Closest Pair of Points (2D)

DnC vs *Brute Force*

Kelompok 4

Sammi Aldhi Yanto (k)
M·Fachren
Winner Wijaya
Hendy Saputra
Yochiko Narasaki
Ghaza Muhammad Alghazali
Seteven
Aryo Rachmad
Dian Kusmawati
Rahmatul Izzah Annisa
Elvina Carolina
Siti Khanifatun Ma'rifah

Fachren

```java
class Point{
    public double x;
    public double y;
    public Point(double x,double y){
        this.x = x;
        this.y = y;
    }
    public String toString(){
        return "("+x+","+y+")";
    }
}
```

Fachren

```
public Point[] getClosestPair(Point[] ps){
    // urutkan berdasarkan absis x
    sortPointsByX(ps);

    // kemudian hitung
    return getClosestPair(ps,0,ps.length-1);
}
```

Winner

```java
/**
 * menggunakan quick sort
 * kompleksitas: O(n log n)
 */
public void sortPointsByX(Point[] ps) {
    int size = ps.length;
    quickSort(ps,0,size-1);
}
```

Hendy

```java
public void quickSort(Point[] ps, int left, int right) {
    if (left >= right)
        return;
    int i = left, j = right;
    Point tmp;
    double pivot = (ps[left].x + ps[right].x) / 2;

    while (i <= j) {
        while (ps[i].x < pivot)
            i++;
        while (ps[j].x > pivot)
            j--;
        if (i <= j) {
            tmp = ps[i];
            ps[i] = ps[j];
            ps[j] = tmp;
            i++;
            j--;
        }
    }

    if (left < i - 1)
        quickSort(ps, left, i - 1);
    if (i < right)
        quickSort(ps, i, right);
}
```

Yochiko

```java
/**
 * @param ps: sebelumnya udah di sort berdasarkan absis
 */
public Point[] getClosestPair(Point[] ps, int start, int end){
    if(start==end)
        return null;
    if(start+1==end)
        return new Point[]{ps[start],ps[end]};

    // menentukan medium X
    int mediumX = (start+end)/2;
```

Ghaza

```
// kemudian solve secara rekursif
Point[] leftClosestPair = this.getClosestPair(ps, start, mediumX);
Point[] rightCloestPair = this.getClosestPair(ps, mediumX, end);
Point[] returnPair;
double returnDelta;
```

Seteven

```
// cari minimal distance untuk masing-masing (sebelah kiri dan kanan)
double leftDelta = this.getDistance(leftClosestPair);
double rightDelta = this.getDistance(rightCloestPair);
double delta;
if(leftDelta < rightDelta){
    delta = leftDelta;
    returnPair = leftClosestPair;
}else{
    delta = rightDelta;
    returnPair = rightCloestPair;
}
returnDelta = delta;
```

Aryo

```
// mentukan jarak minimal antara pasangan titik yg terletak di satu sisi dan yg terletak di sisi lain
int leftEdge = this.getLeftEdgeOfDelta(ps, mediumX, leftDelta);
if(leftEdge < start)
    leftEdge = start;
int rightEdge = this.getRightEdgeOfDelta(ps, mediumX, rightDelta);
if(rightEdge > end)
    rightEdge = end;
```

Dian

```
// urutkan points yang edge sebelah kiri dan kanan berdasarkan ordinat atau Y
// Time: O(n log n)
int yOrderInfo[][] = this.sortPointsByY(ps,leftEdge,rightEdge);
int idxToOrderY[] = yOrderInfo[0];
int orderYToIdx[] = yOrderInfo[1];
```

Rahmatul

```java
// periksa apakah ada titik di [mediumX, rightEdge] yang memiliki jarak kurang dari delta ke titik ini
for(int i=leftEdge;i<=mediumX;i++){
    int orderY = idxToOrderY[i-leftEdge];
    for(int j=orderY-1;j>=0;j--){
        Point[] pair = new Point[]{ps[i],ps[orderYToIdx[j]]};
        if(this.getYDistance(pair) > delta)
            break;
        double distance = this.getDistance(pair);
        if(distance < returnDelta){
            returnDelta = distance;
            returnPair = pair;
        }
    }

    for(int j=orderY+1;j<idxToOrderY.length;j++){
        Point[] pair = new Point[]{ps[i],ps[orderYToIdx[j]]};
        if(this.getYDistance(pair) > delta)
            break;
        double distance = this.getDistance(pair);
        if(distance < returnDelta){
            returnDelta = distance;
            returnPair = pair;
        }
    }
}
return returnPair;
```

Elvina

```java
// mengurutkan ordinat menggunakan quick sort, cost n log n
public int[][] sortPointsByY(Point[] ps, int leftEdge, int rightEdge) {
    int size = rightEdge - leftEdge + 1;
    int orderToIdx[] = new int[size];
    for(int i=0;i<size;i++)
            orderToIdx[i] = leftEdge+i;
    quickSortForDeltaArea(ps, orderToIdx, 0, size-1);
    int idxToOrder[] = new int[size];
    for(int i=0;i<size;i++){
            idxToOrder[orderToIdx[i]-leftEdge] = i;
    }
    return new int[][]{idxToOrder,orderToIdx};
}
```

Siti

```java
// quick sort untuk delta area, kiri medium & kanan medium
public void quickSortForDeltaArea(Point[] ps,int[] indexs, int left, int right) {
    if (left >= right)
        return;
    int i = left, j = right;
    int tmp;
    double pivot = ( ps[indexs[left]].y + ps[indexs[right]].y) / 2;

    while (i <= j) {
        while (ps[indexs[i]].y < pivot)
            i++;
        while (ps[indexs[j]].y > pivot)
            j--;
        if (i <= j) {
            tmp = indexs[i];
            indexs[i] = indexs[j];
            indexs[j] = tmp;
            i++;
            j--;
        }
    }


    if (left < i - 1)
        quickSortForDeltaArea(ps, indexs, left, i - 1);
    if (i < right)
        quickSortForDeltaArea(ps, indexs, i, right);
}
```

Chiko

```java
// hitung tepi kanan area delta
public int getRightEdgeOfDelta(Point[] ps, int mediumX, double rightDelta) {
    for(int i=mediumX;i<ps.length;i++){
        if(ps[i].x - ps[mediumX].x>rightDelta)
            return i-1;
    }
    return ps.length-1;
}
```

Winner

```
// hitung tepi kiri area delta
public int getLeftEdgeOfDelta(Point[] ps, int mediumX, double leftDelta) {
    for(int i=mediumX;i>=0;i--){
        if(ps[mediumX].x - ps[i].x>leftDelta)
            return i+1;
    }
    return 0;
}
```

Ghaza

```java
// = √(x2-x1)^2+(y2-y1)^2
public double getDistance(Point[] ps) {
    if(ps==null||ps.length!=2){
        return -1;
    }
    return Math.pow(
        (ps[0].x-ps[1].x) * (ps[0].x-ps[1].x) +
        (ps[0].y-ps[1].y) * (ps[0].y-ps[1].y), 0.5);
}
```

Hendy

```java
// generate titik acak
public Point[] generateRandomPoints(int size,double maxX, double minX,double maxY,double minY){
    Point[] points = new Point[size];
    for(int i=0;i<size;i++){
        points[i] = new Point(
            Math.random()*(maxX-minX) + minX,
            Math.random()*(maxY-minY) + minY);
    }
    return points;
}
```

Seteven

```java
// kebutuhan log, input, dan waktu
logs = new StringBuilder();
int totalLoop = Integer.parseInt(points.getText());
int totalPointsPerLoop = Integer.parseInt(loops.getText());

double minX = Double.parseDouble(minx.getText());
double maxX = Double.parseDouble(maxx.getText());
double minY = Double.parseDouble(miny.getText());
double maxY = Double.parseDouble(maxy.getText());

int passed = 0;
long nlognTime = 0;
long nsqTime = 0;
long startTime;
```

Siti

```
indikator.setText("Done");
log.setText(logs.toString());
sukses.setText(passed + "/" + totalLoop);
bft.setText(String.valueOf(nsqTime));
dnct.setText(String.valueOf(nlognTime));
```

Dian

```java
private void resetActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_resetActionPerformed
    // set button
    points.setText("");
    loops.setText("");
    minx.setText("");
    miny.setText("");
    maxx.setText("");
    maxy.setText("");
    indikator.setText("");
    log.setText("");
    sukses.setText("");
    bft.setText("");
    dnct.setText("");
}
```

Sammi

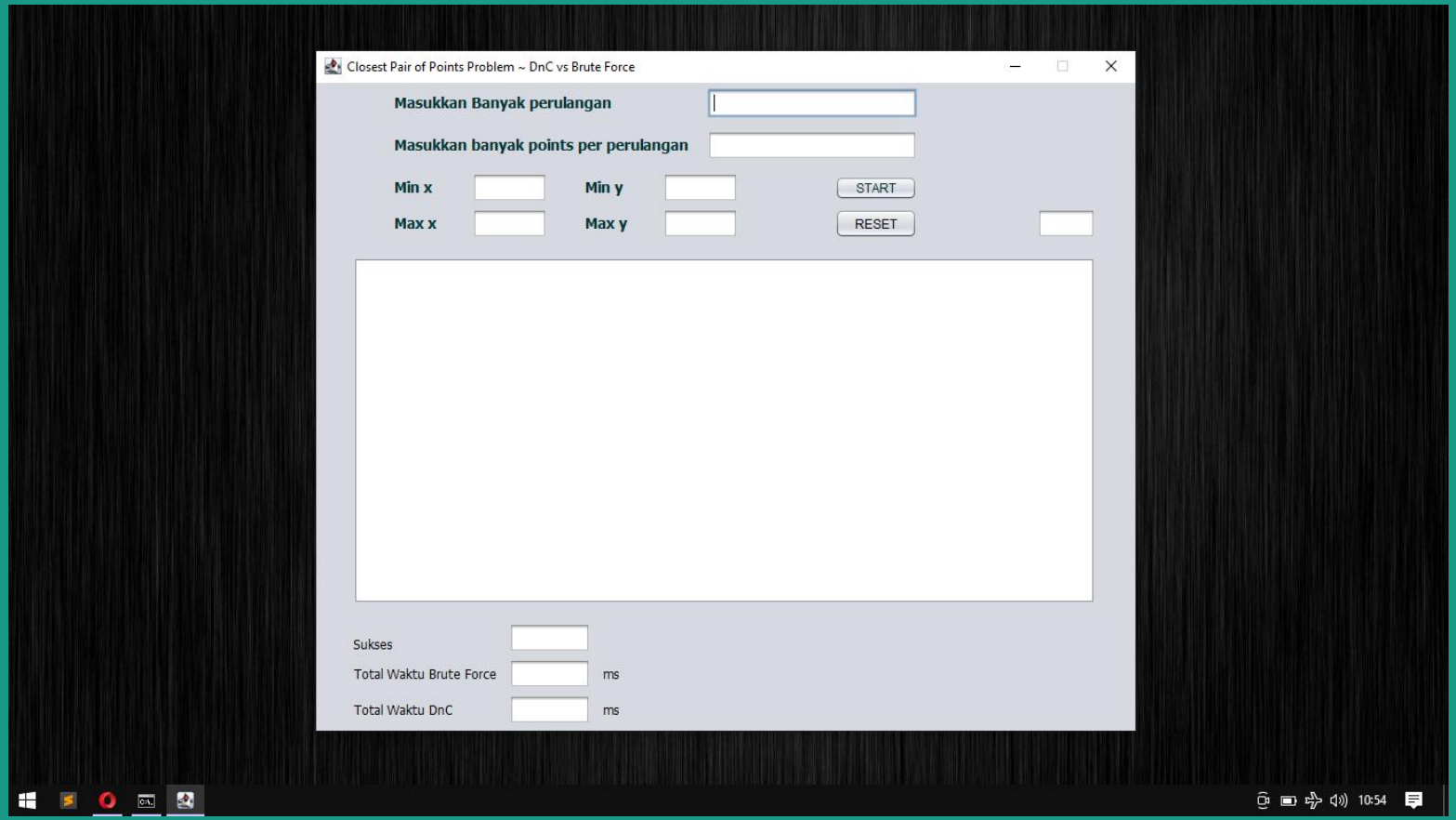
```
for(int k=0;k<totalLoop;k++){
    Point[] ps = cpp.generateRandomPoints(totalPointsPerLoop, maxX, minX, maxY, minY);
    startTime = new Date().getTime();

    Point[] pair = cpp.getClosestPair(ps);
    nlognTime += (new Date().getTime() - startTime);

    double distance = cpp.getDistance(pair);
    logs.append("[ok] = " + pair[0].toString() + "," + pair[1].toString() + " = " + distance + "\n");

    startTime = new Date().getTime();
    pair = cpp.bruteForce(ps);

    nsqTime += (new Date().getTime() - startTime);
    double distance2 = cpp.getDistance(pair);
    if(distance==distance2){
            passed++;
    }
}
```


Closest Pair of Points Problem ~ DnC vs Brute Force
Masukkan Banyak perulangan
Masukkan banyak points per perulangan
Min x
Min y
START
Max x
Max y
RESET
Sukses
Total Waktu Brute Force
ms
Total Waktu DnC
ms
10:54

DEMO

Efisiensi

Setiap instance permsalahan akan terjadi N pass untuk setiap ttik dan masing-masing pass akan meninjau kemabali n titk. Sehingga kompleksitas untuk algotima brute force O(n^2) nested loop

Fungsi conquer, memiliki kompleksitas O(n), namun ppada awal fungsi conquer, titik disekitar garis pembagi diurutkan berdasarkan ordinat, (asumsi algo berorder ***n log n****, maka kompleksitas total algoritmanya adalah o(****n log n****)*

Kelebihan & Kekurangan

- ***Brute force*** Cukup baik untuk **N** nya yang tidak terlalu besar
- Untuk **N** yg besar (n > 100·000), komputasi dengan algoritma *brute force* ini akan memakan waktu yg lama
- Karena *Algoritma DnC* memiliki kompleksitas **O(n log n)**, untuk **N** yang besar masih cukup mangkus untuk digunakan

Thanks