Problem Tanımı

Sosyal ağlar, birçok bireyin (düğümler) birbirleri ile kurduğu ilişkileri (arkadaşlık, takip, etkileşim gibi) içeren grafik yapılarıdır. Bu proje, verilen bir sosyal ağ grafiğindeki toplulukları tanımlamak için özel olarak tasarlanmış bir graf algoritması geliştirmeyi amaçlanmaktadır. Bu topluluklar, ağ içinde belirli bir ilişki türü veya diğer ortak özelliklere sahip bireylerin oluşturduğu gruplardır. Bu gruplar, genellikle birbirleriyle daha fazla etkileşimde bulunan veya benzer özelliklere sahip olan bireyleri içerirler.

Problem Çözümü

Öncelikle dosyadan var olan komşuluk listelerini okumamız gerekti. Dosyadaki formata göre okuma yapıldı. Öncelikle her düğümün komşuları bir **Node** struct yapısında tutuldu. **Node** struct yapısının içinde kaç tane komşusu ve komşularının bilgisi tutuldu. Dosyada karakterler olduğu için karakter şeklinde işlem yapmaktansa integer olarak işlem yapılması tercih edildi. Her karakterin ASCII tablosunda karşılığı olduğu için her karakteri 65 sayısına göre modu alındı. 'A' harfi 65 sayısı ile başlandığı için sonra da alfabetik olarak ASCII karşılığı da arttığı için böyle bir şey yapıldı. Elimizde günün sonunda Node tipinde dizimiz oldu. Komşuluk listesi diyebiliriz. Dosyadan okuma yaparken başka şeyleri de Graph yapısında saklıyoruz. Kenarların ağırlık bilgisi de saklanır. En başta dosyada okurken hepsi '0' olarak atanır.

Elimizde graf yapısı olduğuna göre artık herhangi bir noktadan tüm noktalara olan giden yolları bulmam gerek. Ondan dolayı BFS yöntemi ile gezinmem gerek. Tüm graf dolaşılırken aynı zamanda '**prev**' adlı integer dizimde o düğüme daha önce hangi düğümden gelinmiş onun bilgisi var. Tüm noktalar gezildikten sonra elimde ısı/iz haritası olmuş oldu. Böylelikle o noktadan başka herhangi noktaya hangi düğümler üzerinden gidilmişse kolay şekilde bulanabilecek. Elimde sonuçta harita var. Kaybolmayız.

Tüm düğümler için üst paragrafta anlatılan şekilde gezildikten sonra elimde her düğümün ısı/iz haritası var. Şimdi artık her kenarlar üzerinden ne kadar geçilmiş onu bulmak gerek. '**prev**' adlı integer dizimi baştan aşağı gezilecek. Şu şekilde gezilecek. Grafın içindeki başlangıç düğüm alınıp sonra grafın içindeki her düğüme nasıl gidilir prev dizisine bakarak anlaşılacak. Misal elime gidilecek herhangi bir düğüm alındı. Sonra o düğümden önce hangi düğüme gelinmiş ona bakılır. Eğer ondan önce herhangi bir düğüme gelinmemişse demek ki başlangıç düğümünden gidilmek istenilen düğüme yol yok. Eğer bir önceki değer varsa o değere gidilecek. Sonra bu başlangıç değerine ulaşıncaya kadar devam edecek. Bu gezinmeyi yaparken aynı zamanda kenarları da artırmış olunur. Kenar demek 2 düğüm arasında olduğuna göre o kenardan ne kadar geçildiğini 1 artırarak günün sonunda ulaşılabilir. Kenarların ağırlığı da 'weight' adlı integer matrisinde tutulur.

Elimizde kenarların ağırlığı da var. Artık maksimum kenarı bulup o iki düğüm arasındaki komşuluk ilişkisini kesmek lazım. Yani komşuluk listesinde değerini -1 yapmak lazım. Komşuluk listesinde toplam komşu sayılarını azaltılması lazım.

Silindikten sonra en baştan döngünün başlaması gerek. BFS ile tekrardan bir düğümden diğer tüm düğümler gezilecek. Ağırlıklar hesaplanılacak sonra duruma göre tekrar silinecek.

Durma koşulu ise kullanıcıdan alınan 2 bilgiye göre durur. Eğer belli iterasyon sayısından sonra topluluk sayısı aynı kalır. Bir de topluluk içerisindeki düğüm sayısı belli sayı ya da altına düşerse sonlanır.

Karşılaşılan Problemler

Öncelikle döngü olması direct problemdi benim için. Hatta problem olmaya devam etti maalesef.

Komşuluk matrisinde mi yoksa listesinde mi tutulsa daha iyi olur diye düşünüldü. Link liste yapısı tercih edilmedi. Önceden belli olan graf yapısı olduğu için komşuluk matrisi ve listesi karışımı bir şey yapıldı. Node tipinde dizim max ASCII değeri kadar oldu. İçindeki komşu içeriği de liste oldu. Ondan karışımı oldu.

Graf BFS ile gezildikten sonra tüm yollar nasıl bulunur o biraz uğraştırdı. Ben de gezerken hazır herhanfgi bir düğümden başlangıç düğümene nasıl gidileceğini bulmak için dizide o düğümden önce gelen düğüm kaydedidldi.

Birden fazla maksimum ağrlıkta kenar silinmesi biraz uğraştırdı.

Algoritma Karmaşıklığı

Graftaki Düğümler: N
Maksimum sayisi: M

- Başlangıç düğümünden diğer tüm düğümlere giderken zaman karmaşıklığı N olur. O(N)
- Tüm kenarların ağırlığını hesaplarken graftaki tüm düğümler geziliyor ve bu tekrarlı biçimde yapıldığı için en kötüsü O(N^2) olur.
- Kenarın yok edilmesi O(1) karmaşıklıkta olur. Hangi nokta silinecek belli.

Senaryo

Input:

```
A:B;
B:A,C,D;
D:B,E;
C:B;
E:D;
```

```
D 2
E 1
    A  B  C  D  E
A   0  4  0  0  0
B   4  0  4  6  0
C   0  4  0  0  0
D   0  6  0  0  4
E   0  0  0  4  0
        1. iterasyon
B - D kenari silindi
Dugumlerin Komsu Sayisi
A 1
B 2
C 1
D 1
E 1
    A  B  C  D  E
A   0  2  0  0  0
B   2  0  2  0  0
C   0  2  0  0  0
D   0  0  0  0  1
E   0  0  0  1  0
****************************************
Topluluk Sayisi: 0
C B A
E D

--------------------------------
Process exited after 2.565 seconds with return value 0
Press any key to continue . . .
```