LAPORAN TUGAS KECIL 2
IF2211 STRATEGI ALGORITMA SEMESTER II 2022-2023

Pencarian Pasangan Titik Terdekat 3D dengan Algoritma Divide and Conquer

Disusun oleh:

Haidar Hamda 13521105

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG 2023

1. ALGORITMA

1.1 Algoritma Sorting

1. Pilih elemen larik terakhir sebagai pivot
2. Definisikan i dan j sebagai indeks pertama dari larik
3. Pertukarjan elemen larik[i] dengan larik[j] jika larik[i] <= pivot, *increment* j
4. *Increment* i
5. Ulangi langkah 3-4 selama i < banyak nya elemen dalam larik
6. Pertukarkan elemen larik[j] dengan elemen terakhir larik

1.2 Algoritma Pencarian Sepasang Titik Terdekat

1. Pilih elemen larik terakhir sebagai pivot
2. Definisikan i dan j sebagai indeks pertama dari larik
3. Pertukarjan elemen larik[i] dengan larik[j] jika larik[i] <= pivot, increment j
4. Increment i
5. Ulangi langkah 3-4 selama i < banyak nya elemen dalam larik
6. Pertukarkan elemen larik[j] dengan elemen terakhir larik

1.3 Algoritma Brute Force

1. Tetapkan jarak minimal dengan nilai tak terbatas
2. Iterasikan setiap titik dan bandingkan dengan titik lain yang memiliki indeks i+1
3. Jika jarak antar titik lebih rendah dari jarak minimal, simpan jarak antar titik dan pasangan titik tersebut

2. SOURCE CODE DALAM BAHASA RUBY

2.1 Main.rb

```
require_relative 'Algorithm.rb'
require_relative 'utils.rb'
require_relative 'plot.rb'

puts "number of points:"
n=gets.to_i
while n<2
    puts "ulangi input:"
    n=gets.to_i
end
puts "number of dimension:"
d=gets.to_i
while d<1
    puts "ulangi input:"
    d=gets.to_i
end
points= []
i=0
while i<n
    points.append([d.times.map{rand(-999.0..999.0)},i])
```

```
    i+=1
end
# points=points.sort_by{|item| item[0][0]}
points=quickSortPoint(points,0,points.length-1)

puts "divide and conquer:"
startdq=Time.now
pairs = divideNConquer(points)
finishdq=Time.now
print "distance: ",pairs[0]," between points: "
printPoint(getPointById(points,pairs[1]))
print " and "
printPoint(getPointById(points,pairs[2]))
print "\n"
print "number of euclidean distance calculation: ",
EuclidCounter.new.getCount-1,"\n"
print "execution time: "
print finishdq-startdq,"s\n\n"


EuclidCounter.new.setToZero
startbf=Time.now
bf=bruteForce(points)
finishbf=Time.now
puts "brute force:"
print "distance: ",bf[0]," between points: "
printPoint(getPointById(points,bf[1]))
print " and "
printPoint(getPointById(points,bf[2]))
print "\n"
print "number of euclidean distance calculation: ",
EuclidCounter.new.getCount-1,"\n"
print "execution time: "
print finishbf-startbf,"s\n"

plotPoints(points,getPointById(points,pairs[1]),getPointBy
Id(points,pairs[2]))
```

2.2 Utils.rb

```
def printPoint(point)
  i=0
  print "("
  while i<point[0].length()
    print point[0][i]
    if i!=point[0].length-1
      print ","
    end
```

```
    i+=1
  end
  print ")"
end

def quickSortPoint(array, low, high)
  if low < high
    partitionIdx = partition(array, low, high)
    quickSortPoint(array, low, partitionIdx-1)
    quickSortPoint(array, partitionIdx+1, high)
  end
  return array
end

def partition(array, low, high)
  pivot = array[high][0][0]
  j = low
  i = low
  while j < high
    if array[j][0][0] <= pivot
      array[j], array[i] = array[i], array[j]
      i += 1
    end
    j += 1
  end
  array[i], array[high] = array[high], array[i]
  return i
end

def getPointById(points,id)
  return points.find{|item| item[1]==id}
end
```

2.3 Algorithm.rb

```
class EuclidCounter
  @@count=0
  def initialize
    @@count+=1
  end
  def getCount
    return @@count
  end
  def setToZero
    @@count=0
  end
end
```

```
def calculateDistance(p1,p2)
  i=0
  temp=[]
  while i<p1.length()
    temp.append((p1[i]-p2[i])**2)
    i+=1
  end
  p=EuclidCounter.new()
  return Math.sqrt(temp.sum)
end

def getPointInside(midPoint,points,delta)
  pointsInside = []
  i=0
  while i<points.length()
    if points[i][0][0]<midPoint+delta &&
points[i][0][0]>midPoint-delta
      pointsInside.append(points[i])
    end
    i+=1
  end
  return pointsInside
end


def bruteForce(points)
  min=[Float::MAX,-1,-1]
  i =0
  while i<points.length()
    j=i+1
    while j<points.length()
      d=calculateDistance(points[i][0],points[j][0])
      if d<min[0]
        min=[d,points[i][1],points[j][1]]
      end
      j+=1
    end
    i+=1
  end
  return min
end

def divideNConquer(points)
  if points.length()<=3
    return bruteForce(points)
  else
    rightPoints=points.slice(0,points.length/2)
```

```
leftPoints=points.slice(points.length/2,points.length())
    rightPairs=divideNConquer(rightPoints)
    leftPairs=divideNConquer(leftPoints)
    if leftPairs[0]<rightPairs[0]
      closest=leftPairs
    else
      closest=rightPairs
    end

pointsInside=getPointInside(points[points.length/2][0][0],
points,closest[0])
    if !pointsInside.empty?
      insidePairs=bruteForce(pointsInside)
      if insidePairs[0]<closest[0]
        return insidePairs
      else
        return closest
      end
    else
      return closest
    end
  end
end
```

2.4 Plot.rb

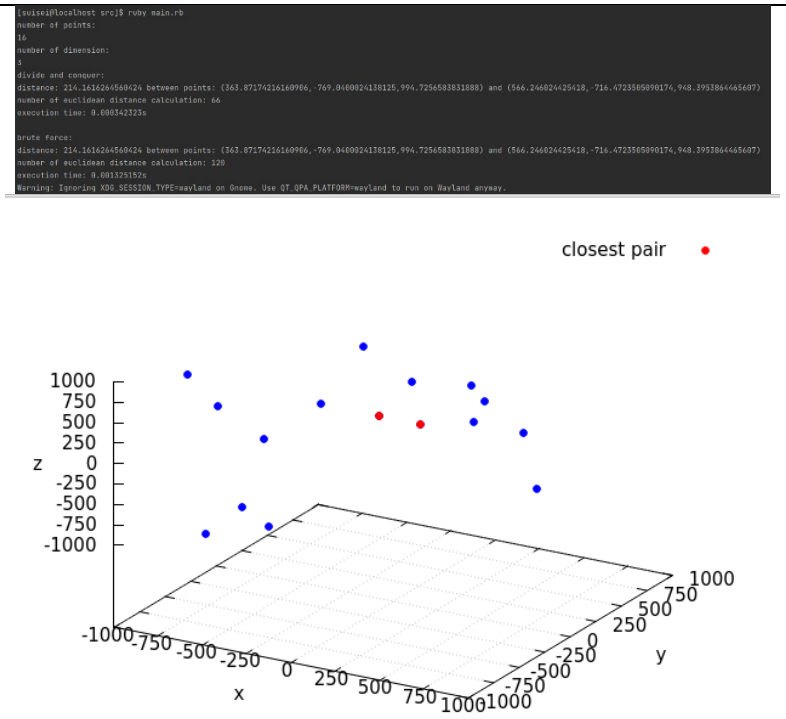
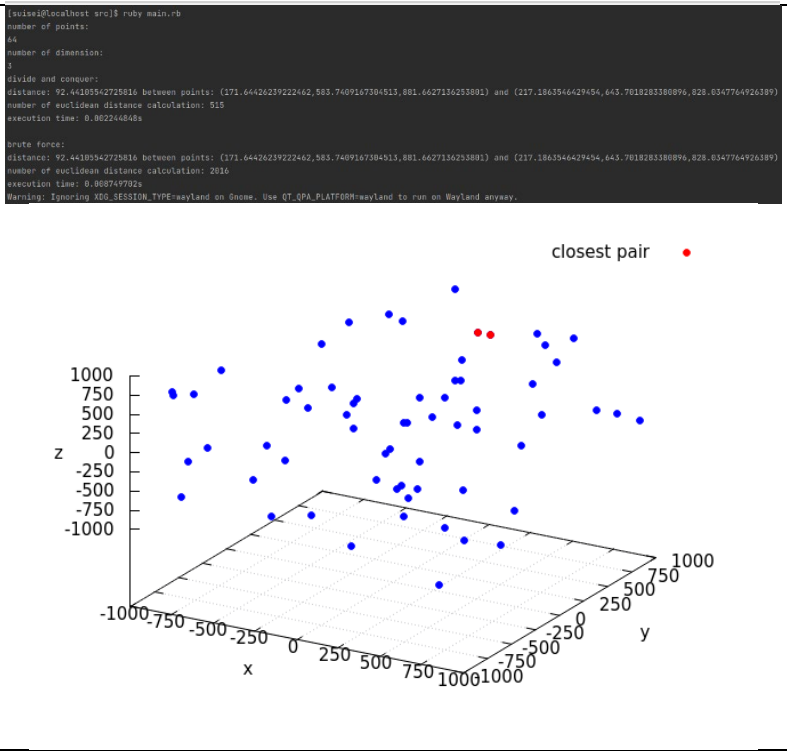
```
require 'gnuplot'

def plotPoints(points,p1,p2)
  if p1[0].length==3
    Gnuplot.open do |gp|
      Gnuplot::SPlot.new(gp) do |plot|
        plot.grid
        plot.xrange "[-1000:1000]"
        plot.yrange "[-1000:1000]"
        plot.zrange "[-1000:1000]"
        plot.xtics 250
        plot.ytics 250
        plot.ztics 250
        x=points.clone
        x.map{|item| item[0][0]}
        y=points.clone
        y.map{|item| item[0][1]}
        z=points.clone
        z.map{|item| item[0][2]}
        plot.xlabel "x"
        plot.ylabel "y"
```

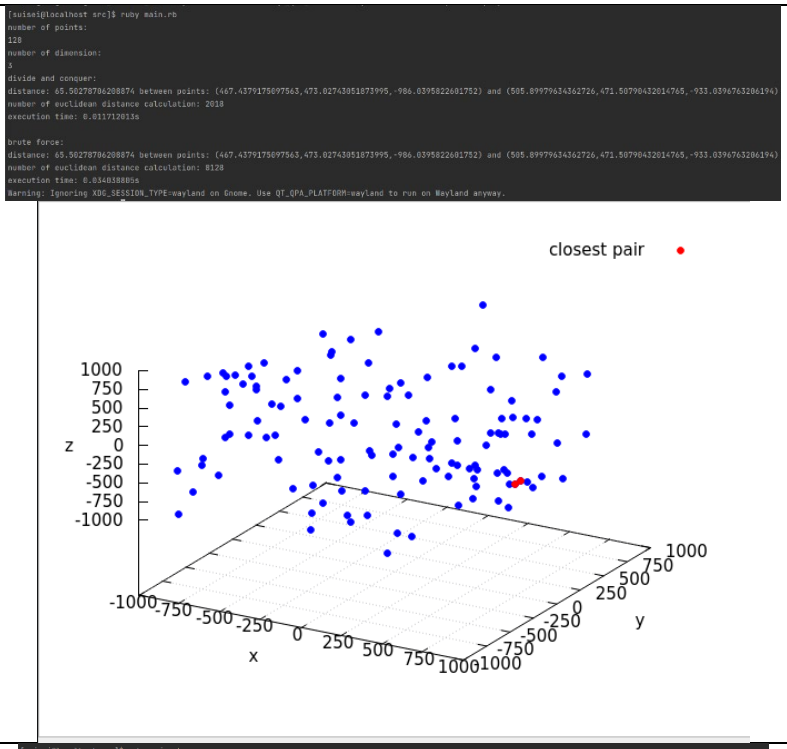
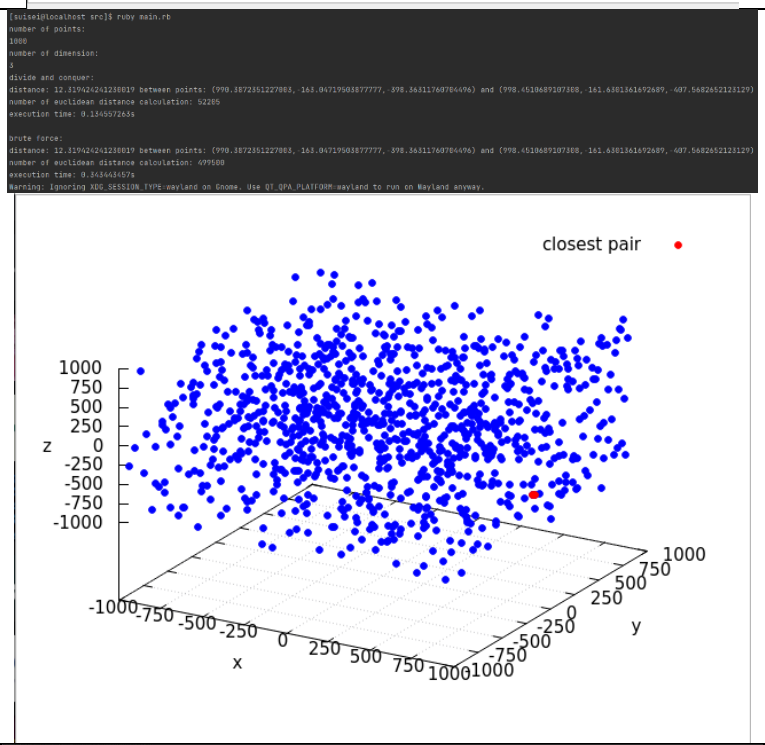
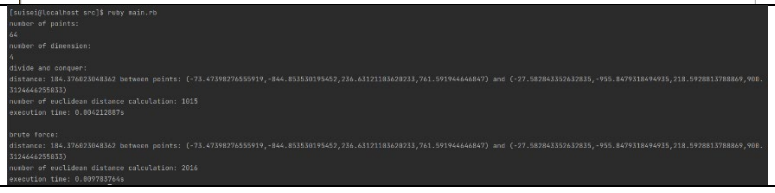
```
        plot.zlabel "z"
        plot.data<<Gnuplot::DataSet.new([x,y,z]) do |ds|
          ds.with="points ps 1 pt 7 lc rgb 'blue'"
          ds.notitle
        end
        x2=[p1[0][0],p2[0][0]]
        y2=[p1[0][1],p2[0][1]]
        z2=[p1[0][2],p2[0][2]]
        plot.data<<Gnuplot::DataSet.new([x2,y2,z2]) do
|ds1|
          ds1.with="points ps 1 pt 7 lc rgb 'red'"
          ds1.title= 'closest pair'
        end
      end
    end
    elsif p1[0].length==2
    Gnuplot.open do |gp|
      Gnuplot::Plot.new(gp) do |plot|
        plot.grid
        plot.xrange "[-1000:1000]"
        plot.yrange "[-1000:1000]"
        plot.xtics 250
        plot.ytics 250
        x=points.clone
        x.map{|item| item[0][0]}
        y=points.clone
        y.map{|item| item[0][1]}
        plot.xlabel "x"
        plot.ylabel "y"
        plot.data<<Gnuplot::DataSet.new([x,y]) do |ds|
          ds.with="points ps 1 pt 7 lc rgb 'blue'"
          ds.notitle
        end
        x2=[p1[0][0],p2[0][0]]
        y2=[p1[0][1],p2[0][1]]
        plot.data<<Gnuplot::DataSet.new([x2,y2]) do |ds1|
          ds1.with="points ps 1 pt 7 lc rgb 'red'"
          ds1.title= 'closest pair'
        end
      end
    end
  end
end
```

3. HASIL EKSEKUSI PROGRAM

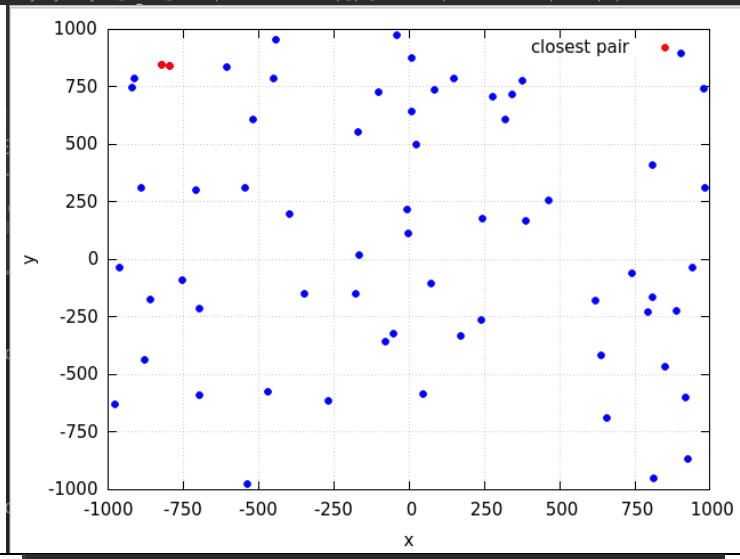
Pengujian eksekusi dilakukan pada computer dengan spesifikasi sebagai berikut:

CPU: Intel Core i7-8550U
RAM: 8gb 2400MHz
Dengan sistem operasi: Red Hat Enterprise Linux 9

Banyak Titik (n)	Dimensi	Tangkapan Layar
16	3	
64	3	

128	3	
1000	3	
64	4	

<table>
<tr>
<td>64</td>
<td>2</td>
<td>
<pre>[suisei@localhost src]$ ruby main.rb
number of points:
64
number of dimension:
2
divide and conquer:
distance: 27.402152585027114 between points: (-821.1394177298171,845.1741849553036) and (-794.7001297551199,837.9740448397129)
number of euclidean distance calculation: 298
execution time: 0.001578514s

brute force:
distance: 27.402152585027114 between points: (-821.1394177298171,845.1741849553036) and (-794.7001297551199,837.9740448397129)
number of euclidean distance calculation: 2016
execution time: 0.006418063s
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.</pre>


</td>
</tr>
<tr>
<td>Masukan tidak sesuai</td>
<td></td>
<td>
<pre>[suisei@localhost src]$ ruby main.rb
number of points:
0
ulangi input:
-1
ulangi input:
d
ulangi input:
a
ulangi input:
a aja j
ulangi input:
10
number of dimension:
a
ulangi input:
-1
ulangi input:
a w d
ulangi input:
1
divide and conquer:
distance: 5.87449445598304 between points: (722.4792512575216) and (728.3537457135046)
number of euclidean distance calculation: 8
execution time: 0.000270545s</pre>
</td>
</tr>
</table>

4. LAMPIRAN

4.1 *Repository GitHub*

https://github.com/haidarhamda/Tucil2_13521105

4.2 Tabel Ketercapaian Program

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa ada 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	✓	

Referensi

- Closest Point Search in High Dimensions - Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Co (columbia.edu)
- 1976ShamosBentley.pdf (cmu.edu)