
c. Urutkan titik berdasarkan besar proyeksinya pada e_2.
d. Iterasi seluruh elemen *left_points*.
    i. Masukkan seluruh titik pada *right_points* yang jaraknya pada e_2 kurang dari *best* ke sebuah struktur data BBST(terurut berdasarkan e_3). Aturan memasukkan dan mengeluarkan titik dari BBST mirip dengan penggeseran iterator pada alternatif pertama.
    ii. Lakukan binary search untuk memperoleh titik-titik yang jaraknya pada e_3 kurang dari *best*.

Dapat dijamin pada tahap combine pada alternatif kedua, setiap titik hanya dipasangkan dengan kurang dari 27 titik lainnya. Dengan demikian, proses combine hanya membutuhkan waktu O(n log n). Dengan teorema master dan asumsi log n dapat dikeluarkan dari persamaan, diperoleh kompleksitas waktu akhir adalah $O(n \log^2(n))$.

Pada ruang R^d dengan $d > 3$, dapat digunakan struktur data lain yang dapat memasukkan dan mengeluarkan titik dengan waktu O(log n) atau $O(\log^d n)$ seperti k-d tree atau segment tree. Hal tersebut tidak dibahas pada laporan tugas kecil ini.

III. Kode Program

1. *file* bf.py

```python
import math


# Brute Force algorithm (calculate the distance for each pair of points)
def bruteForce(n, r, points):
    closest = 999 ** 9
    count = 0
    for i in range(n):
        for j in range(i+1, n):
            distance = 0
            for k in range(r):
                distance += (points[i][k] - points[j][k]) ** 2
            distance = math.sqrt(distance)
            count += 1
            if distance < closest:
                closest = distance
                p1 = i
                p2 = j
    return closest, p1, p2, count
```

2. *file* dnc.py

```python
import math
import sort

def euclideanDistance(p1, p2):
    """ Return euclidean distance of two points. \n
    len(p1) = len(p2)

    Args:
        p1 (List of double): first point
        p2 (List of double): second point

    Returns:
        double: euclidean distance

    """
    global DISTANCE_FUNCTION_CALLED
    DISTANCE_FUNCTION_CALLED += 1
    total = 0.0
    for i in range(len(p1)):
        total += (p1[i] - p2[i]) ** 2
    return math.sqrt(total)
```

```python
def DnC(points, dimension):
    """ Return closest pair from points using divide and conquer

    Args:
        points (List of (int, List of double)): List of points indices and location

    Returns:
        double: closest pair distance
        int: index of first point in closest pair
        int: index of second point in closest pair

    """
    if len(points) == 1:
        return None, None, None
    if len(points) == 2:
        points = sort.merge_sort(points, lambda x:x[1][dimension])
        return euclideanDistance(points[0][1], points[1][1]), points[0][0], points[1][0]

    # Divide points
    # points.sort(key=lambda x: x[1][dimension])
    points = sort.merge_sort(points, lambda x:x[1][dimension])
    mid = len(points)//2
    l_points = points[:mid]
    r_points = points[mid:]
    midpoint = l_points[-1][1][dimension]

    # Conquer each part
    nextdimension = (dimension + 1) % len(points[0][1])
    l_best, l_index1, l_index2 = DnC(l_points, nextdimension)
    r_best, r_index1, r_index2 = DnC(r_points, nextdimension)
```

```python
    # Merge both part
    if l_best != None and (r_best == None or l_best < r_best):
        best, index1, index2 = l_best, l_index1, l_index2
    else:
        best, index1, index2 = r_best, r_index1, r_index2
    l_points = sort.merge_sort(l_points, lambda x:x[1][nextdimension])
    r_points = sort.merge_sort(r_points, lambda x:x[1][nextdimension])

    l_mid = []
    for i in range(len(l_points)):
        if midpoint - l_points[i][1][dimension] <= best:
            l_mid.append(l_points[i])

    r_mid = []
    for i in range(len(r_points)):
        if r_points[i][1][dimension] - midpoint <= best:
            r_mid.append(r_points[i])

    r_start = 0
    for x in l_mid:
        while r_start + 1 < len(r_mid) and x[1][nextdimension] - r_mid[r_start][1][nextdimension] > best:
            r_start += 1
        for j in range(r_start, len(r_mid)):
            if r_mid[j][1][nextdimension] - x[1][nextdimension] > best:
                break
            dist = euclideanDistance(x[1], r_mid[j][1])
            if dist < best:
                best, index1, index2 = dist, x[0], r_mid[j][0]


    return best, index1, index2
```

```
def getClosestPair(points):
    """ Return closest pair, distance and indices

    Args:
        points (List of (int, List of double)): List of points indices and location

    Returns:
        double: closest pair distance
        int: index of first point in closest pair
        int: index of second point in closest pair
        int: number of distance function called

    """
    global DISTANCE_FUNCTION_CALLED
    DISTANCE_FUNCTION_CALLED = 0
    closestPair = DnC(points.copy(), 0)
    return closestPair[0], closestPair[1], closestPair[2], DISTANCE_FUNCTION_CALLED
```

3. ***file*** **sort.py**

```
import random
import time

def random_array(size):
    return [round(random.uniform(-1000, 1000), 2) for i in range(size)]

def selection_sort(arr, key):
    for i in range(len(arr)):
        minidx = i
        for j in range(i + 1, len(arr), 1):
            if key(arr[j]) < key(arr[minidx]):
                minidx = j
        tmp = arr[i]
        arr[i] = arr[minidx]
        arr[minidx] = tmp
    return arr
```

```
def merge(larr, rarr, key):
    arr = [0 for i in range(len(larr) + len(rarr))]
    lidx = 0
    ridx = 0
    for i in range(len(arr)):
        if lidx == len(larr):
            arr[i] = rarr[ridx]
            ridx += 1
        elif ridx == len(rarr):
            arr[i] = larr[lidx]
            lidx += 1
        elif key(larr[lidx]) < key(rarr[ridx]):
            arr[i] = larr[lidx]
            lidx += 1
        else:
            arr[i] = rarr[ridx]
            ridx += 1
    return arr

def merge_sort(arr, key = lambda x:x):
    """ return arr, sorted using merge sort

    """
    if len(arr) <= 43:
        return selection_sort(arr, key)
    mid = len(arr) // 2
    larr = merge_sort(arr[:mid], key)
    rarr = merge_sort(arr[mid:], key)

    return merge(larr, rarr, key)
```

4. ***file*** **main.py**

```
import random
import time
import matplotlib.pyplot as plt

import dnc
import bf


# generate random points and add it into a list
def generatePoints(n, r):
    points = []
    p = []
    for i in range(n):
        point = []
        for j in range(r):
            x = round(random.uniform(-10**3, 10**3), 3)
            point.append(x)
        t = (i, point)
        points.append(point)
        p.append(t)
    return points, p
```

```python
# print a point
def printPoint(point):
    print('(', end='')
    for i in range(len(point)-1):
        print(point[i], end=', ')
    print(point[r-1], end=')\n')


# display gui for 3D points visualization
def printVisualization(points, p1, p2):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    for p in points:
        x = p[0]
        y = p[1]
        z = p[2]
        ax.scatter(x, y, z, color='black')
    ax.scatter(points[p1][0], points[p1][1], points[p1][2], color='red')
    ax.scatter(points[p2][0], points[p2][1], points[p2][2], color='red')
    plt.show()
```

```python
print("        PENCARIAN PASANGAN TITIK TERDEKAT          ")
print("===================================================")
n = int(input("Masukkan banyaknya titik : "))
# n input validation
while (n < 2):
    print('Masukan tidak valid!')
    n = int(input("Masukkan banyaknya titik  : "))

r = int(input("Masukkan dimensi vektor  : "))
# r input validation
while (r <= 0):
    print('Masukan tidak valid!')
    r = int(input("Masukkan dimensi vektor  : "))

print()

# generate n points
points, p = generatePoints(n, r)
print(f'MEMBANGKITKAN {n} TITIK ACAK')
for point in points:
    printPoint(point)

print()
print('Memproses dengan algoritma Brute Force...')
print()

# Brute Force algorithm
startTime = time.time() * 1000
closest, p1, p2, count = bf.bruteForce(n, r, points)
endTime = time.time() * 1000
```

```
# Brute Force statistics
print('ALGORITMA BRUTEFORCE')
print(f'Titik 1 : ', end='')
printPoint(points[p1])
print(f'Titik 2 : ', end='')
printPoint(points[p2])
print(f'Jarak   : {round(closest, 3)}')
print(f'Banyak Perhitungan : {count}')
print(f'Waktu Eksekusi     : {endTime - startTime} milisekon')

print()
print('Memproses dengan algoritma Divide and Conquer...')
print()

# DnC algorithm
startTime = time.time() * 1000
closest, p1, p2, count = dnc.getClosestPair(p)
endTime = time.time() * 1000

# DnC statistics
print('ALGORITMA DIVIDE AND CONQUER')
print(f'Titik 1 : ', end='')
printPoint(points[p1])
print(f'Titik 2 : ', end='')
printPoint(points[p2])
print(f'Jarak   : {round(closest, 3)}')
print(f'Banyak Perhitungan : {count}')
print(f'Waktu Eksekusi     : {endTime - startTime} milisekon')
```

```
# ask for visualization (3D only)
if r == 3:
    print()
    show = input('Ingin menampilkan visualisasi? [Y/n] : ').lower()
    while (show != 'y' and show != 'n'):
        print('Masukan tidak valid!')
        show = input('Ingin menampilkan visualisasi? [Y/n] : ').lower()

    if (show == 'y'):
        print('Menampilkan visualisasi...')
        printVisualization(points, p1, p2)

print('================================================')
print('    TERIMA KASIH TELAH MENGGUNAKAN PROGRAM KAMI    ')
```

IV. *Screenshot* Uji Kasus

Tangkapan layar pada bagian ini dilakukan pada laptop dengan spesifikasi sebagai berikut.

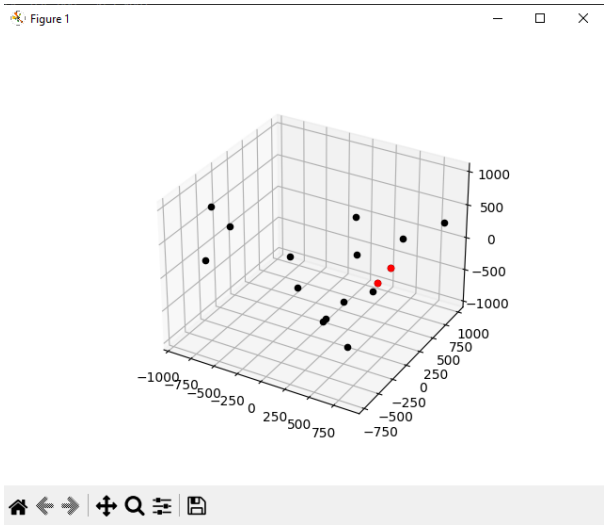
CPU : Intel Core i5-5200U

RAM : 8 GB 1600 MHz

Sistem Operasi : Windows 10

1. n = 16, dimensi 3

```
        PENCARIAN PASANGAN TITIK TERDEKAT
================================================
Masukkan banyaknya titik : 16
Masukkan dimensi vektor  : 3

MEMBANGKITKAN 16 TITIK ACAK
(566.097, 140.039, -268.71)
(344.128, 865.798, -569.165)
(407.451, 975.809, -162.488)
(-916.456, 74.584, 156.843)
(493.91, -182.922, -897.0)
(293.897, 716.968, -722.278)
(765.972, -712.507, 347.504)
(-666.631, -678.153, 318.491)
(439.456, 59.094, 290.999)
(-470.999, 518.358, -957.29)
(-436.517, 343.42, -317.276)
(-696.585, -524.234, 981.519)
(416.284, 78.769, 820.635)
(195.992, -59.867, -687.948)
(876.009, 890.938, 332.276)
(550.886, -688.951, -62.174)
```

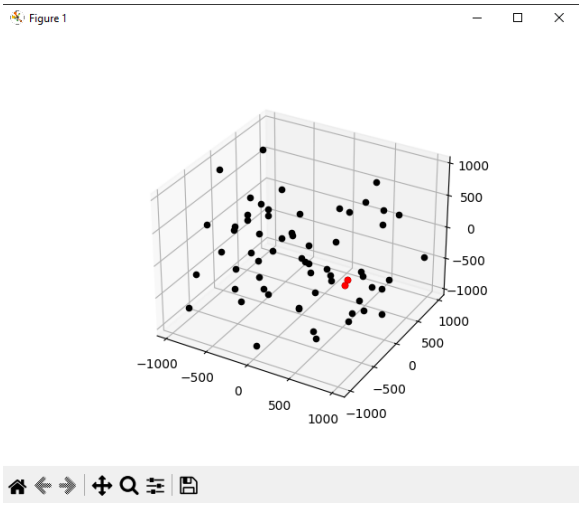



```
Memproses dengan algoritma Brute Force...

ALGORITMA BRUTEFORCE
Titik 1 : (344.128, 865.798, -569.165)
Titik 2 : (293.897, 716.968, -722.278)
Jarak   : 219.356
Banyak Perhitungan : 120
Waktu Eksekusi     : 2.0009765625 milisekon

Memproses dengan algoritma Divide and Conquer...

ALGORITMA DIVIDE AND CONQUER
Titik 1 : (293.897, 716.968, -722.278)
Titik 2 : (344.128, 865.798, -569.165)
Jarak   : 219.356
Banyak Perhitungan : 21
Waktu Eksekusi     : 1.00244140625 milisekon

Ingin menampilkan visualisasi? [Y/n] : y
Menampilkan visualisasi...
================================================
    TERIMA KASIH TELAH MENGGUNAKAN PROGRAM KAMI
```

2. n = 64, dimensi 3

```
        PENCARIAN PASANGAN TITIK TERDEKAT
================================================
Masukkan banyaknya titik : 64
Masukkan dimensi vektor  : 3

MEMBANGKITKAN 64 TITIK ACAK
(280.261, -340.039, 66.119)
(6.818, 265.213, -910.682)
(371.054, -710.875, -343.453)
(-291.666, -870.469, -196.314)
(631.662, 983.621, 107.46)
(891.466, -379.92, -241.914)
(689.334, 100.398, -824.516)
(-331.891, -534.474, 626.908)
(635.582, -419.658, 578.042)
(37.599, -764.181, -229.975)
(-377.72, 803.908, -646.312)
(146.203, -447.352, 504.115)
(-383.437, -41.43, 468.988)
(-435.502, 706.638, -96.258)
(-612.906, -27.931, 555.519)
(-113.736, 260.679, -463.646)
(56.401, 543.066, -897.11)
(622.327, 209.729, -367.508)
(-916.122, -142.45, 954.91)
(313.702, -617.061, -440.112)
(998.75, 884.821, -372.61)
(-796.562, 627.531, -202.577)
(750.128, -5.775, 951.897)
(464.047, 758.17, 696.911)
(-830.861, 455.468, -841.162)
(-361.653, -202.686, -688.282)
(896.771, 131.54, -819.03)
(-633.431, -314.649, 279.869)
(730.874, 413.869, 334.794)
(349.194, -362.508, -960.576)
(-924.535, 744.589, 741.68)
(920.238, -150.92, -182.15)
(996.187, -798.3, -224.685)
(816.229, -379.469, -469.733)
(-593.198, 86.914, -777.926)
(-964.951, -341.335, 200.064)
(-750.624, 409.84, 141.588)
(688.226, -693.58, 368.61)
(-128.154, -187.903, 940.213)
(-324.457, -558.898, 719.04)
(-300.299, -858.121, 688.707)
(821.59, -206.722, 52.773)
(-767.657, -385.079, -120.104)
(-74.34, 315.959, -659.433)
(-695.677, -971.095, -61.933)
(546.205, -374.185, 4.387)
(844.142, 211.916, -489.549)
(648.894, 756.094, -793.829)
(-970.304, -721.269, -877.012)
(29.835, -980.635, -877.206)
(598.899, -80.276, 801.132)
(701.654, -928.283, -525.099)
(-465.783, 794.167, -886.531)
(241.772, 569.544, -830.365)
(544.509, -181.029, 894.568)
(-189.908, -921.231, -313.214)
(-655.087, 24.995, -364.405)
(-339.304, -771.496, 30.006)
(244.403, 504.411, -867.468)
(46.378, -499.359, 461.323)
(-970.239, 715.09, -619.451)
(691.873, 507.122, 486.463)
(-705.581, 572.064, -713.8)
(-537.824, 700.322, -437.099)
```

```
Memproses dengan algoritma Brute Force...

ALGORITMA BRUTEFORCE
Titik 1 : (241.772, 569.544, -830.365)
Titik 2 : (244.403, 504.411, -867.468)
Jarak   : 75.006
Banyak Perhitungan : 2016
Waktu Eksekusi     : 16.000244140625 milisekon

Memproses dengan algoritma Divide and Conquer...

ALGORITMA DIVIDE AND CONQUER
Titik 1 : (244.403, 504.411, -867.468)
Titik 2 : (241.772, 569.544, -830.365)
Jarak   : 75.006
Banyak Perhitungan : 74
Waktu Eksekusi     : 7.968505859375 milisekon

Ingin menampilkan visualisasi? [Y/n] : y
Menampilkan visualisasi...
================================================
    TERIMA KASIH TELAH MENGGUNAKAN PROGRAM KAMI
```



3. n = 128, dimensi 3

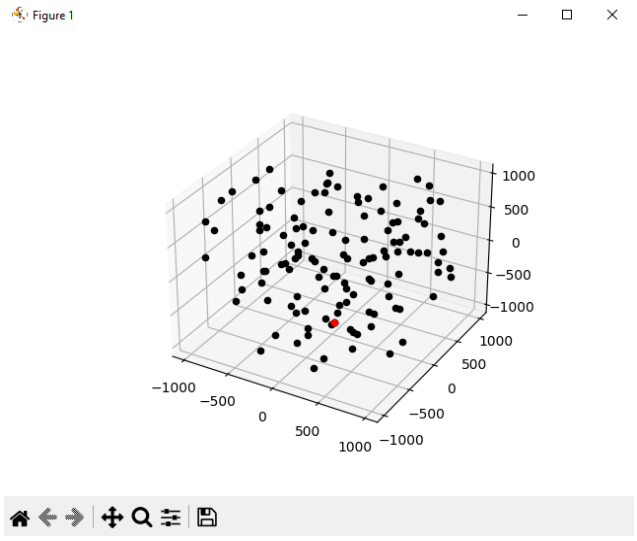
```
        PENCARIAN PASANGAN TITIK TERDEKAT
================================================
Masukkan banyaknya titik : 128
Masukkan dimensi vektor  : 3

MEMBANGKITKAN 128 TITIK ACAK
(-141.321, -428.531, -296.604)
(349.652, -566.71, -387.757)
(-922.904, 160.68, 107.058)
(-280.489, -439.897, 564.086)
(635.144, -634.713, -391.261)
(-825.631, 209.363, 937.796)
(-840.222, 759.306, -726.761)
(-698.849, -546.782, -331.668)
(-551.827, 863.239, 438.425)
(622.408, -826.991, 198.295)
(-683.739, 194.257, 684.267)
(274.601, -487.823, -509.113)
(-216.503, -274.637, 243.452)
(-88.189, 424.752, -742.817)
(-74.883, -205.877, 109.917)
(79.797, 550.754, 617.557)
(-251.164, -897.992, 940.583)
(95.437, 948.089, -102.419)
(-281.704, 711.942, -474.129)
(604.133, 823.749, 826.895)
(982.918, 364.129, 509.317)
(134.016, -730.552, 791.962)
(17.192, 980.687, 503.525)
(511.539, 214.716, -483.45)
(-654.705, -173.158, -283.108)
(657.752, 532.774, 24.064)
(-256.075, -787.17, -150.769)
(353.459, -506.27, -290.285)
(-365.194, -183.913, -702.041)
(620.654, 716.899, 327.418)
```

```
(-965.738, 174.877, 754.985)
(861.055, 564.355, -203.06)
(184.267, 705.927, -239.81)
(604.472, 986.129, -975.64)
(-54.371, -589.881, -390.07)
(679.441, 54.668, 803.021)
(-792.999, -983.751, 925.796)
(-367.506, -94.439, 569.305)
(-79.503, 623.432, 548.607)
(949.349, -882.777, 211.314)
(-728.965, -92.733, -77.309)
(471.114, 843.952, 863.685)
(774.146, 838.217, -111.861)
(776.277, 729.629, -208.481)
(-330.062, -916.916, 565.47)
(-372.629, -459.588, -956.158)
(-989.436, -545.956, 441.362)
(373.676, -512.186, -153.331)
(603.539, 96.289, 738.314)
(-194.807, 219.921, 944.264)
(58.8, -227.131, -696.609)
(-973.349, 286.993, -718.107)
(-320.743, -354.968, 76.423)
(268.328, 128.481, -4.199)
(748.777, 75.836, -980.232)
(232.878, -5.28, -405.588)
(14.449, -694.164, -751.075)
(985.492, 79.145, 451.1)
(-766.059, 690.707, 189.865)
(-542.094, 89.371, -392.499)
(-704.241, -68.153, 286.153)
(594.516, 646.31, 436.738)
(154.775, -409.948, -77.611)
(473.75, -225.221, 657.138)
(559.356, 332.906, 277.661)
(193.041, -685.179, 969.445)
```

```
(32.467, 242.727, -112.925)
(984.292, -626.115, -545.095)
(-953.749, -279.7, 863.946)
(-506.847, 176.767, 190.316)
(117.483, -659.729, -606.668)
(55.211, 398.326, 640.268)
(776.184, 722.263, 719.026)
(264.128, -225.996, -65.231)
(277.94, 581.955, -209.639)
(-956.357, -767.902, 196.819)
(-658.46, 788.896, 298.282)
(594.851, 39.821, -127.166)
(-19.86, -38.595, 743.789)
(856.649, -772.504, -85.267)
(-415.114, -630.967, 986.161)
(864.906, -37.363, 489.723)
(803.629, -722.12, 57.612)
(-964.536, -137.346, -494.422)
(561.306, 740.099, 490.577)
(-730.826, 664.387, -175.107)
(141.128, -221.552, -21.459)
(361.888, -579.206, -381.679)
(988.378, 561.866, -95.328)
(-320.187, -821.823, -917.076)
(-452.913, 887.51, 404.741)
(-484.486, 696.673, 422.583)
(-199.744, 357.842, 124.52)
(530.254, -149.029, 363.728)
(-158.306, 350.531, -524.104)
(58.794, -559.306, -604.467)
(-162.648, -676.143, 342.558)
(-903.995, -328.937, -739.994)
(-307.258, -805.237, 104.246)
(837.347, -170.76, -261.248)
(-301.137, 352.216, -464.274)
```

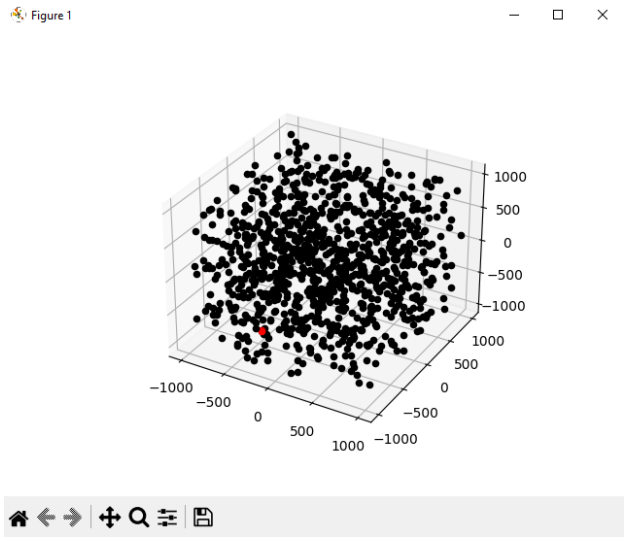
```
(495.996, -467.248, -543.427)
(283.905, -134.36, -832.78)
(31.26, 923.287, 186.462)
(-187.479, -376.546, 214.988)
(84.854, -670.567, -254.855)
(-39.363, 697.447, 216.649)
(923.033, -411.353, -54.109)
(782.349, 26.699, 631.214)
(354.01, -967.962, -779.412)
(-468.278, 12.786, 58.95)
(359.802, 670.002, -154.555)
(521.873, -170.769, 25.134)
(477.873, 400.918, 775.465)
(484.375, -143.821, 14.722)
(-174.32, 28.386, -315.949)
(313.349, -715.669, -837.012)
(829.655, 405.194, 942.406)
(-979.173, -425.577, 817.81)
(-523.129, 491.957, -484.339)
(-965.183, 422.442, 191.251)
(-278.157, 431.549, 933.989)
(197.878, 995.758, -777.302)
(533.442, -554.249, -719.091)
(15.643, 221.865, 184.595)
(796.055, 978.2, -586.838)
(101.588, 557.068, -291.482)
(371.321, 556.474, 72.501)
```

```
Memproses dengan algoritma Brute Force...

ALGORITMA BRUTEFORCE
Titik 1 : (349.652, -566.71, -387.757)
Titik 2 : (361.888, -579.206, -381.679)
Jarak   : 18.515
Banyak Perhitungan : 8128
Waktu Eksekusi     : 37.995361328125 milisekon

Memproses dengan algoritma Divide and Conquer...

ALGORITMA DIVIDE AND CONQUER
Titik 1 : (349.652, -566.71, -387.757)
Titik 2 : (361.888, -579.206, -381.679)
Jarak   : 18.515
Banyak Perhitungan : 172
Waktu Eksekusi     : 16.9931640625 milisekon

Ingin menampilkan visualisasi? [Y/n] : y
Menampilkan visualisasi...
================================================
    TERIMA KASIH TELAH MENGGUNAKAN PROGRAM KAMI
```



4. n = 1000, dimensi 3

```
        PENCARIAN PASANGAN TITIK TERDEKAT
================================================
Masukkan banyaknya titik : 1000
Masukkan dimensi vektor  : 3

MEMBANGKITKAN 1000 TITIK ACAK
(97.754, -235.05, 17.221)
(969.544, -497.662, 962.809)
(57.769, -866.583, 117.136)
(-237.375, -960.055, 434.765)
(850.69, 584.33, -422.824)
(-117.119, -547.169, -15.819)
(837.666, -885.627, 94.489)
(912.504, 568.919, 634.929)
(-956.367, 759.456, 382.431)
(-366.891, 807.052, 238.139)
(-481.987, 931.259, 816.292)
(-456.657, -556.866, -601.759)
(-15.636, 164.295, 995.795)
(827.741, -755.477, -37.185)
(664.337, -71.028, -85.155)
(319.811, 497.904, 795.181)
(-296.181, -114.759, 757.55)
(-461.13, -238.849, 530.788)
(-771.086, -680.486, 418.111)
(-225.19, -479.103, -187.188)
(129.405, 630.841, -872.642)
(-434.075, 530.913, -89.101)
(-137.124, 568.801, -973.646)
(252.707, -11.748, 939.62)
(520.825, 226.451, 438.232)
(-81.989, 537.33, 446.079)
(876.601, 814.638, -126.593)
(-38.181, 496.935, 937.226)
(-683.125, -430.362, 569.161)
(720.753, 913.61, -630.929)
(-55.827, 910.06, 780.923)
(358.507, 523.792, 496.782)
(599.812, -831.652, -157.679)
(-652.585, 834.004, -8.806)
(309.997, -46.539, -920.042)
(198.413, -915.988, -976.122)
(-413.854, -516.233, 320.43)

Memproses dengan algoritma Brute Force...

ALGORITMA BRUTEFORCE
Titik 1 : (-357.297, -668.251, -754.303)
Titik 2 : (-369.777, -652.112, -757.728)
Jarak   : 20.687
Banyak Perhitungan : 499500
Waktu Eksekusi     : 1236.549560546875 milisekon

Memproses dengan algoritma Divide and Conquer...

ALGORITMA DIVIDE AND CONQUER
Titik 1 : (-369.777, -652.112, -757.728)
Titik 2 : (-357.297, -668.251, -754.303)
Jarak   : 20.687
Banyak Perhitungan : 1269
Waktu Eksekusi     : 187.993896484375 milisekon

Ingin menampilkan visualisasi? [Y/n] : y
Menampilkan visualisasi...
================================================
    TERIMA KASIH TELAH MENGGUNAKAN PROGRAM KAMI
```



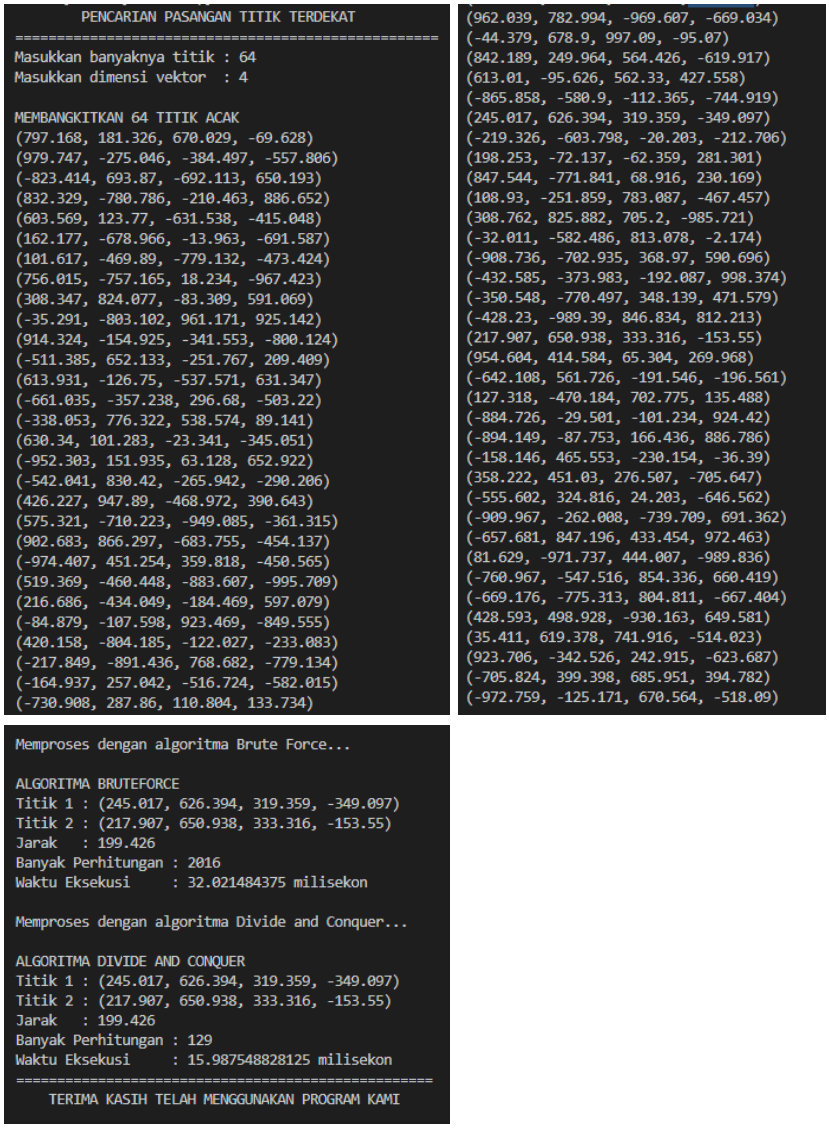
5. n = 64, dimensi 2

```
        PENCARIAN PASANGAN TITIK TERDEKAT
================================================
Masukkan banyaknya titik : 64
Masukkan dimensi vektor  : 2

MEMBANGKITKAN 64 TITIK ACAK
(9.328, 103.245)
(-271.582, 742.591)
(511.373, -464.13)
(694.062, -684.793)
(-143.661, -761.482)
(-709.314, -93.103)
(-509.801, 3.49)
(-354.849, -560.914)
(-518.962, -39.218)
(-109.224, 489.256)
(716.609, 142.209)
(-97.387, -155.825)
(-874.204, 965.293)
(401.623, 938.729)
(322.881, -645.621)
(486.561, -862.095)
(905.151, 343.298)
(624.283, -187.047)
(-617.219, -530.987)
(-264.888, -277.882)
(887.498, 985.064)
(800.464, -323.558)
(-595.03, 128.116)
(975.625, 133.251)
(295.254, -309.346)
(-409.295, 402.782)
(165.622, -879.232)
(-895.885, 443.985)
(-462.337, 805.519)
(-823.181, -681.856)
(-737.868, 258.377)
```

```
(641.813, 218.759)
(-192.32, -784.334)
(-720.104, -283.988)
(159.128, 951.178)
(346.547, 227.349)
(657.342, -872.172)
(461.102, -121.8)
(523.311, -683.236)
(750.751, -995.882)
(-875.201, -165.67)
(-119.138, -824.106)
(788.577, -630.002)
(963.715, -927.345)
(505.839, 108.367)
(368.896, 170.982)
(644.145, 527.349)
(-333.14, -547.125)
(750.478, -343.071)
(926.205, -608.819)
(-381.992, 274.435)
(-830.495, -131.947)
(-104.887, 391.366)
(351.322, 61.679)
(253.573, 656.26)
(-448.011, -440.25)
(-230.193, 188.813)
(965.581, -621.215)
(379.245, -612.039)
(-729.552, 449.919)
(933.539, 849.153)
(12.686, -114.523)
(-733.392, 914.561)
(-618.492, 667.688)
```

```
Memproses dengan algoritma Brute Force...

ALGORITMA BRUTEFORCE
Titik 1 : (-354.849, -560.914)
Titik 2 : (-333.14, -547.125)
Jarak   : 25.718
Banyak Perhitungan : 2016
Waktu Eksekusi     : 16.001953125 milisekon

Memproses dengan algoritma Divide and Conquer...

ALGORITMA DIVIDE AND CONQUER
Titik 1 : (-354.849, -560.914)
Titik 2 : (-333.14, -547.125)
Jarak   : 25.718
Banyak Perhitungan : 60
Waktu Eksekusi     : 7.996337890625 milisekon
================================================
    TERIMA KASIH TELAH MENGGUNAKAN PROGRAM KAMI
```

6. n = 64, dimensi 4

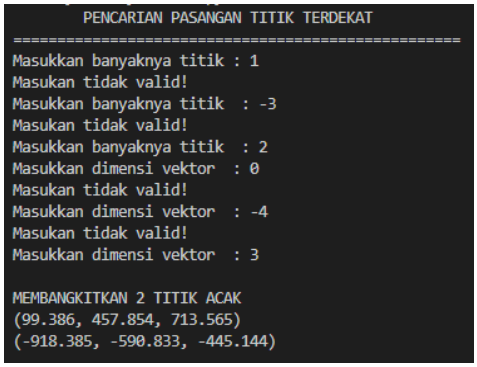


```
        PENCARIAN PASANGAN TITIK TERDEKAT
==================================================
Masukkan banyaknya titik : 64
Masukkan dimensi vektor  : 4

MEMBANGKITKAN 64 TITIK ACAK
(797.168, 181.326, 670.029, -69.628)
(979.747, -275.046, -384.497, -557.806)
(-823.414, 693.87, -692.113, 650.193)
(832.329, -780.786, -210.463, 886.652)
(603.569, 123.77, -631.538, -415.048)
(162.177, -678.966, -13.963, -691.587)
(101.617, -469.89, -779.132, -473.424)
(756.015, -757.165, 18.234, -967.423)
(308.347, 824.077, -83.309, 591.069)
(-35.291, -803.102, 961.171, 925.142)
(914.324, -154.925, -341.553, -800.124)
(-511.385, 652.133, -251.767, 209.409)
(613.931, -126.75, -537.571, 631.347)
(-661.035, -357.238, 296.68, -503.22)
(-338.053, 776.322, 538.574, 89.141)
(630.34, 101.283, -23.341, -345.051)
(-952.303, 151.935, 63.128, 652.922)
(-542.041, 830.42, -265.942, -290.206)
(426.227, 947.89, -468.972, 390.643)
(575.321, -710.223, -949.085, -361.315)
(902.683, 866.297, -683.755, -454.137)
(-974.407, 451.254, 359.818, -450.565)
(519.369, -460.448, -883.607, -995.709)
(216.686, -434.049, -184.469, 597.079)
(-84.879, -107.598, 923.469, -849.555)
(420.158, -804.185, -122.027, -233.083)
(-217.849, -891.436, 768.682, -779.134)
(-164.937, 257.042, -516.724, -582.015)
(-730.908, 287.86, 110.804, 133.734)
(962.039, 782.994, -969.607, -669.034)
(-44.379, 678.9, 997.09, -95.07)
(842.189, 249.964, 564.426, -619.917)
(613.01, -95.626, 562.33, 427.558)
(-865.858, -580.9, -112.365, -744.919)
(245.017, 626.394, 319.359, -349.097)
(-219.326, -603.798, -20.203, -212.706)
(198.253, -72.137, -62.359, 281.301)
(847.544, -771.841, 68.916, 230.169)
(108.93, -251.859, 783.087, -467.457)
(308.762, 825.882, 705.2, -985.721)
(-32.011, -582.486, 813.078, -2.174)
(-908.736, -702.935, 368.97, 590.696)
(-432.585, -373.983, -192.087, 998.374)
(-350.548, -770.497, 348.139, 471.579)
(-428.23, -989.39, 846.834, 812.213)
(217.907, 650.938, 333.316, -153.55)
(954.604, 414.584, 65.304, 269.968)
(-642.108, 561.726, -191.546, -196.561)
(127.318, -470.184, 702.775, 135.488)
(-884.726, -29.501, -101.234, 924.42)
(-894.149, -87.753, 166.436, 886.786)
(-158.146, 465.553, -230.154, -36.39)
(358.222, 451.03, 276.507, -705.647)
(-555.602, 324.816, 24.203, -646.562)
(-909.967, -262.008, -739.709, 691.362)
(-657.681, 847.196, 433.454, 972.463)
(81.629, -971.737, 444.007, -989.836)
(-760.967, -547.516, 854.336, 660.419)
(-669.176, -775.313, 804.811, -667.404)
(428.593, 498.928, -930.163, 649.581)
(35.411, 619.378, 741.916, -514.023)
(923.706, -342.526, 242.915, -623.687)
(-705.824, 399.398, 685.951, 394.782)
(-972.759, -125.171, 670.564, -518.09)
```



```
Memproses dengan algoritma Brute Force...

ALGORITMA BRUTEFORCE
Titik 1 : (245.017, 626.394, 319.359, -349.097)
Titik 2 : (217.907, 650.938, 333.316, -153.55)
Jarak   : 199.426
Banyak Perhitungan : 2016
Waktu Eksekusi     : 32.021484375 milisekon

Memproses dengan algoritma Divide and Conquer...

ALGORITMA DIVIDE AND CONQUER
Titik 1 : (245.017, 626.394, 319.359, -349.097)
Titik 2 : (217.907, 650.938, 333.316, -153.55)
Jarak   : 199.426
Banyak Perhitungan : 129
Waktu Eksekusi     : 15.987548828125 milisekon
==================================================
    TERIMA KASIH TELAH MENGGUNAKAN PROGRAM KAMI
```

7. Masukan Tidak Sesuai (Validasi Input)



```
        PENCARIAN PASANGAN TITIK TERDEKAT
==================================================
Masukkan banyaknya titik : 1
Masukan tidak valid!
Masukkan banyaknya titik  : -3
Masukan tidak valid!
Masukkan banyaknya titik  : 2
Masukkan dimensi vektor  : 0
Masukan tidak valid!
Masukkan dimensi vektor  : -4
Masukan tidak valid!
Masukkan dimensi vektor  : 3

MEMBANGKITKAN 2 TITIK ACAK
(99.386, 457.854, 713.565)
(-918.385, -590.833, -445.144)
```