ALGORİTMA ANALİZİ BİRİNCİ ÖDEV RAPORU

YÖNTEM BÖLÜMÜ

Problem şu ki,elimizde n adet nokta çifti var ve bunlar arasından birbirine en yakın olan noktayı bulmak istiyoruz.

Çözüm için Closest-Pair algoritmasın kullanabiliriz.Closest-Pair algoritmasına göre problemi küçük parçalara bölüp o parçalar için elde ettiğimiz çözümleri asıl problem için birleştirerek bulmak.

N sayıdan 3 tane varsa brute-force ile çözülür.Daha fazlaysa closest-pair ile çözülür.

Benim çözümümde kullanıcıdan kaç adet nokta olduğu bilgisi alınır daha sonra Input.txt dosyasından noktalar okunur.Bunun için fopen komutunu kullandım.Daha sonra bu noktaları bir integer dizisine attım.Struct kullanılamıyordu tek tek okuduğum için.Daha sonra indisine göre çiftse o anki x koordinatına,tek ise o anki y koordinatına atadım.Daha sonra "okumuş mu?" kontrolünü yapmak için koordinatDuzlemi adlı diziyi yazdırdım.Asıl ana kısmına gelirsek (n==3||n==2) komutuyla eleman sayısı 3 veya 2 ise brute force algoritmasını uyguladım "brute_force" fonksiyonunda noktalar arasındaki uzaklık hesaplanır ve uzaklık ekrana yazılır.

Eğer nokta sayısı 3'den fazla ise ,bu durumda closest_pair ile çözülenmesi gerekir.Ayrıca bu recursive ve azalan bir yapı olduğu için mesela 5 nokta için closest_pair ile hesaplarken nokta sayısı 3 e düştüğünde brute_force ile çözüm aranacaktır.

2 den küçük noktalar için programın hata mesaji vermesi beklenir.

FOKSİYONLARIN AÇIKLAMASI

minumum_uzaklik : dr ve dl noktalarından gelen sonuçlardan küçük olan sonucu döndüren fonksiyon.

y_sirala : X koordinatlarını sıraladıktan sonra aynı X değerine sahip olan noktaların y koordinatlarına göre de sıralanmasını istediğim fonksiyon.Mesela (1,1)(1,5)(1,4) diye x e göre sıraladık bende (1,1)(1,4)(1,5) olarak sıralaması için bu fonksiyonu yazdım.

oklid : Gelen 2 struct noktanın birbirine olan uzaklığını bulan fonksiyon.

quickSort : Noktaların x koordinatının sıralanması için yazdığım quickSort sıralama algoritması.Quick Sort'da closest pair gibi ikiye bölerek çözüm üretmeye çalışır.Batan ortaya bir sıralama,ortadan başa bir divide yapar.

brute_force : Karşılaştırılacak 3 veya 3 den az nokta kaldığında bu noktalardan hangisinin birbirine yakın olduğunu bulan ve en yakın olan iki noktanın uzaklıklarını return eden fonksiyondur.

closest_pair : Yine yukarıda belirttiğim gibi 3 ve 3 den daha az nokta olduğunda ilk if de brute_force'yi çağırdım. Else'inde ise medyan değeri olan orta noktayı belirleyip , daha sonra dl ve dr için fonksiyonu çağırdım. Dltmp 'yi dl ye göre güncelledim.Drtmp yi ise dr ye göre güncelledim. Daha sonra dltmp ve drtmp'yi en az uzaklığı bulması için minumum_uzaklik'a gönderdim.Daha sonra orta şerit olan çizgiye d den daha yakın olan noktaları serit dizisine attım.

en_yakin_nokta : Hesaplanan üst parça ve alt parçadan karşılaştırma yapıp,hangisinin en küçük olduğuna karar veren ve bunu return eden fonksiyondur.

ALGORİTMANIN KARMAŞIKLIĞI

Algoritmamızın zaman karmaşıklığı T(n) olsun.Quick Sort' un zaman karmaşıklığı ;

T(1)=0 kabul edersek, best case =T(n)=(n*lg2^n) n>1 olduğu durumda best case i bu şekildedir.

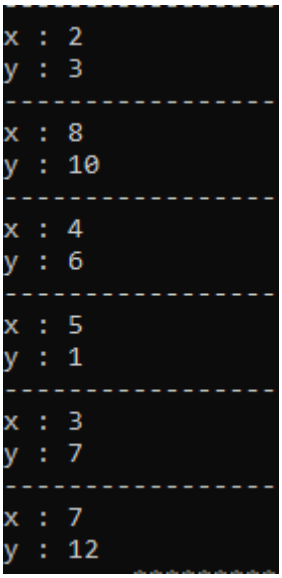
Algoritma ikiye bölerek ilerliyor.Bu durumda lg tabanında bir değeri vardır.Şerit iki bölgeye bölüyor.

Şeride en yakın noktayı bulma O(n) zaman alır.Buna göre T(n)=2T(n/2)+O(n) olur 2 parçamız var ve ikiside yarım o yüzden 2T(n/2) olur. Sonuç olarak;

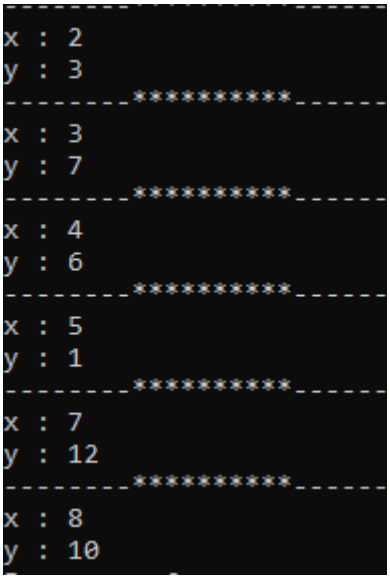
T(n)=T(nLogn) olarak bulunur.

UYGULAMA BÖLÜMÜ

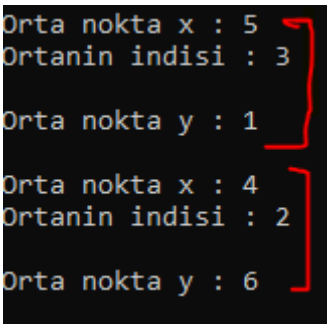
1)Dosyadan okunan noktalar ;



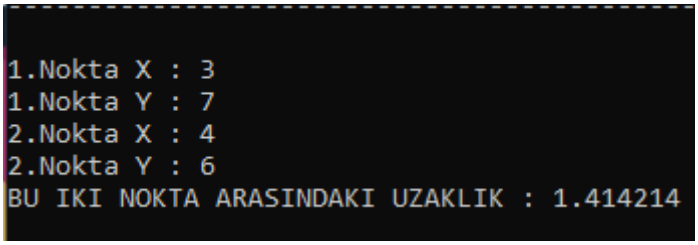
2)Noktaların sıralanmış hali;



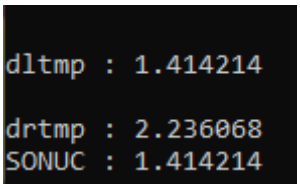
3)Seçilen orta noktalar;



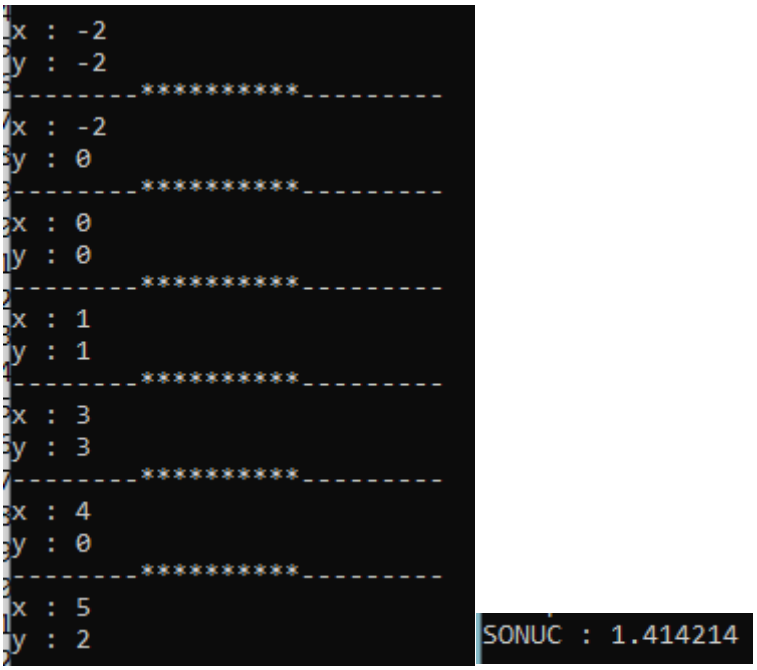
4)Her bir nokta çifti için bu şekilde uzaklık hesaplanır;



5)En son kalan iki nokta hesaplanarak sonuç bulunur;



Örnek



Örnek

```
x : 1
y : 0
--------**********---------
x : 1
y : 5
--------**********---------
x : 1
y : 9
--------**********---------
x : 2
y : 4
--------**********---------
x : 3
y : 7
SONUC : 3.162278
```

Örnek

```
x : 4
y : 8
--------**********---------
x : 5
y : 6
--------**********---------
x : 7
y : 6
--------**********---------
x : 15
y : 2
--------**********---------
x : 20
y : 22
SONUC : 2.000000
```