Maksimum Akış ve Minimum Kesme Algoritması

*Yaşar Can Çilingir

*Bilgisayar Mühendisliği Bölümü, Kocaeli Üniversitesi

Kocaeli, Türkiye

yasarcanccd@gmail.com

Anahtar Kelimeler – "Düğüm", "Akış", "Graf"

Özet— Uygulama başlangıcında kullanıcıdan graf oluşturulması istenir. Kullanıcı her bir düğümü ve düğümler arasındaki edge(kenar)'ların kapasitelerini girer. Kullanıcıdan alınan değerlere göre graph oluşturulur ve kullanıcıya gösterilir. Kullanıcı "Maximum flow" butonuna bastığında graph üzerinde maksimum akış algoritmasının işleyişi adım adım gösterilir. Aynı şekilde "Min Cut" butonuna bastığında graph üzerinde minimum kesme algoritmasının sonucu gösterilir. Buradaki düğümler muslukları/vanaları, düğümler arasındaki kenarlar (edges) ise musluklar arasındaki boru hatlarının kapasitesini temsil etmektedir.

I.GİRİŞ

Projede, literatürde azami akış (maximum flow) olarak geçen ve düğümler (nodes) arasında akış kapasiteleri belirli bir şekildeki (graph) bir başlangıçtan bir hedefe en fazla akışın sağlandığı problemler çözülmektedir. Devamında ise akışın sistemden geçmemesi için literatürde min-cut olarak geçen yöntem uygulanmaktadır. Buradaki düğümler muslukları/vanaları, düğümler arasındaki kenarlar (edges) ise musluklar arasındaki boru hatlarının kapasitesini belirtmektedir. Her bir kenar, kenarın izin verebileceği maksimum akış limiti olan ayrı bir kapasiteye sahiptir. Projedeki amaç kullanıcı istediği zaman 0 düğümünden 5 düğümüne (daha fazla ya da az miktarda düğüm olabilir değerler örnek olarak verilmiştir) azami miktarda akış sağlayabilmektir ve bunu kullanıcıya grafiksel olarak adım adım gösterebilmektir. Akış sağlama ve değerleri güncelleme aşamasında kullanılan sığ öncelikli arama (breadth first search) algoritması ile hangi yolların bulunduğu ilgili kenarlar maviye boyanarak adım adım kullanıcıya gösterilmiştir. Aynı şekilde yine kullanıcı istediği zaman mininmum kesme (min-cut) algoritmasının ilgili graf üzerine uygulanması ve bunu kullanıcıya grafiksel olarak göstermesi gerçekleştirilmiştir.

Uygulama çalıştırıldığında başlangıçta graf oluşturulması için kullanıcıdan alınması gereken bilgilerin olduğu küçük bir arayüz ile başlar. Kullanıcıdan oluşturacağı grafta olacak düğüm sayısı, her bir düğüm arasındaki boru(edge) kapasitesi bilgileri alınır. Başlangıç kaynak düğümü 0. düğüm olarak belirlenmiştir. Hedef düğüm ise en son girilmiş olan düğümdür. Kullanıcı oluşturduğu grafın maksimum akış ve minimum kesme algoritmalarını uygulayabilmesi için ilgili butonlara tıklamalıdır. Butonlara tıklanınca maksimum akış algoritması adım adım grafiksel olarak kullanıcıya gösterilmektedir. Aynı şekilde minimum kesme algoritması da kullanıcıya grafiksel olarak gösterilmektedir.

Kullanıcı arayüzünde 2 tane de örneğin gösterilebilmesi için butonlar bulunmaktadır. Bu butonlar ile önceden oluşturulmuş matris yapısı ile örnek problemlerin maksimum akışı ve minimum kesme grafikleri kullanıcıya gösterilmektedir. Algoritmanın doğru çalıştığını hızlı bir şekilde görebilmek için yapılmışlardır.

II. TEMEL BİLGİLER

Program Java dilinde eclipse geliştirme aracı ile yazılmıştır. Graf sistemlerini görseleştirmek için graphstream kütüphanesi kullanılmıştır. Windows 10 ortamında sorunsuz çalışmaktadır.

Graf, bir olay veya ifadenin düğüm ve çizgiler kullanılarak gösterilmesi şeklidir.
Graf, matematiksel anlamda düğümler ve düğümler arasındaki ilişkiyi gösteren kenarlardan oluşan bir kümedir. Mantıksal ilişki düğüm ile düğüm veya düğüm ile kenar arasında kurulur.
Köse (vertex) adı verilen düğümlerden ve kenar (edge) adı verilip köseleri birbirine bağlayan bağlantılardan oluşan veri yapısıdır. Bağlantılı listeler ve ağaçlar grafların özel örneklerindendir. Her kenar iki düğümü birleştirir.

Ağırlıklı Çizge (weighted graph) : Her kenara bir ağırlık (weight) veya maliyet (cost) değerinin atandığı çizgedir(graftır).

Komşuluk Matrisi: Düğümlerden düğümlere olan bağlantıyı gösteren bir kare matristir. Graf iki boyutlu matrisle gösterilir. Herhangi iki node'un komşu olup olmadığına çok kısa sürede karar verilebilir.

Graf Üzerinde Dolaşma: Graf üzerinde dolaşma grafın düğümleri ve kenarları üzerinde istenen bir işi yapacak veya bir problemi çözecek biçimde hareket etmektir.

Graf üzerinde dolaşma yapan birçok yaklaşım yöntemi vardır. Bu projede kullanılan iki tanesi :BFS (Breadth First Search) Yöntemi ve DFS (Depth First Search) Yöntemi.

BFS Algoritması temel prensip olarak en yakın düğümleri ziyaret etmeyi hedefler. En yakındaki düğümler taranarak uzaktaki düğümlere gidilir. Yakın olan düğümler öncelikli olarak ziyaret edileceği için bizim bir veri yapısı kullanmamızı gerektirmektedir. BFS Algoritmasında kuyruk veri yapısı kullanılır. Ziyaret edilen düğümün komşuları kuyruğa atılır. Her adımda kuyruktan bir eleman çıkartılır ve bu mantıkla ziyaret edilmemiş düğümler kuyruğa eklenir. Bu şekilde tüm düğümler dolaşılır. Eğer bir düğümü arıyorsanız sadece bir if satementı ile işinizi halledebilirsiniz.

DFS algoritmasında kuyruk veri yapısı yerine Stack Veri Yapısı kullanmaktadır. DFS Algoritmasında ziyaret edilen düğümler yığına eklenir. Gidilecek komşu bulunamadığında pop ile yığından eleman atılır ve ziyaret edilebilecek komşu aranır.

Maksimum akış algoritmasının amacı düğümler (nodes) arasında akış kapasiteleri belirli bir şekildeki (graph) bir başlangıçtan bir hedefe en fazla akışın sağlandığı problemleri çözmektir. Genel olarak bildinen ve kullanılan iki çeşit maksimum akış algoritması vardır. Bunlar "Edmonds Karp" algoritması ve "Ford Fulkerson" algoritmalarıdır. Bu programda "Ford Fulkerson maksimum akış" algoritması kullanılmıştır.

Minimum kesme algoritması bir grafı ikiye bölen ve akışın hedefe akmasını tamamen engelleyen en az kenarın kesilmesi (akışının durdurulması) işlemidir.

Minimum kesme algoritmasında akış tamamen engellendiği için kesilen kenarların doluluk toplamları ile maksimum akış ile akan akışın toplamı eşittir.

III. KARMAŞIKLIK ANALİZİ

Projede kullanılan Ford-Fulkerson algoritmasının karmaşıklığı **O(max_flow * E)** 'dir. Worst Case (en kötü durum) olarak her döngüde 1 birim akış eklenir. Bu yüzden karmaşıklık **O(max_flow*E)** 'dir. (E : edge, kenar sayısı)

IV. MİMARİ

Graf yapısı "Adjacent Matrix" (Komşuluk Matrisi) yapısı ile tanımlanmıştır. Graf matrisinin her satırı bir düğümü temsil etmektedir. Her satır içerisindeki indeksler ve değerleri ise ilgili düğümün hangi düğüme hangi kapasite ile bağlandığını göstermektedir.

Örneğin 5x5 boyutlu bir komşuluk matrisinin 2. satırı, graftaki 2. düğümü temsil etmektedir. 2. satırın 4. indexindeki değer ise 2. düğümün 4. düğüme hangi kapasite ile bağlandığını temsil etmektedir.

Algoritma komşuluk matrisi üzerinden çözülmektedir.

Graf yapısını görselleştirip kullanıcıya göstermek için ise graphstream yapısı kullanılmıştır.

V.KULLANILAN FONKSİYONLAR

IV. I *Int fordFulkersonAlgorithm: Çağırıldığı zaman; ullanıcıdan alınmış değerler ile oluşturulmuş matrisi, ford-fulkerson algoritması olarak bilinen maksimum akışı bulma algoritmasına aktarır. Kullanıcıya her adımda güncellenen değerleri grafiksel olarak döndürür. Sonuç olarak maksimum akış değerini döndürür.*

IV. II *Boolean bfs:* Ford-fulkerson algoritması içerisinde kullanılır. Graf yapıları için arama algoritmasıdır. Kullanılma amacı başlangıç akışının bitiş akışına ulaşıp ulaşmadığını kontrol etmek ve ulaştı ise hangi yollar üzerinden ulaştığını bulabilmektir. Akış hedef düğüme ulaştı ise "true", ulaşamadı ise "false" döndürür. Ford-fulkerson algoritması içerisindeki döngü bfs fonksiyonundan "true" değeri döndüğü sürece çalışır. Kullanıcıya her

adımı grafiksel olarak gösterir, hangi yollar üzerinden hedef düğüme gidildiğini adım adım kenarları mavi ile boyayarak belirtir.

IV. III *Void dfs:* Uygulama ekranı açıldığında ilk çağırılan fonksiyondur.

IV. IV *void drawMinCutt: Kullanıcı ilgili butona tıkladığında oluşturulmuş olan graf yapısının, minimum kesme algoritmasına göre hani yolların kesilmesi gerektiğini kullanıcıya grafiksel olarak gösterir.*

VI. SONUÇ

Tüm isterler gerçekleştirilmiştir. İstenilen şekilde kullanıcıdan alınan değerlere göre graf yapıları oluşturulup maksimum akış ve minimum kesme değerleri bulunmuştur ve bulunma aşamasındaki tüm adımlar ve güncellenen değerler kullanıcıya temsili grafiksel yapı ile gösterilmiştir.

VII. KABA KOD

1.ADIM Başla
2.ADIM Kullanıcıdan bilgilerin alınacağı arayüz penceresini oluştur.
3.ADIM Node sayısına göre komşuluk matris boyutunu güncelle
4.ADIM Alınan kenar ve kapasite bilgilerine göre komşuluk matris değerlerini güncelle
5.ADIM Alınan kaynak düğümü bilgisine göre kaynak düğümü değişkenini güncelle
6.ADIM Komşuluk matrisini, kaynak bilgisini ve matris boyutunu maksimum akış bulma algoritmasına aktar
7.ADIM Başlangıç graf matrisine göre arayüzü oluştur ve ilgili düğümleri değerleri ile bağla.
8.ADIM Başlangıç düğümünü kırmızı, hedef düğümünü mavi yap.
9.ADIM Başlangıç graf matrisini kopyala
10.ADIM Kopyalanmış matrisi while döngüsü şartı olarak Breadth First Search algoritmasına aktar.
11.ADIM Matris boyutu kadar visited dizisi oluştur.
12.ADIM Visited dizisinin tüm elemanlarının değerine “false” ata.
13.ADIM Kuyruk yapısı oluştur.
14.ADIM Başlangıç düğümünü kuyruk yapısına aktar.
15.ADIM Visited dizisinin ilgili indeksine “true” değerini ata.
16.ADIM Kuyruk uzunluğu sıfırdan büyük olduğu sürece
17.ADIM Eğer ilgili düğümün komşularının indeksi, visited dizisinde “false” ise
18.ADIM İlgili düğümün komşularını kuyruğa aktar.
19.ADIM İlgili düğümü kuyruktan çıkart.
20.ADIM Eğer hedef düğüme ulaşılmış ise true döndür.
21.ADIM Eğer hedef düğüme ulaşılmamış ise false döndür.
22.ADIM Matris değerlerini güncelle.
23.ADIM Arayüzde grafiksel olarak göster.
24.ADIM Depth First Search algoritmasına kopya matrisi, kaynak düğümünü ve visited dizisini aktar.
25.ADIM Kaynak düğümünden hedef düğümüne gidebilen yolları kontrol et.
26.ADIM Kaynak düğümünden hedef düğüme giden akışı tamamen durduracak minimum sayıda kenarların düğümlerini bul.
27.ADIM Grafiksel olarak ilgili düğümler arasını kırmızı renge boya.
28.ADIM Kullanıcıya göster.

VIII. KAYNAKLAR

http://bilgisayarkavramlari.sadievrenseker.com/2009/03/08/graflarda-kesitler-cut-in-graphs-agaclarda-kesitler/

https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/,

https://www.geeksforgeeks.org/minimum-cut-in-a-directed-graph/

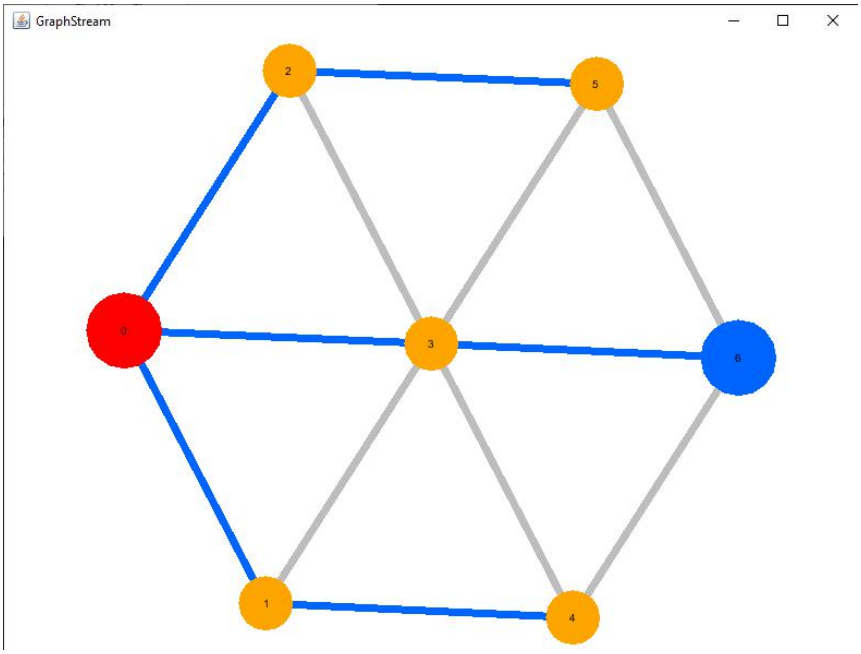
https://www.muhendisbeyinler.net/ford-fulkerson-algoritmasi/

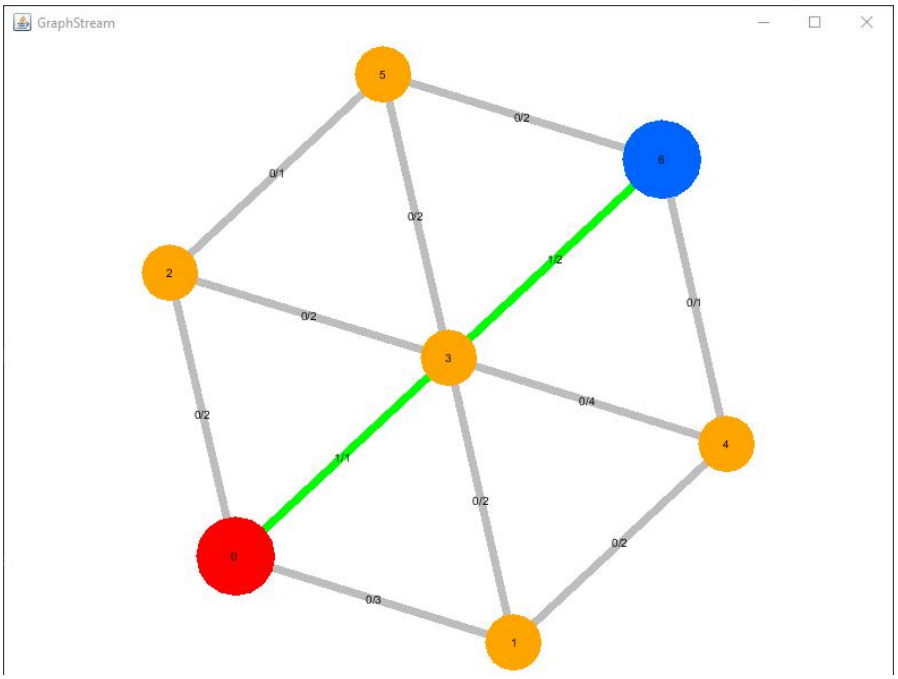
http://bilgisayarkavramlari.sadievrenseker.com/2010/05/22/ford-fulkerson-algoritmasi/

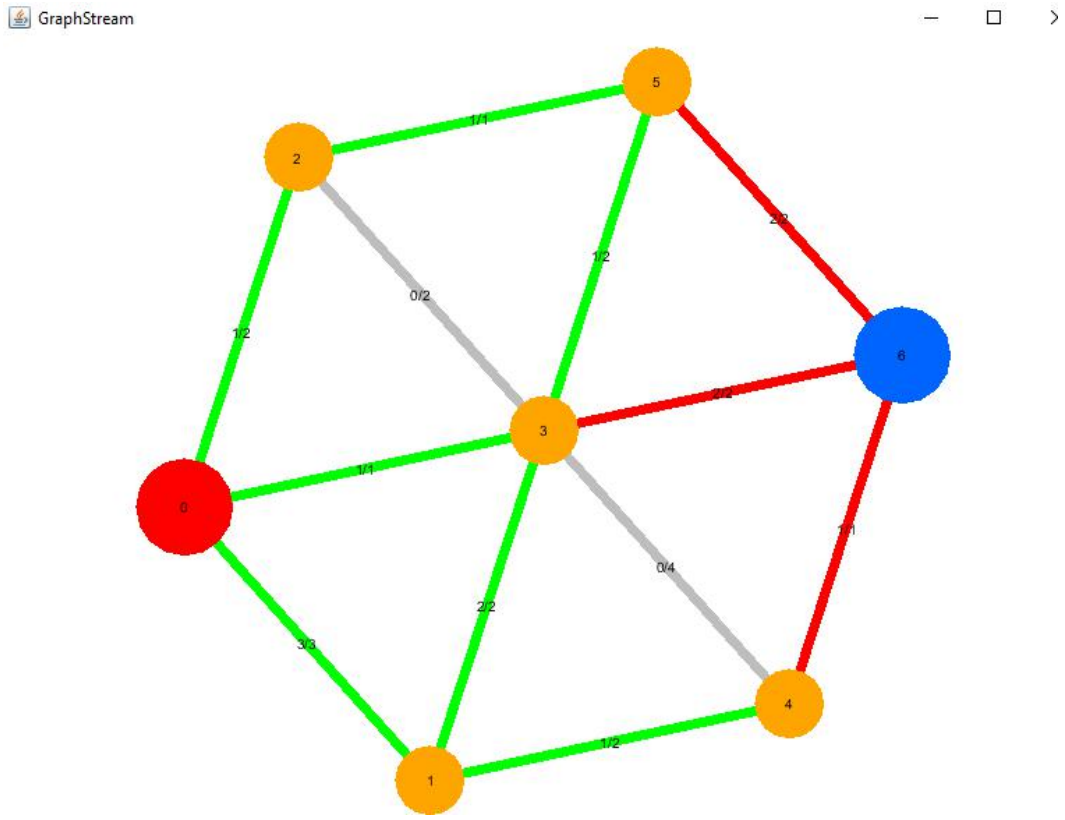
https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph
/
https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/

https://www.geeksforgeeks.org/max-flow-problem-introduction/

IX. BAZI EKRAN GÖRÜNTÜLERİ








```
Min cut ile kesilen yollar:
3 - 6
4 - 6
5 - 6
Max flow:
5
```