



Dersin adı:Algoritma Analizi

Grup Numarası-2

Ödev-1

Recep Furkan Koçyiğit

16011043

Ödev Konusu:Divide And Conquer Algoritmaları

Yöntem:

a-)Problem: Bu ödevde “input.txt” dosyası içerisinde kartezyen düzleminde verilen n adet nokta arasında birbirine en yakın 2 noktanın bulunması istenmektedir.

b-)Çözüm Yaklaşımı: Problemin çözümü için Divide And Conquer yaklaşımı kullanılacaktır. İlk olarak okunan n adet noktanın x eksenine göre sıralanacak daha sonra düzlem medyan değerine göre bölünerek küçük parçalara ayrılacaktır. Rekürsif fonksiyonumuzun durma noktası ise verilen nokta sayısının 3 ten düşük olmasıdır. Bu durumda Brute Force yaklaşım kullanılacaktır. Rekürsif fonksiyonun çağırılma koşulu ise medyan değerinin sol tarafı ve sağ tarafı için fonksiyonu çağırma işlemi yapılmaktadır.

c-)Algoritmanın karmaşıklığının hesaplanması: Problemi medyan değerine bölerek ilerlediğimiz ayrıca sağ ve sol olarak ikiye ayırdığımız için $T(n)=2T(n/2)$ gelecektir. Rekürsif fonksiyonu her çağırışımızda medyan değerinden x ekseninde d kadar uzak noktaları bulmak için gereken işin karmaşıklığı $O(n)$ ve medyan değerinden uzak olan noktalar arasındaki en kısa mesafeyi bulmak için gereken işin karmaşıklığı da $O(n)$ olduğundan bu bağıntıyı yazabiliriz:

$$T(n)=2T(n/2)+O(n)+O(n)$$

Master teoreminden $a=2, b=2$ ve $d=1$ olduğundan dolayı;

Karmaşıklığı $O(n \log n)$ dir.

Uygulama:

1.Örnek: Input.txt değerleri aşağıdaki gibi olsun:

2 3

3 7

4 6

5 1

7 12

8 10

En yakın iki noktayı bulmak için func fonksiyonu çağırıldığında şu işleri yapacaktır:

`Func(space,0,5)=min(Func(space,0,2),Func(space,2,5))`

//Func(space,0,2) eleman sayısı 3 olduğu için brute force yaklaşım izleyecektir.

$\text{Func}(\text{space}, 0, 2) = 1.41$

$\text{Func}(\text{space}, 2, 5) = \min(\text{Func}(\text{space}, 2, 3), \text{Func}(\text{space}, 3, 5))$

$\text{Func}(\text{space}, 2, 3) = 5.10$

$\text{Func}(\text{space}, 3, 5) = 2.24$

$\text{Func}(\text{space}, 2, 5) = 2.24$

$\text{Func}(\text{space}, 0, 5) = 1.41$ olacaktır.

2.Örnek: Input.txt değerleri aşağıdaki gibi olsun:

3 7

4 6

5 1

7 12

8 10

$\text{Func}(\text{space}, 0, 4) = \min(\text{Func}(\text{space}, 0, 2), \text{Func}(\text{space}, 2, 4))$

$\text{Func}(\text{space}, 0, 2) = 1.41$

$\text{Func}(\text{space}, 2, 5) = 2.24$

$\text{Func}(\text{space}, 0, 4) = 1.41$

3.Örnek: Input.txt değerleri aşağıdaki gibi olsun:

8 10

4 6

5 1

7 12

$\text{Func}(\text{space}, 0, 3) = \min(\text{Func}(\text{space}, 0, 1), \text{Func}(\text{space}, 1, 3))$

$\text{Func}(\text{space}, 0, 1) = 5.10$

$\text{Func}(\text{space}, 1, 3) = 2.24$

$\text{Func}(\text{space}, 0, 3) = 2.24$

4.Örnek(Son Taşa Ulaşabilen):Input.txt değerleri aşağıdaki gibi olsun:

4 13

5 6

3 7

7 12

2 1

$\text{Func}(\text{space},0,4)=\min(\text{Func}(\text{space},0,2),\text{Func}(\text{space},2,4),\text{medyandan } d \text{ kadar uzak değerler arası mesafe})$

$\text{Func}(\text{space},0,2)=6.08$

$\text{Func}(\text{space},2,4)=3.16$

medyandan d kadar uzak mesafeler arası=2.24

$\text{Func}(\text{space},0,4)=2.24$

C dilindeki kod:

```
#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<math.h>

#define MIN(a,b) (a)<(b)?(a):(b)

#define INF 100000005

#define MAX 50

struct Dot{

    int x;

    int y;

};

typedef struct Dot Dot;

void qSort(Dot *space,int lo,int hi){

    if(lo<hi){

        int q=partition(space,lo,hi);

        qSort(space,lo,q-1);

        qSort(space,q+1,hi);

    }

}

int partition(Dot *arr,int lo,int hi){

    int i,x,j,temp;

    x=hi;

    i=lo-1;

    for(j=lo;j<=hi;j++){

        if(arr[j].x<arr[x].x){
```

```

        i++;

        temp=arr[i].x;
        arr[i].x=arr[j].x;
        arr[j].x=temp;

        temp=arr[i].y;
        arr[i].y=arr[j].y;
        arr[j].y=temp;

    }
}

i++;
temp=arr[i].x;
arr[i].x=arr[x].x;
arr[x].x=temp;
temp=arr[i].y;
arr[i].y=arr[x].y;
arr[x].y=temp;
return i;
}

float findDist(Dot d1, Dot d2){//noktalar arasındaki uzaklığın bulunması
    return sqrt((d1.x-d2.x)*(d1.x-d2.x)+(d1.y-d2.y)*(d1.y-d2.y));
}

void printSpace(Dot *space, int n){//Noktaların x ve y değerlerini yazdırma
    int i;
    for(i=0; i<n; i++){
        printf("%d %d\n", space[i].x, space[i].y);
    }
    printf("\n");
}

```

```
}
```

float BForce(Dot *space,int start,int end){//3 e esit veya daha az nkta için brute force yaklasımı kullanılarak en küçük mesafe dondurulmaktadır.

```
int i;//iterasyon amaçlı kullanılmıştır.
```

```
int j;//iterasyon amaçlı kullanılmıştır.
```

```
float min=INF;//minimum uzaklığı bulmak için kullanılmıştır.
```

```
for(i=start;i<end;i++)//Aralıktaki tüm d leri karşılaştırma
```

```
    for(j=i+1;j<=end;j++)
```

```
        if(findDist(space[i],space[j]) < min)
```

```
            min=findDist(space[i],space[j]);
```

```
    return min;
```

```
}
```

float func(Dot *space,int l,int r){//noktalar arasındaki en küçük mesafeyi döndürür.

```
    if(r-l<3){
```

```
        return BForce(space,l,r);
```

```
    }
```

```
    int m=(l+r)/2;//sıralanmış düzlemde medya degeri orta noktadır.
```

```
    float dl=func(space,l,m);//medyanın solundaki en küçük d değerini bulma.
```

```
    float dr=func(space,m,r);//medyanın sağındaki en küçük d değerini bulma.
```

```
    float dm=MIN(dl,dr);//minimum d yi bulmak için kullanılmıştır.
```

```
    int i;//space dizisi için iterasyon amaçlı kullanılmıştır.
```

```
    int j=0;//range dizisi için iterasyon amaçlı kullanılmıştır.
```

```
    int lenRange;//range dizisinin boyutunu tutmak için kullanılmıştır.
```

```
    Dot *range=(Dot*)malloc(sizeof(Dot)*(r-l+1));
```

```
    for(i=l;i<=r;i++)//medyan degerinden d uzaklıktaki noktaların belirlenmesi
```

```
        if((space[i].x > space[m].x-dm) && (space[i].x<space[m].x+dm)){
```

```
            range[j++]=space[i];
```

```

    }

    lenRange=j;

    for(i=0;i<lenRange-1;i++){//bu aralıkta eğer d degerinden daha küçük bir deger
varsa dm guncellenir

        for(j=i+1;j<lenRange;j++){

            if(abs(range[i].y - range[j].y) < dm && findDist(range[i],range[j]) <
dm){

                //eger bakılan noktalar arasındaki y dm degerinden buyukse
onların arasındaki d > dm olur.

                dm=findDist(range[i],range[j]);

            }

        }

    }

    printf("dl=%.2f dr=%.2f dm=%.2f l=%d r=%d\n",dl,dr,dm,l,r);

    return dm;

}

```

void printAllDist(Dot *space,int n){//noktaların birbirleriyle olan uzaklıklarının yazdırılması

```

    int i;//iterasyon amaçlı

    int j;//iterasyon amaçlı

    printf(" \t");

    for(i=0;i<n;i++)printf("%d \t",i);

    printf("\n");

    for(i=0;i<n;i++){

        printf("%d\t",i);

        for(j=0;j<n;j++){

            printf("%.2f\t",findDist(space[i],space[j]));

        }

    }

```



```

        printf("\n");
    }
}

int main(){
    FILE *fp;//dosya okuma
    char *filename="input.txt";
    fp=fopen(filename,"r");
    if(fp == NULL){
        printf("Not Found File.\n");
        return 0;
    }

    int i;//space dizisi için iterasyon amaçlı kullanılmıştır.
    int n;//kartezyen düzlemdeki nokta sayısını tutmak için kullanılmıştır.
    Dot space[MAX];//dosyadaki noktaların saklanması için kullanılmıştır.
    while(!feof(fp)){//dosya okuma
        fscanf(fp,"%d %d",&space[i].x,&space[i].y);
        i++;
    }

    n=i-1;
    fclose(fp);
    qSort(space,0,n-1);
    printSpace(space,n);
    printAllDist(space,n);
    printf("%.2f",func(space,0,n-1));
    return 0;
}

```