Algoritma Analizi Ödev – 1

Konu : Divide and Conquer Algorithms
Problem: Closest-Pair Problem

Ders Yürütücüsü:
Doç. Dr. M. Elif KARSLIGİL

Ali Alparslan PINAR
14011079

Yöntem

Problem:

Kullanıcı tarafından girilen N adet nokta içerisinden birbirine en yakın olan ikilinin bulunması gerekmektedir.

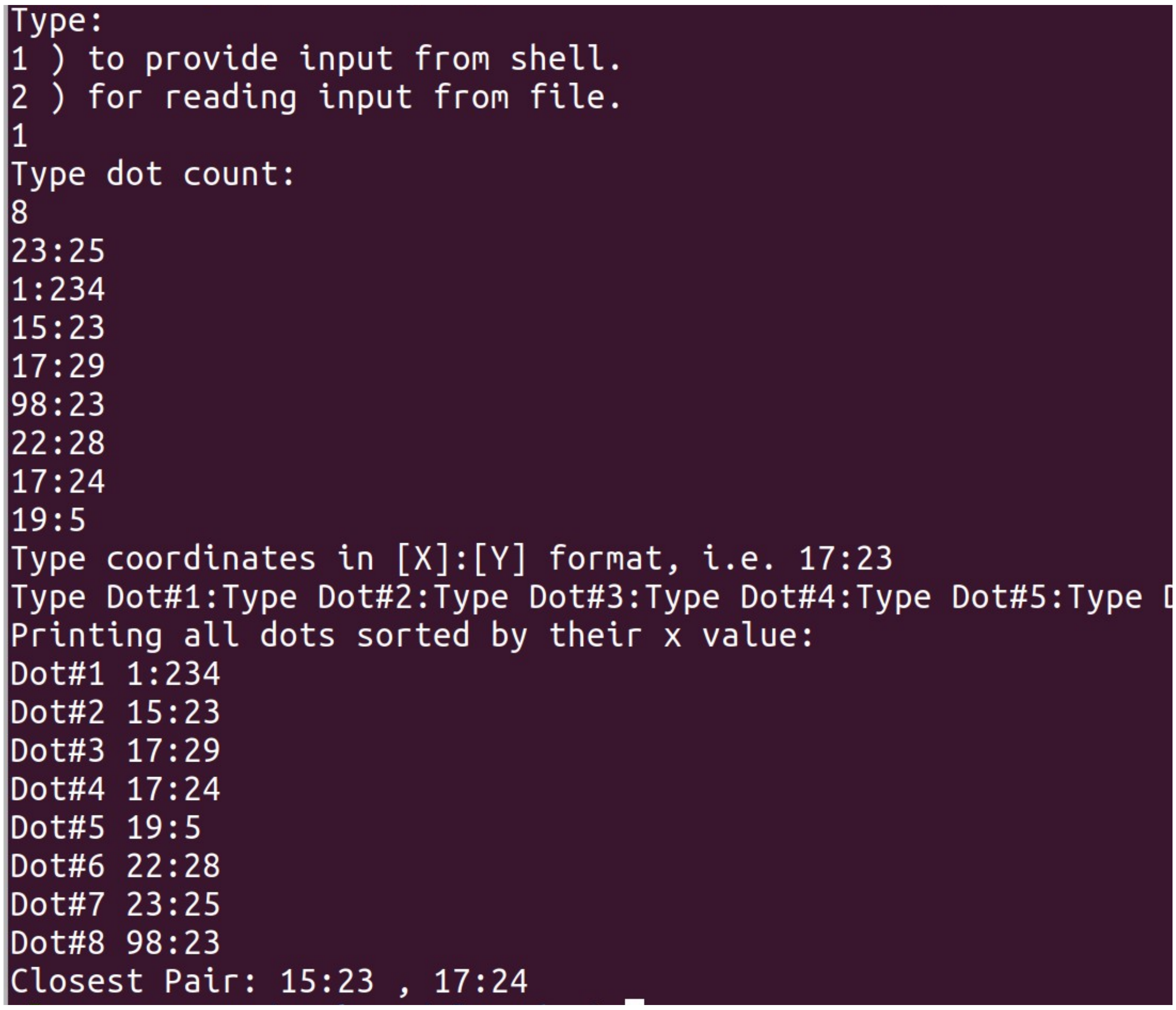
Çözüm:

Noktaların sayısı ve koordinatları kullanıcıdan alındıktan sonra Quick Sort algoritması ile noktalar x koordinatlarına göre küçükten büyüğe sıralanmaktadır.
Sonrasında birbirine en yakın iki nokta recursive findClosestPair fonksiyoni ile bulunmaktadır.
Fonksiyon şu şekilde çalışmaktadır:
Nokta sayısı üç veya daha az ise tüm noktalar birbiri ile karşılaştırılarak birbirine en yakın nokta bulunur ve döndürülür.
Nokta sayısının üçten fazla olması durumunda ise:
Noktaların x değerlerinin medyanı hesaplanır. Medyanın iki tarafından (noktaların herhangi biri direkt medyan üzerinde de olabilir.) en yakın noktalar sınır olarak alınarak mevcut aralık iki aralığa bölünür ve fonksiyonun kendisi bu aralıklar için çağrılır. Yapılan iki çağrıdan hangisinin sonucu birbirine daha yakın bir çift verirse o çift mevcut en yakın çift olarak atanır.
Mevcut en yakın çiftin birbirine uzaklığını d olarak kabul edersek; Medyan-d ile Medyan ve medyan ile medyan+d aralığında kalan noktalar birbirileri ile döngü içerisinde karşılaştırılarak birbirine daha yakın bir nokta bulunması halinde mevcut en yakın çift olarak atanır.
Fonksiyon mevcut en yakın çifti ve uzaklığı içeren veri yapısını döndürerek biter.
Fonkisyonun karmaşıklığı: (recursive call sayısı)

```
F(n) = {1          n<=3;
       {2*f(n/2)   n>3;
```

Ortalama rescursive call sayısı F(n) = n/2 olmaktadır.

Uygulama



```
Type:
1 ) to provide input from shell.
2 ) for reading input from file.
1
Type dot count:
8
23:25
1:234
15:23
17:29
98:23
22:28
17:24
19:5
Type coordinates in [X]:[Y] format, i.e. 17:23
Type Dot#1:Type Dot#2:Type Dot#3:Type Dot#4:Type Dot#5:Type D
Printing all dots sorted by their x value:
Dot#1 1:234
Dot#2 15:23
Dot#3 17:29
Dot#4 17:24
Dot#5 19:5
Dot#6 22:28
Dot#7 23:25
Dot#8 98:23
Closest Pair: 15:23 , 17:24
```

Kaynak Kodu:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define ALLOCINCREMENT 5 //How much size of dot array will be incremented when readin
g input from a file

typedef struct{
   int x;
   int y;
}dot;

typedef struct{
   dot* dot1; // Pointer to first dot
   dot* dot2; // Poniter to second dot
   double distance; // Distance between dots
}pair;

// Swaps coordinates of two dots.
void swap(dot* dot1, dot* dot2){
   dot temp;
   temp = *dot1;
   *dot1 = *dot2;
   *dot2 = temp;
}

// Sorts dots by x coordinate in place
void quickSort(dot* dots, int l, int r){
   if(l >= r){
      return;
   }
   int pivot = dots[r].x; //Pick last dot as pivot
   int nextPos = l; // This variables stores where will be the next dot with lower x than pivot
, placed.
   //Shifts dots with lower x than pivot to beginning
   for(int i = l; i <= r; i++){
      if(dots[i].x <= pivot){
         swap(&dots[nextPos], &dots[i]);
         nextPos++;
      }
   }
   // nextPos-1 is where pivod placed right now
   quickSort(dots, l, nextPos-2); // Sort left side of pivot
   quickSort(dots, nextPos, r);    // Sort right side of pivot
}

//Prints coordinates of dot
void printDots(int n, dot* dots){
   int i;
```

```
    for(i = 0; i < n; i++){
        printf("Dot#%d %d:%d\n", i+1, dots[i].x, dots[i].y);
    }
}

// Returns the distance between two dots
double calcDistance(dot dot1, dot dot2){
    return pow( pow(dot1.x-dot2.x, 2)+pow(dot1.y-dot2.y, 2), 0.5 );
}

// returns and array consists of indexes of closest pairs
pair findClosestPair(dot* dots, int l, int r){
    double median; // Median of X axis values of coords
    pair closestPair; // To store closest pair found
    pair closestPairR; // To stores results from recursive call responsible from rigt side
    pair closestPairL; // To stores results from recursive call responsible from left side
    double distance; // Temp var for storing calculated distances
    int count = r-l+1; // Dot count in current range
    int i, j; // loop variables
    int jmax; // Stores the index of first dot that falls the other side of median+d
    if(count <= 3){ // If count of dots less or equal than 3 then find the closest pair with brut
e-force
        closestPair.dot1 = &dots[l];
        closestPair.dot2 = &dots[l+1];
        closestPair.distance = calcDistance(dots[l], dots[l+1]);
        for(i = l; i <= r; i++){ // Compare all dots with loops
            for(j = i+1; j <= r; j++){
                distance = calcDistance(dots[i],dots[j]);
                if( distance < closestPair.distance){
                    closestPair.dot1 = &dots[i];
                    closestPair.dot2 = &dots[j];
                    closestPair.distance = calcDistance(dots[i],dots[j]);
                }
            }
        }
        return closestPair;
    }

    // Finding median of X coordinates
    median = (count%2 == 0) ? (dots[l+count/2-1].x+dots[l+count/2].x)/2.0 : dots[l+count/
2].x;

    // Recursive calls for both sides of median
    closestPairR = findClosestPair(dots, l, l+count/2-1);
    closestPairL = findClosestPair(dots, l+count/2, r);

    // Pick the one that has lower distance
    closestPair = (closestPairR.distance > closestPairL.distance) ? closestPairL : closestPairR;

    //Checking dots in the areas between x=m and x=distance, -distance
    for(i = l+count/2-1; i >= l && closestPair.distance >= (median-dots[i].x); i--){
        for(j = l+count/2; j <= r && closestPair.distance >= (dots[j].x-median); j++){
            distance = calcDistance(dots[i], dots[j]);
            //If distance between two dots shorter than current closests pair's then pick those as
new closest pair
            if(distance < closestPair.distance){
                closestPair.distance = distance;
                closestPair.dot1 = &dots[i];
                closestPair.dot2 = &dots[j];
            }
        }
    }
    return closestPair;
}

void main(){
    int i; // Loop var.
    int count; // Count of dots
    pair closestPair; // To store indexes of closest pair
    dot* dots; // Array that hold coordinates of dots
    int inputType; // Stores selection of source of data. 1 for shell, 2 for file
    char fileName[50]; // name of the input file in case of source is selected to be from a file
    FILE* fileHandle; // Needed when reading from file
    printf("Type:\n1 ) to provide input from shell.\n2 ) for reading input from file.\n");
    scanf("%d",&inputType);
    if(inputType == 1){
        printf("Type dot count:\n");
        scanf("%d",&count);
        dots = malloc(sizeof(dot) * count);
        printf("Type coordinates in [X]:[Y] format, i.e. 17:23\n");
        for(i = 0; i < count; i++){
            printf("Type Dot#%d:", i+1);
            scanf("%d:%d", &dots[i].x, &dots[i].y);
        }
    }else if(inputType == 2){
        printf("Type Filename:");
        scanf("%s", fileName);
        fileHandle = fopen(fileName, "r");
        if(!fileHandle){
            printf("Couldn't open file to read!");
            return;
        }
        dots = malloc(sizeof(dot) * ALLOCINCREMENT);
        count = 0;
        while(fscanf(fileHandle, "%d %d", &dots[count].x, &dots[count].y) > 0){
            count++;
            if(count%ALLOCINCREMENT == 0){ //If it is true, then means we need to increase c
apacity
                dots = realloc(dots, sizeof(dot) * (count + ALLOCINCREMENT) );
            }
        }
    }else{
        printf("It is not hard to type just 1 or 2 !\n");
        return;
    }

    quickSort(dots, 0, count-1);

    printf("\nPrinting all dots sorted by their x value:\n");
    printDots(count, dots);

    closestPair = findClosestPair(dots, 0, count-1);

    printf("Closest Pair: %d:%d , ", closestPair.dot1->x, closestPair.dot1->y);
    printf("%d:%d\n", closestPair.dot2->x, closestPair.dot2->y);

    free(dots);
}
```