İlk önce kısaca classların detaylarını anlatayım.

SkylineData Class'ı:

İçinde okunan dosyaların tutulduğu 2d bir array list(Sütun sayısı daimi 3 oluyor)

Köşelerin hesaplanıp tutulduğu XofPoint ve YofPoint arrayListleri

Ve Skyline'ların en yüksek ve en sağ taraftaki noktasının tutulduğu MaxHeight and MaxWidth

Ve Bunları hesaplayan metodlar(Altta detaylı anlattım).

Files Class'ı

Dosyayı açtığım tek bir metod

Dosyadaki her satırı tek boyutlu bir array liste alıp, SkylineData objesindeki 2d array liste ekliyor.

DrawLines Class'ı

JPanelden extend ediyor. İçinde X ve Y noktalarının tutulduğu arrayListler var.

DrawLinesWithData'ya gelen arraylistleri X ve Y arraylistlerine atıyor,gelen Height ve Width'e göre pencerenin boyutlarını belirliyor.

Çizgilerin gözükmesi için koordinatların 20 katını alıyorum.

Çizgilerin kalın Gözükmesi için +1 -1 (1/20 boyutunda şekli bozabilir fakat küçük bir oran) şeklinde 3 çizgi çiziyorum

Main classımda sadece main metodu bulunuyor ve sırayla çağırdığı metodlarla kısaca

*Yeni bir SkylineData objesi oluşturuyorum, ve data.txt'teki binaları objeye ekliyorum

* Sonra objedeki binaları location'larına göre sıralıyorum,Köşelerini bulup köşelerini ekrana basıyorum
* Daha sonra Ekrana çizdirmek için bir DrawLines objesi oluşturuyorum,
* İlk oluşturduğum SkylineData objesindeki yükseklik,genişlik ve köşelerin koordinatlarını
* DrawLinesWithData metoduna atıp ekrana çizgileri çizdiriyorum.

SkylineData Classı Detaylar

İlk satırlarda kullanmam gereken setter ve getterler var.

SetXandYofPoint(int i,int j) metodu

Koordinat tuttuğum 2 Ayrı liste, beraber atama yapması için yazdığım bir metod. Tam anlamıyla bir setter değil.

SortSkyData() metodu

Comparator ve lambdalar yardımıyla Datamızın ilk halini(dosyadan okunan) Listteki 2.sütuna(location) göre sıralıyor.

Burdaki sort kodunu Collections.sort(list,comparator..) şeklinde kullanmıştım. Kullandığım IDE bu halde aynı işlevi yaptığını söyledi. Öğrenilmesi daha kolay olduğu için list.sort(Comparator..) şeklinde kullandım.

FindSkyCorners() metodu

Kısaca:

İçinde tuttuğu maxHeight değişkeni gezdiği locationdaki en yüksek noktayı, tempHeight ise bir önceki en yüksek noktayı tutuyor.

NOT: maxHeight!=MaxHeight

MaxHeight = tüm skylinedaki en yüksek nokta, class değişkeni

maxHeight = FindSkyCorners içindeki noktasal en yüksek nokta değişkeni.

Bunları kıyaslayarak ve locationu arttırarak Skylinedaki köşeleri buluyorum.

Detaylı:

İlk önce sıralanmış büyük arrayde gezerek location+widthin(yani en sağdaki noktanın) ne olduğu bulup MaxWidth'e atıyorum.

Daha sonra ilk elemanın location'undan MaxWidthe kadar bir döngüde;

Her noktanın, tüm binalara bakarak;

(BinanınLocation <= BulunduğumLocation <= BinanınLocation+Width)

Olması koşulunda(yani o noktanın bina içinde bulunması lazım)

Şeklinde o noktanın en yüksek noktasını kıyaslayarak buluyorum. Bulduktan sonra maxHeighte atıyorum. Döngü biterken tempHeighti maxHeighte eşitliyorum

Sonra döngü devam ettiğinde maxHeighti bulduğumda, eğer önceki tempHeightle eşit değilse o nokta bir kırılma noktası(köşe) oluyor.

Mesela 1,4 -- 2,4 -- 3,4 te maxHeight ve tempHeight 4 iken 4—11 noktasına geçersem Kırılma noktası olduğunu anlıyor ve 4,4 ve 4,11 noktalarını köşelere ekliyor

Döngü bitincede position değişkenin final değerine göre position,maxHeight ve position,0 noktalarınıda dataya ekliyor.

Kodda son dakika değiştirdiğim birkaç değişken hakkında açıklama;

tempY(yüksekliğin azaldığı durumda o bölgeyi içeren sıradaki bina)

ve tempmaxHeight(o noktadaki 2. en yüksek konum, bir nevi yükseklik azalınca ineceği noktanın yüksekliği)

Son dakika ödevde bulduğum 2-3 bugu düzeltmek için koyduğum yeni 2 değişken yukarıda belirttim.

Detayları;

Eğer binalar arasında tek birim fark varsa aşağıda kalan yüksekliği daima yok sayıyordu, bunun için positionu double olarak değiştirip positionu içeren binaların arasında yükseklik kıyasladım

örnek: “5.5” positionunu “1 1 5” şekilde datası olan bina içeriyor(5 ve 6 arasında).

Aradaki tek birim fark için 2.yüksekliği(eğer yoksa sıfır) olacak şekilde aldım ve kodumdaki yüksekliğin azaldığı duruma 2 adet araya 2.yüksekliği alan koordinat ekledim ve hatayı çözdüm.

Efficiency ve Time Complexity

TimeComplexity’i döngülerin çalışma süresine göre yorumluyorum.

Files Classı için;

Dosyada kaç satır varsa döngü o kadar dönecek. Tek döngü olduğu için liner ve çalışma süresi SatırSayısına(N) bağlıdır.

DrawLines Classı için;

Graphics metodları hakkında bilgim yok, yorum yapamam.

Döngümüz X array listinin, yani kaç koordinat bulduysam onun bir eksiği kadar dönüyor. Yani çalışma süresi linerdir ve bulunan koordinatların -1’ine (N-1) bağlıdır.

N=X arrayListinin size’ı;

SkylineData Classı için;

Getter ve setter metodları sabit zamanda çalışır.

SortSkyData metodu çalışma algoritması N*(log(N))

FindSkyCorners;

İlk döngümüz en sağ noktayı buluyor çalışma süresi SkyData’nın yani data.txt’in satır sayısı(N) kadar dönecek yani linerdir.

İkinci döngümüz genişlik(M) kadar, içindeki döngüde Satır Sayısı(SkyDatanın boyutu=N) dönecek. Döngünün çalışma süresi N*M’dir.

Tüm methodun çalışma süresi N*M ‘dir çünkü yukarıdaki döngü kısa süreceği için görmezden gelinebilir.

PrintCorners;

Bulduğumuz koordinatlar(N) kadar çalışır. Çalışma süresi linerdir yani N.