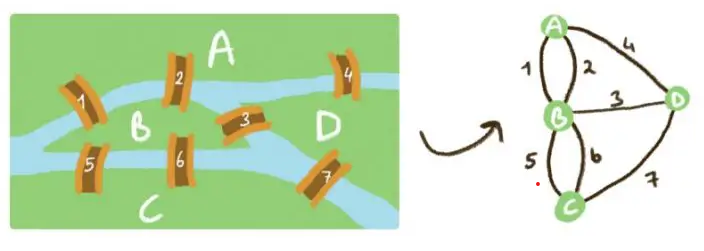
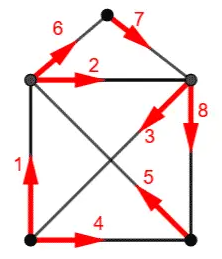
**GRAF TEORİSİ NEDİR?**

Matematikte graf ya da çizge , düğümler ve bu düğümleri birbirine bağlayan kenarlardan oluşan bir tür ağ yapısıdır. 1735 yılında Leonhard Euler tarafından “Doğu Prusya’daki Königsberg şehrindeki 7 köprüden bir ve yalnız bir kez geçerek başladığın yere geri dönebilir misin? “ sorusu ile ortaya atıldı. Euler, bu tür bir sorunun yeni bir düşünme biçimi gerektirdiğini fark etti. Problemin konum geometrisi ile ilgili olduğunu fark eden Euler, böyle bir rota tasarlamanın imkansız olduğunu göstermek için yeni bir geometri türü geliştirdi. Noktalar arasındaki mesafeler önemli değildi. Sayılan tek şey noktalar arasındaki bağlantılardı.



Şekil 1:Köprü Problemi

Euler’in başka bir mantığı da vardı. Mantığına göre , bu yürüyüşünü tamamlamak için her köşeden kaç kere çıkıyorsak o kadar geri girmemiz gerekiyordu. Aksi takdirde yürüyüşümüzü başladığımız noktadan başka bir noktada bitirmiş oluyorduk. Bu mantıkla her köşenin derecesi çift olmalıydı. Derece kavramı ise bir düğümden çıkan hat sayısıdır. Fakat Königsberg grafının köşelerinin dereceleri ise 5,3,3,3 idi. O halde böyle bir yürüyüş gerçekleştirilemezdi. Fakat bu hala bizim sorumuzu çözmeye yeterli değil çünkü bizim sorumuzdaki graftaki köşelerin dereceleri 4,4,3,3,2’dir.

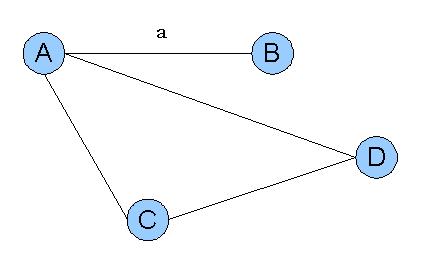
Euler’in bir başka ifadesi ise ise “Euler Yolu” olmuştur. Euler yolu her kenardan tam olarak bir kez geçen bir yoldur. Euler yolunun bulunması için ise gereken şart grafın iki köşesi hariç diğer her köşesinin derecesinin çift olmasıdır. Bu durumda derecelerimizin 4,4,3,3,2 olduğu şekli elimizi hiç kaldırmadan ve bir kenardan sadece bir kez geçecek şekilde çizebiliriz.

Şekil 2:Euler Yolu ile Problemin Çözümü

**GRAF ÇEŞİTLERİ**

**1.Undirected Graph (Yönsüz Graf)**

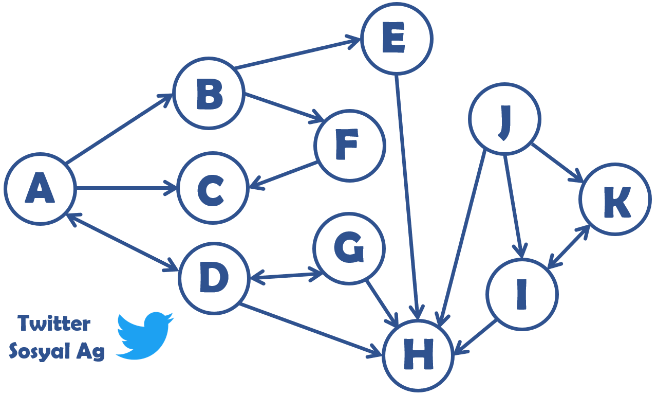
Bu graf çeşidinde düğümler arasındaki hatların yönü yoktur. Bu durumda iki düğüm arasında bulunan kenar , her iki yönlü de hareket edileceğini ifade eder. Örneğin bir sosyal medyada A kişisi B kişisi ile arkadaş olduğunda B kişisi de A kişisi ile arkadaş olmuş olur.



Şekil 3:Yönsüz Graf Örneği

**2.Directed Graph (Yönlü Graf)**

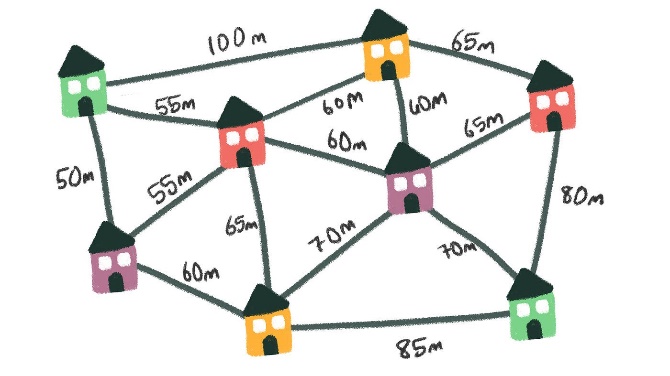
Bu graf çeşidinde düğümler arası yöne dayalı bir ilişki vardır. Örnek olarak A noktası B noktasına bağlı diye B noktası da A noktasına bağlıdır anlamı çıkaramayız.



Şekil 4:Yönlü Graf Örneği

**3.Weighted Graph (Ağırlıklı Graf)**

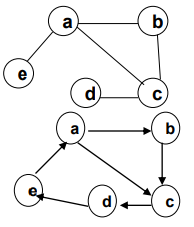
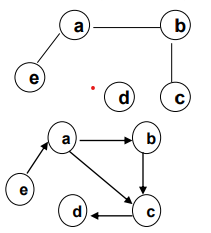
Bu graf çeşidinde hatların bir değeri vardır. Hatlar maliyet, uzunluk, zaman gibi özelliklerine göre ağırlıklandırılır. Örnek olarak bir mahalle krokisi çizilebilir ve yol uzunluğuna göre hatları ağırlıklandırabiliriz.



Şekil 5:Ağırlıklı Graf Örneği

**4.Connected Graph (Bağlı veya Bağlı Olmayan Graflar)**

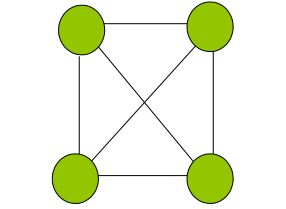
Eğer bir graftaki tüm düğümler arasında en azından bir yol varsa bağlı graftır. Eğer bir grafta herhangi iki düğüm arasında yol bulunmuyorsa bağlı olmayan graftır.

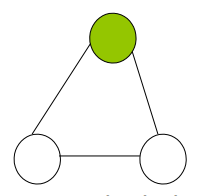


Şekil 7:Bağlı Olmayan Graf Örneği

Şekil 6:Bağlı Graf Örneği

**5.Complete Graph (Komple Graf)**

Bir graftaki iki düğüm arasında bir kenarvarsa komple graftır.



Şekil 8: 3 Düğüm ile Komple Graf Örneği

Şekil 9:4 Düğüm ile Komple Graf Örneği

Dağıtım planı koşulu en çok ihtiyacın duyulacağı yere en kısa zamanda gitmesi gerekmektedir ve bunun için birçok algoritma geliştirilmiştir.Bunlar:

* Kruskal’ın Algoritması
* Prim’in Algoritması
* Sollin’in Algoritması
* Dijkstra Algoritması
* Bellman ve Ford Algoritması
* Folyd Algoritması

**Kruskal’ın Algoritması =>** Daha az maliyetli kenarları tek tek değerlendirerek yol ağacını bulmaya çalışır. Ara işlemler birden çok ağaç oluşturabilir.

**Prim’in Algoritması =>** En az maliyetli kenardan başlayıp onun uçlarından en az maliyetli genişleyecek kenarın seçilmesine dayanır. Bir tane ağaç oluşturur.

**Sollin’in Algoriması =>** Doğrudan paralel programlamaya yatkındır. Aynı anda birden çok ağaçla başlanır ve ilerleyen adımlarda ağaçlar birleşerek tek bir yol ağacına dönüşür.

**Dijkstra Algoritması =>** Ağırlıklı ve yönlü graflar için geliştirilmiştir. Graf üzerindeki kenarların ağırlıkları 0 veya sıfırdan büyük sayılar olmalıdır. Negatif ağırlıklar için çalışmaz.

Başlangıç olarak sadece başlangıç düğümünün en kısa yolu bilinir. (0 dır.)

Tüm düğümlerin maliyeti bilinene kadar devam et.

O anki bilinen düğümler içerisinden en iyi düğümü şeç. (en az maliyetli düğümü seç, daha sonra bu düğümü bilinen düğümler kümesine ekle)

Seçilen düğümün komşularının maliyetlerini güncelle.

**Bellman ve Ford Algoritması =>** Negatif ağırlıklı graflar için geliştirilmiştir.

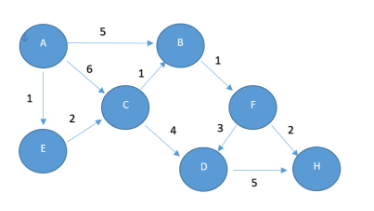
Kullandığımız dijkstra algoritmasını biraz daha inceyecek olursak;

Dijkstra  algoritması düğümler arası en kısa yolu bulmak için kullanılan bir algoritmadır.Günümüzde oldukça popüler olan bu algoritma ,Google Maps,OSPF(Open Shortest Path First) protokolünde ,oyun programlamada ulaşım ağlarında kullanılmaktadır.

Algoritmanın temel amacı graf üzerinde bir düğümden başka bir düğüme giderken en ucuz maliyetle nasıl gidilebileceği hesaplamaktır.

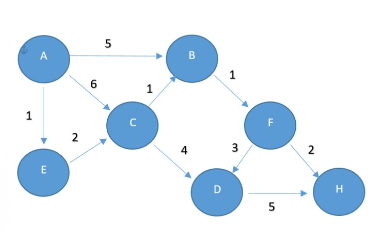
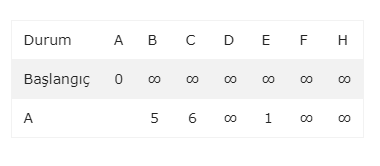
Öncelikle bir başlangıç düğümü seçmemiz gerekir.Başlangıç durumunda diğer düğümlere erişim imkanı olmadığından uzaklık değerine sonsuz değerini atanır.Başlangıçta seçilen düğümden başlanarak komşu olan düğümlere olan uzaklıklara bakılır ve en kısa olan yol seçilerek sonraki düğüme geçilir.Algoritma adım adım çalışırken her adımda mesafe değerlerini günceller.

Bir örnek üzerinde inceleyecek olursak;

A düğümünden H düğümüne en kısa yoldan nasıl gideceğimizi inceleyelim.

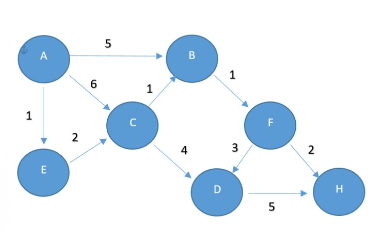
**Adım1:**Başlangıç düğümümüzü A olarak seçelim.Başlangıçta diğer düğümlerin erişimi olmadığı için uzaklık değerlerine sonsuz değerini atarız.

**Adım2:**Başlangıç düğümünden olan A’dan başlayarak komşu olan düğülerine olan uzaklıklarına bakalım.

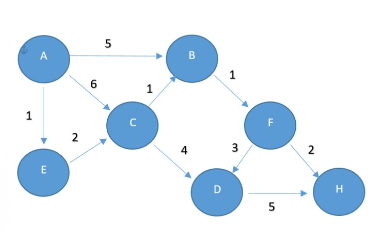


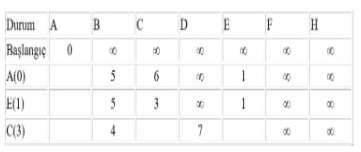
Yolu en kısa olan düğümü seçerek bir sonraki düğüme geçiyoruz.A düğümünün 3 düğüme komşu olduğunu yukarıda görmekteyiz.Burada en az maliyetli olan E düğümünü seçiyoruz.

**Adım3:**E düğümünün C düğümüne komşuluğu bulunduğunu görmekteyiz.C düğümünün yeni değerini 1+2=3 olarak hesaplayabiliyoruz.C değerine E üzerinden gitmek daha az maliyetli olduğu için C değeri üzerinde güncelleme yaparız.Bir sonraki düğüme geçmek için en küçük değere ve daha önceden ziyaret edilmemiş olan C’den devam edelim.

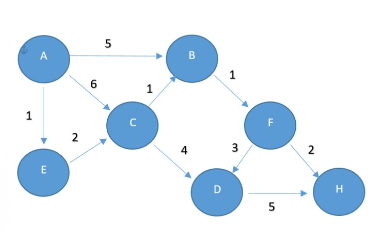
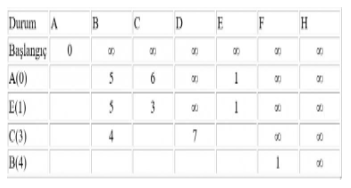


**Adım4:**C düğümündenB ve D düğümlerine komşuluk olduğunu görmekteyiz.B’nin yeni değeri 3+1=4 ve D’nin yeni değeri 3+4=7 oldu.B’nin yeni değeri 5’ten küçük olduğu için B düğümünün değerini güncellemeliyiz. B düğümünün değeri en düşük olduğu için B düğümünden devam ederiz.





**Adım5:**B düğümünün F düğümüne komşu olduğu görülmektedir.F’nin yeni değerini 4+1=5 olarak güncelliyoruz.



**Adım6:**F düğümünün D ve H düğümüne komşu olduğunu görmekteyiz.H düğümünü 5+2=7 olmakta ve D düğümünün değeri ise 5+3’ten 8 olmaktadır fakat 8>7 olduğundan dolayı D 4.adımda güncellenen değerinde kalarak 7 değerinde kalmıştır.

**Sonuç Olarak:**Tüm düğümlere baştan sona baktığımız zaman A düğümünden H düğümüne enkısa yol olarak A-E-C-B-F-H şeklinde olduğu bulduk.Maliyetine bakacak olursak 1+2+1+1+2=7 olduğunu bulmaktayız.

**Kaynakça:**

<https://medium.com/t%C3%BCrkiye/graf-teorisine-giri%C5%9F-c90cbdf9564c>

<http://w3.bilecik.edu.tr/bilgisayar/wp-content/uploads/sites/75/2017/05/10.Hafta_Cizgi_Kumeleri_En_Kucuk_Yol_Agaci.pdf>

<https://bilgisayarkavramlari.com/2008/08/01/yonsuz-graflar-undirected-graphs/>

<https://www.matematiksel.org/graf-cizge-teorisi-nedir/>

<https://mertmekatronik.com/dijkstra-algoritmasi>

Duygu TEKER

1210505072