HAZIRLAYAN: YUNUS KARATEPE, 17011051

DERS: ALGORİTMA ANALİZİ, BLM3021

GRUP: 1 – MİNE ELİF KARSLIGİL

ÖDEV KONUSU: DIVIDE AND CONQUER ALGORITHMS

YÖNTEM:

Problemin çözümünde noktanın x ve y koordinatlarına sahip olan bir struct yapısı kullanıldı :

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

Kullanılan fonksiyonlara göz atmak gerekirse :

1. DOT* createSpace(int);
2. void sortBy_x(int, DOT*);
3. float distance(DOT, DOT);
4. float bruteForce(DOT*, int, DOT[]);
5. float closest_Pair(DOT*, int, DOT[]);
6. void displaySpace(int, DOT*);

şeklinde 6 adet fonksiyon kullanıldı.

Fonksiyon ve Algoritma Açıklamaları:

1) createSpace() fonksiyonu ile kullanıcıdan alınan bir sayı kadar nokta girilmesi istendi.
2) sortBy_x() fonksiyonunda ise kullanıcının girdiği noktalar uzayı x koordinatlarına göre sıralandı.
3) distance() fonksiyonu iki nokta arası uzaklığı bulma işlevini gerçekleştirdi.
4) bruteFotce() fonksiyonu 3 veya daha az noktalar için noktalar arasında en yakın olanları ve bunlar arası mesafeyi bulmada kullanıldı. Dönüş değeri uzaklık değerini, parametre olarak verilen DOT[2] ise en yakın ikiliyi tutmuş oldu.
5) closestPair() fonksiyonunda ise kodun asıl algoritması olan divide and conquer işlemini gerçekleştirerek, recursive bir şekilde diğer fonksiyonların da yardımı ile en yakın ikiliyi buldu.

    Bunu yaparken, noktalar uzayını, uzayın boyutunu, en yakın ikiliyi parametre olarak alıyor.

1) Min değerine önce 9999 atıyor. Bunun nedeni eğer 1 nokta kalır ise en küçük değerinin bozulmaması.
2) Eğer nokta sayısı 3'ten büyük ise bir medyan = noktaSayısı / 2 belirliyor. min_left ve min_right tanımlayıp bunları 9999'a eşitliyoruz, aynı zamanda soldaki ve sağdaki en yakın ikilileri tutmak için closestPair_Left[2] ve closestPair_Right[2] isimli birer çift belirliyoruz.
3) Sonrasında recursive işlemi başlıyor ve nokta uzayımızın önce medyanının sol tarafı sonra sağ tarafı için (medyan sağ tarafta kalacak şekilde) çağırıyoruz. Sol tarafın minimum değerini, min_left'e eşitliyoruz sağ tarafınkini ise min_righte. Aynı zamanda closestPair_Left/Right [2] değerlerini de doldurmuş oluyoruz.

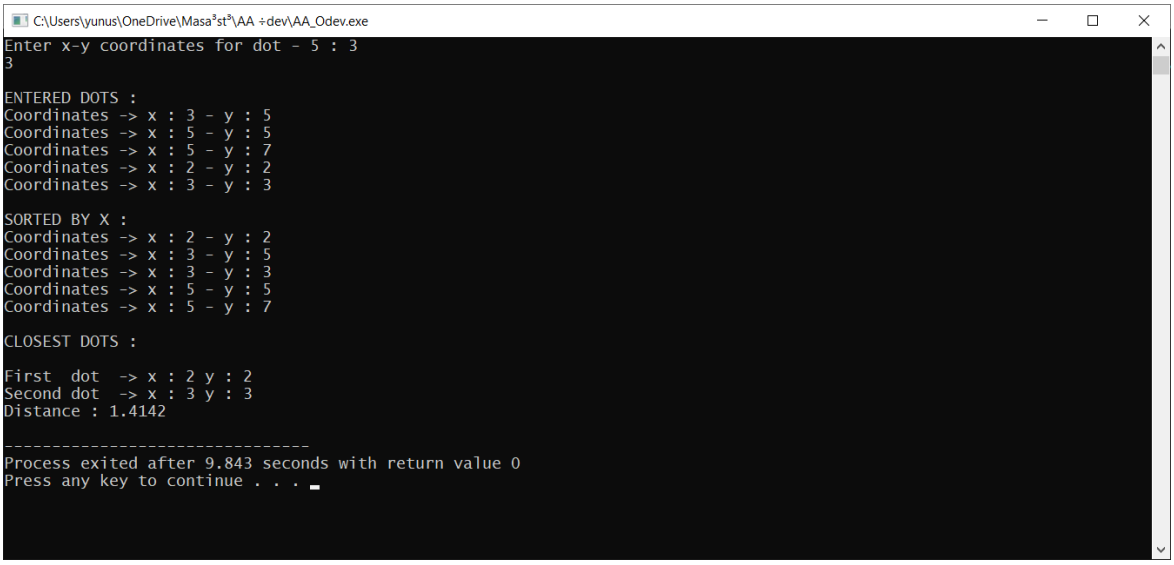
4) Daha sonra bunlardan en yakın olanı (left ve right arasından) bulup min ve closestPair[2] ikilisini o yapıyoruz.
5) Bundan sonra yapacak tek bir şey kalıyor. Solun en yakınını bulduk, aynı şekilde sağın da en yakın çiftini bulduk. Tek sorun en yakın olan ikilinin biri solda, diğerinin sağda kalmış olması. Bunun için de o anki medyan değerine göre sol tarafı ve sağ tarafı (en fazla min değeri kadar medyana uzak olanlar) leftDots[] ve rightDots[] dizilerine alıyoruz.
6) Sonra bunlar arasında bruteForce yaparak (sadece soldakiler ile sağdakiler karşılaştırılıyor) en yakından yani bizim o anki min değerimizden daha küçük bir değer geliyor mu diye kontrol ediyoruz.
7) Eğer böyle bir uzaklık var ise min değerini o uzaklık ile değiştirip en yakın ikiliyi de değiştiriyoruz.
8) Tüm bu işlemleri bitirdikten sonra elimizdeki min değerini döndürüyoruz.
9) Eğer noktaların sayısı 3 ten büyük değilse ve 1 den büyük ise onların içindeki min değerini bruteForce() yaparak bulup verilen en yakın ikili değerini değiştirip min değerini de döndürüyoruz.

Böylece recursive bir yapı ile hatasız bir şekilde her durum için minimum değerini bulmuş oluyoruz.

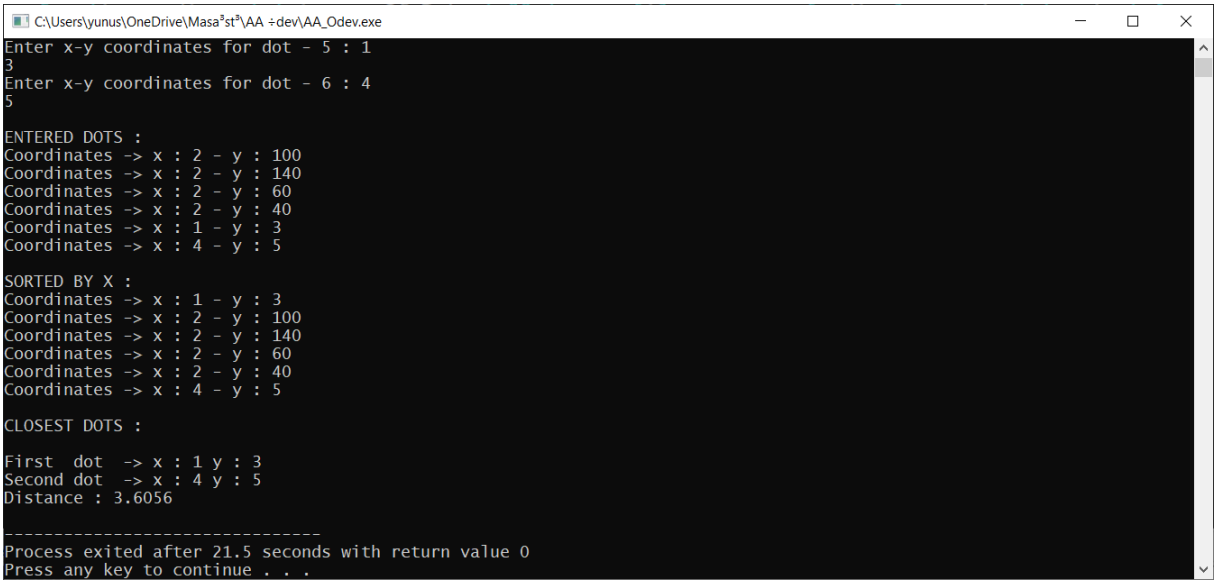
Algoritmanın karmaşıklığına gelecek olursak:

n adet girilen eleman olsun. Bizim yaptığımız ile her adımda (n' > 3 olduğu sürece) eleman sayısını ikiye bölerek ilerleyeceğiz. n' < 3 olduğu durumda ise her ne kadar brute force yapmış olsak da karmaşıklığımız sabit bir sayı olmuş olacak (3'e bağlı olduğu için). Yani bizim n'e bağlı karmaşıklığımız her adımda 2 ye bölmemizden kaynaklanacak. Böylece karmaşıklığımız c * log2(n) olmuş olacak. Bizim durumumuzda ise c değeri değişken olmak ile beraber brute forceden dolayı c = 3^2 diyebiliriz. Yani karmaşıklık 9 * log2(n) olmuş olacak. Bu da O(log2(n)) e eşit olacaktır.

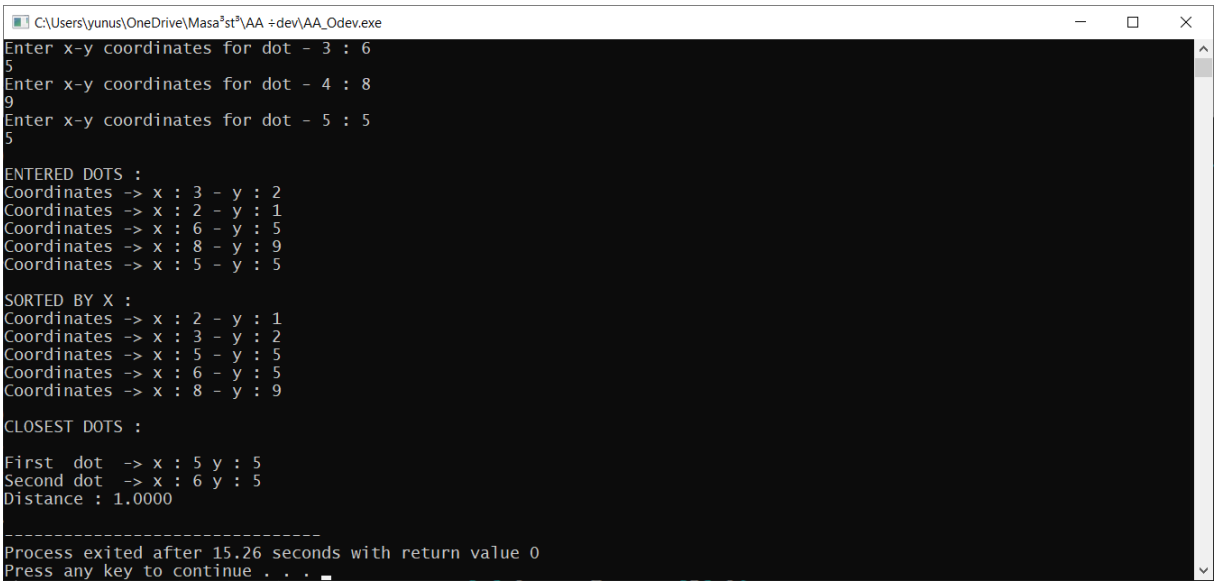
***UYGULAMA*:**



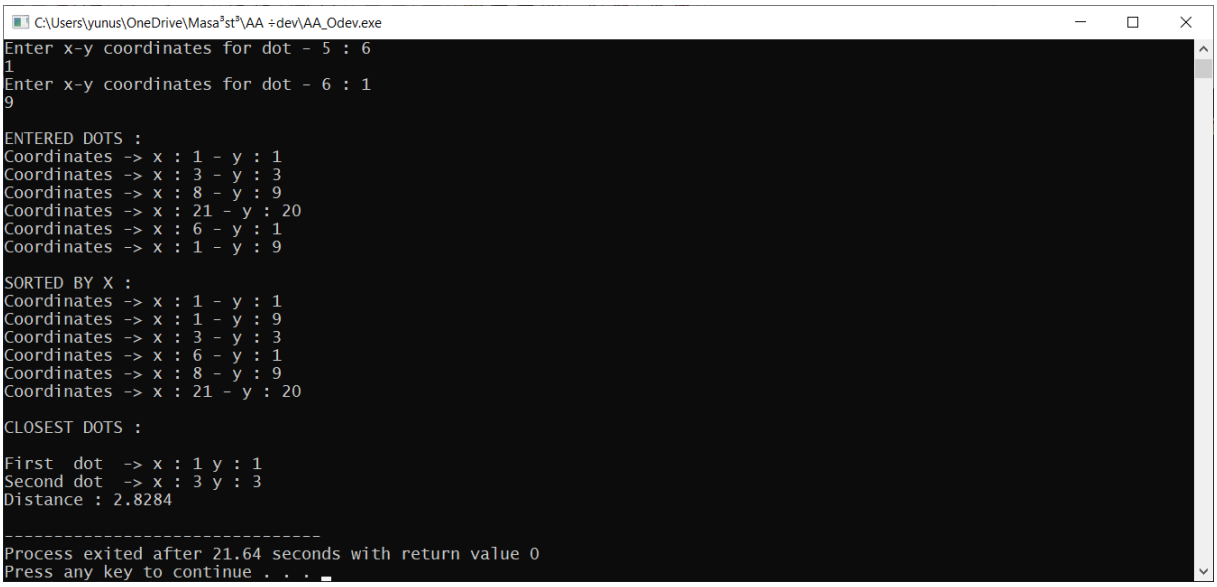
Burada görüldüğü üzere en yakın olan noktalardan 1. si sol tarafta diğeri ise sağ tarafta (medyan sağ tarafta) bulunmaktadır. Program doğru sonucu vermektedir.



Bu örnekte de 1-3 noktası sol tarafta 4-5 noktası ise sağ taraftadır.



Bu örnekte ise noktaların ikisi de sağ taraftadır.



Bu örnekte de verilen noktalardan en yakın olan ikili sol taraftadır.