T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

ELEKTRİK ELEKTRONİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

DERSİN ADI

Algoritma Analizi

BLM3021 GRUP 2

ÖDEV 1

DERSİ VEREN ÖĞRETİM ÜYESİ

Doç. Dr. M. Elif KARSLIGİL

ÖDEVİ YAPAN ÖĞRENCİ

İBRAHİM TIPIRDIK

16011088

ÖDEV TESLİM TARİHİ

29.10.2019

KONU : Closest Pair Problem

Problem : Bu ödevde kartezyen düzleminde verilen n adet nokta arasında birbirine en yakın 2 noktanın Divide-and-Conquer yaklaşımı ile bulunması istenmektedir.

ÇÖZÜM

A) Giriş bilgisi okunmalı ve gelen sayıların kendi oluşturduğumuz bir structa kaydedilmesi gerekmektedir.

B) Öncelikle noktalar x koordinatına göre küçükten büyüğe sıralanmalıdır. (Quick Sort kullanıldı)

C) Medyan değeri bulunarak noktalar iki parçaya ayrılmalıdır.

D) İki bölgede de en kısa mesafeler bulunmalıdır. Bunun için n<=3 olana kadar ikiye bölerek ilerleriz. n<=3 durumuna ulaştığımızda brute-force yöntemi ile noktalar arasında mesafesi en kısa olan ikili bulunur. Buradan dönen değer ile diğer üçlüden dönen değer karşılaştırılıp küçük olan indisler ve mesafe döndürülür ve en kısa mesafe iki bölge için ayrı ayrı bulur(d).

E) Son adımda ise (d) uzaklığı bulunmasına rağmen ikililer her zaman aynı bölgede olmak zorunda değildir. Bunun için medyandan en fazla d kadar uzaklıkta olan noktalar kendi aralarında kontrol edilir. Önceki d uzaklığı ile karşılaştırılır. Daha kısa mesafe bulunmuşsa yeni indislerimiz ve d değerimiz bulunmuş olur.

KARMAŞIKLIK

Kullandığımız sıralama algoritması Quick Sort

Worst Case=O(n^2) Best Case=O(n(logn)) Average Case=(n(logn))

Recursive olarak 2 ye bölerek ilerliyoruz logn

N<=3 olduğunda brute-force ‘a girip n işlem yapıyor n

Karmaşıklık = O(nlogn)

UYGULAMA

```
Dizinin siralanmis hali
0. nokta x == 2 , y == 3
1. nokta x == 3 , y == 11
2. nokta x == 4 , y == 20
3. nokta x == 7 , y == 12
4. nokta x == 8 , y == 1

Soldaki en kisa mesafe 1. nokta x=2 y=3   2. nokta x=3 y=11    uzaklik=8.062258
Sagdaki en kisa mesafe 1. nokta x=4 y=20   2. nokta x=7 y=12    uzaklik=8.544003
En kisa mesafeli 1.nokta x=3 y=11   2. nokta x=7 y=12    uzaklik=4.123106
--------------------------------
Process exited after 4.265 seconds with return value 0
Press any key to continue . . .
```

```
Dizinin siralanmis hali
0. nokta x == -4 , y == -9
1. nokta x == -3 , y == 12
2. nokta x == -1 , y == 12
3. nokta x == 0 , y == -4
4. nokta x == 6 , y == -3
5. nokta x == 8 , y == 0
6. nokta x == 10 , y == 2

Soldaki en kisa mesafe 1. nokta x=-3 y=12   2. nokta x=-1 y=12    uzaklik=2.000000
Sagdaki en kisa mesafe 1. nokta x=-1 y=12   2. nokta x=0 y=-4    uzaklik=16.031219

Soldaki en kisa mesafe 1. nokta x=6 y=-3   2. nokta x=8 y=0    uzaklik=3.605551
Sagdaki en kisa mesafe 1. nokta x=8 y=0   2. nokta x=10 y=2    uzaklik=2.828427

Soldaki en kisa mesafe 1. nokta x=-3 y=12   2. nokta x=-1 y=12    uzaklik=2.000000
Sagdaki en kisa mesafe 1. nokta x=8 y=0   2. nokta x=10 y=2    uzaklik=2.828427

En kisa mesafeli 1.nokta x=-3 y=12   2. nokta x=-1 y=12    uzaklik=2.000000
--------------------------------
```

```
Dizinin siralanmis hali
0. nokta x == -6 , y == 4
1. nokta x == -3 , y == 9
2. nokta x == -2 , y == 8
3. nokta x == -1 , y == 7
4. nokta x == 1 , y == 2
5. nokta x == 2 , y == 3
6. nokta x == 3 , y == 6
7. nokta x == 7 , y == 8

Soldaki en kisa mesafe 1. nokta x=-3 y=9   2. nokta x=-2 y=8    uzaklik=1.414214
Sagdaki en kisa mesafe 1. nokta x=-2 y=8   2. nokta x=-1 y=7    uzaklik=1.414214

Soldaki en kisa mesafe 1. nokta x=1 y=2   2. nokta x=2 y=3    uzaklik=1.414214
Sagdaki en kisa mesafe 1. nokta x=3 y=6   2. nokta x=7 y=8    uzaklik=4.472136

Soldaki en kisa mesafe 1. nokta x=-2 y=8   2. nokta x=-1 y=7    uzaklik=1.414214
Sagdaki en kisa mesafe 1. nokta x=1 y=2   2. nokta x=2 y=3    uzaklik=1.414214

En kisa mesafeli 1.nokta x=1 y=2   2. nokta x=2 y=3    uzaklik=1.414214
--------------------------------
```

```
Dizinin siralanmis hali
0. nokta x == 2 , y == 3
1. nokta x == 3 , y == 7
2. nokta x == 4 , y == 6
3. nokta x == 5 , y == 1
4. nokta x == 7 , y == 12
5. nokta x == 8 , y == 10

Soldaki en kisa mesafe 1. nokta x=3 y=7   2. nokta x=4 y=6    uzaklik=1.414214
Sagdaki en kisa mesafe 1. nokta x=4 y=6   2. nokta x=5 y=1    uzaklik=5.099020

Soldaki en kisa mesafe 1. nokta x=3 y=7   2. nokta x=4 y=6    uzaklik=1.414214
Sagdaki en kisa mesafe 1. nokta x=7 y=12   2. nokta x=8 y=10    uzaklik=2.236068

En kisa mesafeli 1.nokta x=3 y=7   2. nokta x=4 y=6    uzaklik=1.414214
---------------------------------
```

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
struct n{
	int x;
	int y;
};
typedef struct n nokta;
void noktaYazdir(nokta* noktalar,int n){                                    //noktaları yazdırıyoruz
	int i;
	for(i=0;i<n;i++){
		printf("%d. nokta x == %d , y == %d\n",i,noktalar[i].x,noktalar[i].y);
	}
}
FILE* dosyaAc(char* isim){                                                  //dosya açma
işleminin yapıldığı yer
	FILE* dosya;
	dosya=fopen(isim,"r");
	if(dosya==NULL){
		printf("dosya acilamadi\n");
	}else{
		printf("dosya acildi\n");
		return dosya;
	}
}
void sifirla(char *dizi,int n){                                             //textten okurken
kullandığımız diziyi sıfırladığımız fonksiyon
	int i;
	for(i=0;dizi[i]=!'\0' && i<n;i++)
		dizi[i]='\0';
}
nokta duzelt(char *dizi,int n){
```

```
	nokta nokta1;				//okuyacağımız noktaları koordinat değerleri halinde tuttuğumuz değişken
	int i;					//indis
	int sayi1=0;				//okuyacağımız birinci sayı
	int sayi2=0;				//okuyacağımzı 2. sayı
	int negatif=0;				//değerin negatifligini kontrol ediyoruz
	for(i=0;dizi[i]!=' ' ;i++){				//ilk sayının oluşturulduğu yer
		sayi1*=10;
		if(dizi[i] == '-')
			negatif=1;
	//negatif sayi kontrolu
		else
		sayi1+=(dizi[i]-'0');
	}
	if(negatif)
		sayi1*=-1;
	negatif=0;
	i++;
	for(i;dizi[i]!=' ' && dizi[i]!='\0' && dizi[i]!='\n';i++){		//ikinci sayının oluşturulması
		sayi2*=10;
		if(dizi[i] == '-')
			negatif=1;
		else
			sayi2+=(dizi[i]-'0');
	}
	if(negatif)
		sayi2*=-1;
	nokta1.x=sayi1;
	nokta1.y=sayi2;				//noktayı oluşrutuyoruz
	return nokta1;				//noktayı geri döndürüyoruz
}
```

```
nokta* noktalariOku(FILE *dosya,int *n,nokta *noktalar){                    //dosyadan sayıları okuyoruz
        char dizi[10];
                //dosyadan okuyacağımızı yazdığımız char dizisi
        int iter=0;
        while(!(feof(dosya))){
                fgets(dizi,10,dosya);
        //bir kez dosyada kac eleman olduğunu bulmak icin okuyoruz
                iter++;
        }
        noktalar=(nokta*)malloc(sizeof(nokta)*iter);                    //iter kadar nokta var
iterlik yer açtık
        fseek(dosya,0,SEEK_SET);
        while(!(feof(dosya))){
                sifirla(dizi,10);
        //diziyi sıfırlıyoruz
                fgets(dizi,10,dosya);
        //satır satır en fazla 10 karakter okuyacak şekilde okuyoruz
                noktalar[(*n)++]=duzelt(&dizi[0],10);                   //duzeltten
gelen noktayı noktalar dizisine ekliyoruz
        }
        fclose(dosya);
                //dosyayı kapatıyoruz
        return noktalar;
}
void noktaSirala(nokta* noktalar,int ilk,int son){          //quick sort
        int sol=ilk+1;
        int sag=son;
        int temp;
        if(ilk < son){
                int pivot=ilk;
                while(sol <= sag){
                        if(noktalar[sol].x > noktalar[pivot].x && noktalar[sag].x < noktalar[pivot].x){
                                temp=noktalar[sol].x;
                                noktalar[sol].x=noktalar[sag].x;
```

```
                noktalar[sag].x=temp;
                temp=noktalar[sol].y;
                noktalar[sol].y=noktalar[sag].y;
                noktalar[sag].y=temp;
                sol++;
                sag--;
            }
            if(noktalar[sol].x <= noktalar[pivot].x || noktalar[sag].x >= noktalar[pivot].x){
                if(noktalar[sol].x <= noktalar[pivot].x)
                    sol++;
                if(noktalar[sag].x >= noktalar[pivot].x)
                    sag--;
            }
        }
        temp=noktalar[sol-1].x;
        noktalar[sol-1].x=noktalar[pivot].x;
        noktalar[pivot].x=temp;
        temp=noktalar[sol-1].y;
        noktalar[sol-1].y=noktalar[pivot].y;
        noktalar[pivot].y=temp;
        noktaSirala(noktalar,ilk,sol-2);
        noktaSirala(noktalar,sol,son);
    }
}
int medyanBul(int ilkElemanIndis,int sonElemanIndis){          // medyan=n/2
    return (ilkElemanIndis+sonElemanIndis+1)/2;
}
float uzaklikHesapla(nokta* noktalar,int indis1,int indis2){          // Karekok((x1-x2)^2+(y1-
y2)^2) formülü ile uzakligi hesaplıyoruz
    return sqrt((noktalar[indis1].x-noktalar[indis2].x)*(noktalar[indis1].x-
noktalar[indis2].x)+(noktalar[indis1].y-noktalar[indis2].y)*(noktalar[indis1].y-noktalar[indis2].y));
}
```

```
float* brutForce(nokta* noktalar,int ilkElemanIndis,int sonElemanIndis){          //3 ve 3 ten küçük sayıda noktayı brut olarak hesaplıyoruz

        int size=sonElemanIndis-ilkElemanIndis+1;
        //dizinin size ını hesapladığımız kısım

        int i;
                                //indisler

        int j;

        float dist;
                                //uzaklik değişkeni

        float* min=(float*)malloc(sizeof(float)*3);
        //min[0] = ilk nokta indisi     min[1]=ikinci nokta indisi

        min[2]=1000.0;
                        //min[2] = aralarındaki uzaklık

        for(i=ilkElemanIndis;i<size+ilkElemanIndis-1;i++){

                for(j=i+1;j<size+ilkElemanIndis;j++){

                        dist=uzaklikHesapla(noktalar,i,j);
        //iki nokta arasındaki uzaklik hesaplaması

                        if(min[2] > dist){
                        //eğer min uzaklık bulunan uzakliktan büyükse değiştriyoruz

                                min[0]=i;

                                min[1]=j;

                                min[2]=dist;

                        }

                }

        }

        return  min;

}

float* enYakinNoktalar(nokta* noktalar,int ilkElemanIndis,int sonElemanIndis){

        int size=sonElemanIndis-ilkElemanIndis+1;                       //size hesabı

        if(size <= 3){

                return brutForce(noktalar,ilkElemanIndis,sonElemanIndis);                       //size <= 3 durumunda brutforce yapıyoruz

        }else{

                float* noktalar1=(float*)malloc(sizeof(float)*3);
        //noktaların indislerini tuttuğumuz kısım
```

```
		float* noktalar2=(float*)malloc(sizeof(float)*3);				//min
in aynısı


	noktalar1=enYakinNoktalar(noktalar,ilkElemanIndis,medyanBul(ilkElemanIndis,sonElemanInd
is));	// ikiye bölme kısmı


	noktalar2=enYakinNoktalar(noktalar,medyanBul(ilkElemanIndis,sonElemanIndis),sonElemanI
ndis);

		printf("Soldaki en kisa mesafe 1. nokta x=%d y=%d   2. nokta x=%d y=%d
uzaklik=%f\n",noktalar[(int)noktalar1[0]].x,noktalar[(int)noktalar1[0]].y,noktalar[(int)noktalar1[1]].x,
noktalar[(int)noktalar1[1]].y,noktalar1[2]);

		printf("Sagdaki en kisa mesafe 1. nokta x=%d y=%d   2. nokta x=%d y=%d
uzaklik=%f\n\n",noktalar[(int)noktalar2[0]].x,noktalar[(int)noktalar2[0]].y,noktalar[(int)noktalar2[1]].
x,noktalar[(int)noktalar2[1]].y,noktalar2[2]);

		if(noktalar1[2] < noktalar2[2])

			return noktalar1;

		else
				//daha kısa uzunlukta olan noktaları döndürüyoruz

			return noktalar2;

	}

}

float* bulDikdortgen(nokta* noktalar,float* min,int n){

	int i,j;
			//indisler

	int k=-1;
				//dikdörtgendeki noktaları tutacak dizi indisi

	int indisler[20];
		//dikdörtgendeki noktaları tutacak dizi

	int medyan=medyanBul(0,n-1);
	//medyan

	float dist;
				//uzaklık değişkeni

	for(i=0;i<n-1;i++){
			//burada dikdörtgenin içinde olan noktaları buluyoruz


		if(noktalar[i].x < noktalar[medyan].x+min[2] && noktalar[i].x > noktalar[medyan].x-
min[2]){

			indisler[++k]=i;
```

```
		}
	}
	for(i=0;i<k;i++){
		for(j=i+1;j<=k;j++){ //dikdörtgende daha kısa mesafeli noktalar var mı kontrol ediyoruz
			dist=uzaklikHesapla(noktalar,indisler[i],indisler[j]);
			if(min[2] > dist){
				min[0]=i;
				min[1]=j;
	//varsa yeni en yakın noktalarımız bu oluyor
				min[2]=dist;
			}
		}
	}
	return min;
}
int main(){
	nokta *noktalar;
	float *min=(float*)malloc(sizeof(float)*3);
	char dosyaAdi[20];
	//dosya adi
	FILE *dosya;
	int n=0;
	//eleman sayisi
	printf("Acmak istediginiz dosyanin adini giriniz");
	scanf("%s",&dosyaAdi);
	dosya=dosyaAc(&dosyaAdi[0]);
	noktalar=noktalariOku(dosya,&n,noktalar);
	noktaSirala(noktalar,0,n-1);
	printf("\nDizinin siralanmis hali\n");
	noktaYazdir(noktalar,n);
	printf("\n");
```

```
	min=enYakinNoktalar(noktalar,0,n-1);				//recursive olarak en yakın noktaları hesaplıyoruz

	min=bulDikdortgen(noktalar,min,n);

	int koordinat1=(int)min[0];					//float değeri int e çeviriyoruz

	int koordinat2=(int)min[1];

	printf("En kisa mesafeli 1.nokta x=%d y=%d   2. nokta x=%d y=%d uzaklik=%f",noktalar[koordinat1].x,noktalar[koordinat1].y,noktalar[koordinat2].x,noktalar[koordinat2].y,min[2]);


	free(min);

	free(noktalar);

	return 0;

}
```