HAVUZ PROBLEMİ

Mert BİLGİÇ – Nurdan VAYNİ
bilgic.mert44@gmail.com - nurdannvaynii0@gmail.com
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi

Özet

Bu projede, literatürde azami akış (maximum flow) olarak geçen ve devamında akışın sistemden geçmemesi için literatürde minimum cut (minimum s-t cut) olarak geçen yöntemi kullanarak havuz problemi masaüstü uygulaması yapılması amaçlanmıştır.

1. Giriş

Uygulamada arayüz üzerinden kullanıcıdan musluk sayısı(node/düğüm), havuzu doldurmaya başlayacak olan başlangıç düğümü (source) ve son bulacağı düğüm (sink) alınır. Enter the matrix butonu ile kullanıcıdan alınan düğüm sayısına göre matrix değerlerini girmek için textfieldlardan oluşan bir gridlayout alanı oluşur. Graph değerlerini girdikten sonra save butonu ile kaydedilir ya da get static graph ile tanımlanan graph değerleri kullanılabilir.
Draw graph butonu ile graph çizdirilir. Max flow butonu, maximum flow algoritması ile havuzu maksimum kapasitede dolduracak source'den sink'e mimkün olan maksimum akışı bulur ve graphı çizdirir. Min cut butonu, minimum s-t cut algoritması ile musluktan havuza su akmaması için kesilmesi gereken en az sayıda kenar tespiti yapılır, graph çizdirilir ve kesilmesi gereken kenarlar belirtilir.

2. Temel Bilgiler

Projenin gelişiminde;
Programla dili olarak "Java" kullanılmıştır. Geliştirme ortamı olarak "Apache NetBeans" kullanılmıştır.

3. Yöntem

Maksimum Akış Algoritması:

Maksimum akış için Ford Fulkerson algoritmasının uygulaması olan Edmonds Karp kullanılmıştır. Edmonds Karp algoritması arama işlemi sırasında sığ öncelikli arama (Breadth First Search,BFS) kullanır. Bu algoritmanın amacı, 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 problemleri çözmektir.
Edmonds-Karp algoritması, Ford-Fulkerson'dan farklıdır ve breadth first search kullanarak bir sonraki artım yolunu seçer. Bu nedenle, seçim yapılabilecek birden çok artım yolu varsa, Edmonds-Karp kaynaktan lavaboya en kısa artım yolunu seçeceğinden emin olacaktır.
BFS kullanarak, kaynaktan lavaboya bir yol olup olmadığını öğrenebiliriz. BFS ayrıca parent[] dizisini de oluşturur. Parent [] dizisini kullanarak, bununan yol boyunca ilerleriz ve yol boyunca minimum residual capacity bularak bu yoldan olası akışı buluruz. Daha sonra bulunan yol akışını genel akışa ekliyoruz. Önemli olan, artık grafikte kalan kapasitelerin güncellenmesi gerektiğidir. Yol boyunca tüm kenarlardan yol akışını çıkarırız ve ters kenarlar boyunca yol akışını eklememiz gerekir, çünkü daha sonra ters yönde akış göndermemiz gerekebiliyor.

Edmonds-Karp Karmaşıklık (Complexity) :

Edmonds-Karp, karmaşıklık için maksimum akışa bağımlılığı ortadan kaldırır. Her yinelemeyi O(|E|) zamanında çalıştırdığını ve en fazla |V|.|E| yinelemelerinin olduğunu göstererek, Edmonds-Karp $O(|V|.|E^2|)$ ile sınırlıdır.

Minimum Kesim Algoritması:

Minimum kesim için Minimum s-t cut algoritmasından yararlandık.
Bir akış ağında, s-t kesimi, source 's' (kaynak) ve sink 't' (lavabo) nin farklı altkümelerde olmasını gerektiren bir kesi mdir ve kaynağın yanından lavabonun kenarına giden kenarlardan oluşur. Bir
s-t kesiminin kapasitesi, kesim setindeki her bir kenarın kapasitesinin toplamı ile tanımlanır.
Minimum Kesim ve Maksimum Akış :
Maksimum iki taraflı eşleştirme gibi, bu da Ford Fulkerson algoritması kullanılarak çözülebilen bir sorundur. Bu maksimum akışlı minimum kesme teoremine dayanır. Maksimum akışlı minimum kesme

teoremi, bir akış ağında maksimum akış miktarının minimum kesim kapasitesine eşit olduğunu belirtir. Minimum kesimin tüm kenarlarını yazdırmak için uygulanan adımlar;
-Ford Fulkerson algoritması çalıştırılır ve son kalan grafik dikkate alınır.
-Kalan grafikte kaynaktan erişilebilen köşe kümesi bulunur.
-Ulaşılabilir bir tepe noktasından ulaşılamayan tepe noktasına kadar olan tüm kenarlar minimum kesim kenarlarıdır. Tüm bu kenarlar yazdırılır.
Maksimum akışlı minimum cut teoremi nedeniyle, 2 düğümün minimum cut değeri maksimum flow değerine eşittir. Bu durumda, maksimum flow probleminde kullanılan bazı algoritmalarda bu sorunu çözmek için kullanılabilir.

Minimum Kesim Karmaşıklık (Complexity) :

Minimum kesimi bulmak için Max-Flow tabanlı s-t kesim algoritmasını kullanır. Her köşe çiftini kaynak "s" ve "‘ t "olarak düşünürsek ve s-t kesimini bulmak için minimum s-t kesim algoritmasını kullanırız. Bu algoritmanın mümkün olan en iyi zaman karmaşıklığı bir grafik için O (V5) 'tir.

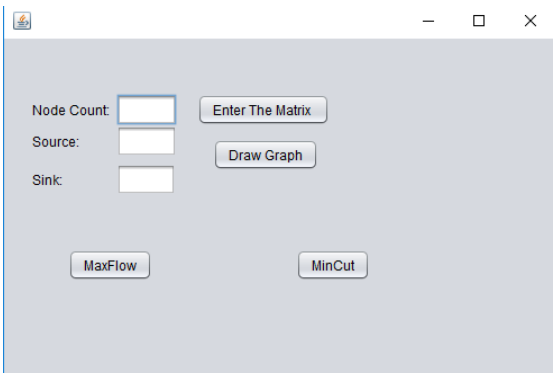
3.1 Sınıflar

-Board.java
-GraphDraw.java
-Main.java
-MatrixDraw.java
-MaxFlow.java
-MinCut.java
-Result.java

4. Arayüz

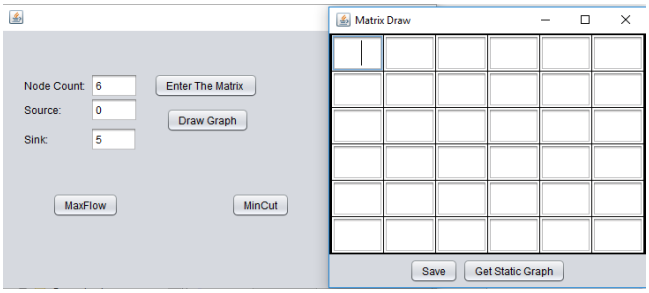
Main Ekran :

Uygulama çalıştırıldığında karşımıza çıkan ana ekran:



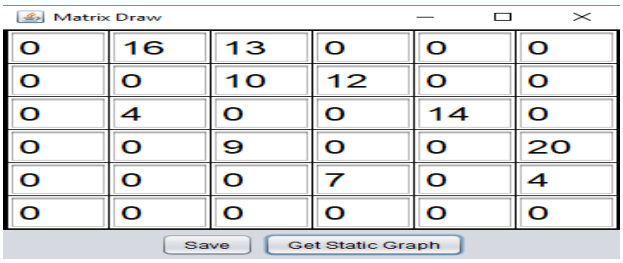
Enter The Matrix :

Düğüm sayısını, source ve sink değerlerini girdikten sonra Enter The Matrix butonu ile graph değerlerini kullanıcıdan aldığımız ekran:



Get Static Graph :

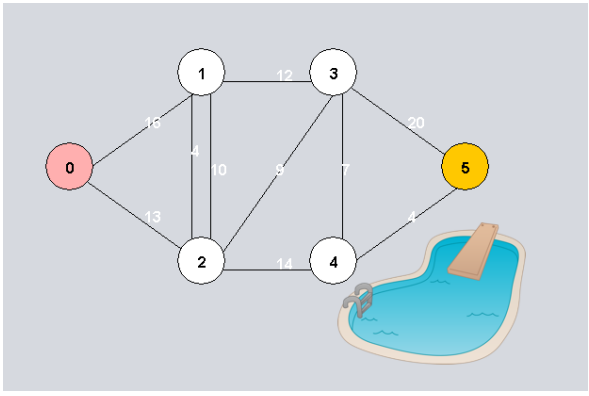
Eğer direkt olarak static graphı kullanmak istersek karşımıza hazır değerleri girilmiş olan ekran çıkar



İki türlü de girilen graph değerlerini kullanmak için Save butonu ile kaydetmemiz gerekir.

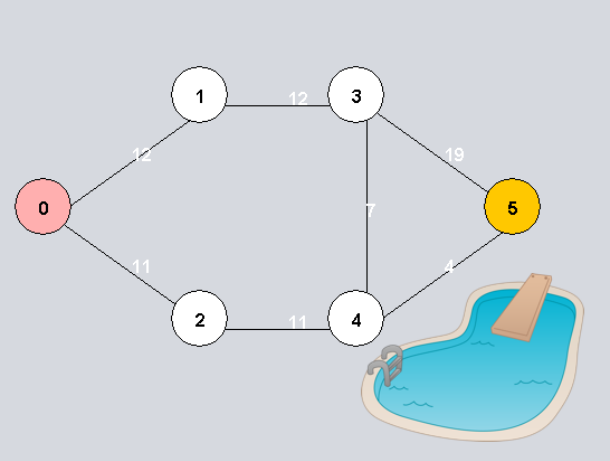
Draw Graph :

Draw Graph butonu ile, graphımızı çizdirebiliriz. Girdiğimiz source değeri pembe rengi ile, sink değeri ise turuncu rengi ile gösterilmiştir. Kenarların kapasiteleri ile düğümlerimizi ve görsel olarak yanında bir havuz resmi görebilmekteyiz.



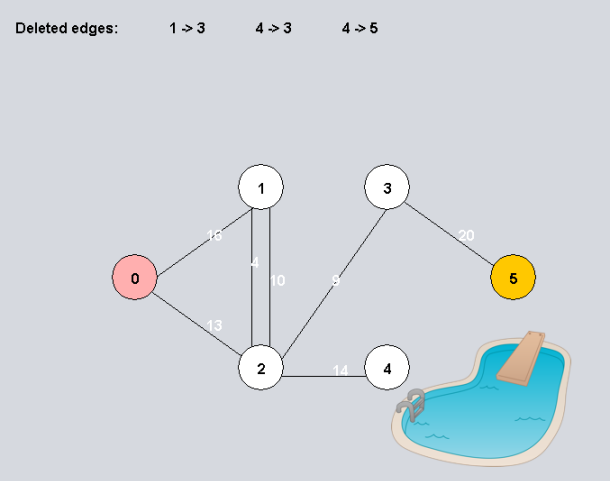
Max Flow :

Maksimum flow algoritması ile oluşturulan graph çizdirilir.



Min Cut :

Minimum cut algoritması ile kesilen kenarlar yazdırılmıştır ve graphın son hali çizdirilmiştir.



5. Kazanımlar

5.1 Java Swing

Java dilinde bulunan ve grafiksel kullanıcı arayüzü geliştirmekte kullanılan Swing kütüphanesini kullandık ve deneyimledik bununla ilgili araştırmalar yaptık.

5.2 Maximum Flow ve Minimum Cut Algoritmaları

Maksimum flow ve minimum cut algoritmalarının ne amaçla kullanıldığını, nasıl çalıştığını araştırdık ve öğrendik. Java dili ile birlikte kullandık.

5.3 Git ve Github Tecrübesi

Grup olarak yaptığımız bu projede, farklı şehirlerde ikamet ettiğimizden ve olağanüstü bir durumdan dolayı uzaktan çalışmak için kontrollü bir şekilde github dan yararlandık ve bu konudaki tecrübemizi arttırdık.

5.4 Dil Tecrübesi

Kaynakların geneli ingilizce olduğu için, araştırma yaparken bunun dil bilgimize de katkı sağladığını ve bize olumlu bir şekilde geri dönüşü olduğunu düşünmekteyiz.

6. Pseudo-code (Yalancı Kod)

6.1. Maksimum akış – Edmonds Karp Algoritması

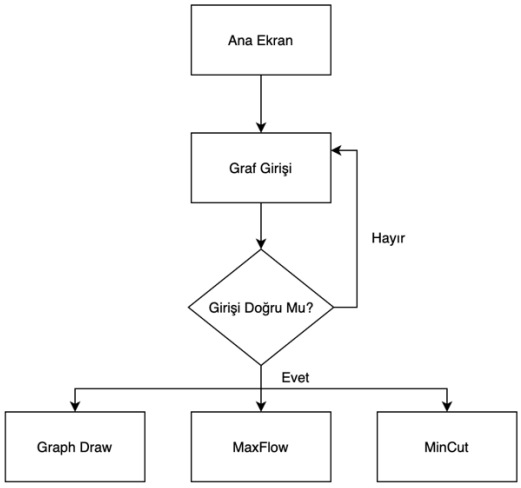
```
algoritma EdmondsKarp
    input:
    Graph
     (graph[v],köşedeki v köşesinden
     çıkan kenarların listesi olmalı
     -dır.
      Orijinal graph ve karşılık
      gelen yapılandırılmış ters
      kenarları geri itme akışı için
      kullanılır.
      Her kenar, parametreler olarak
      Bir kapasiteye, akışa(flow),
      kaynağa(source) ve lavaboya
      (sink)sahip olmalıdır, yanı
      sıra ters kenara bir işaretçi
      içermelidir.

      s       (source vertex)

      t       (sink vertex)


    output:

    flow(akış)
    Maximum akış değeri(Value of
    maximum flow)


    flow := 0
    Akışı sıfıra başlat(Initialize
    flow to zero)

    repeat

    (En kısa s-t yolunu bulmak için
     önce genişlik araması (BFS)
     yapın. Her köşeye ulaşmak için
     alınan kenarı saklamak için
```

```
    'pred' kullanırız, böylece yolu
    daha sonra kurtarabiliriz.)

    q := queue()
    q.push(s)
    pred := array(graph.length)
    while not empty(q)
      cur := q.pull()
      for Edge e in graph[cur] do
        if pred[e.t] = null and e.t ≠ s and
e.cap > e.flow
       then
         pred[e.t] := e
         q.push(e.t)

if not (pred[t] = null) then

(Bir artırım yolu (augmenting path) bulduk.
Ne kadar akış
gönderebileceğimizi görün.)

     df := ∞
     for (e := pred[t]; e ≠ null; e :=
pred[e.s]) do
     df := min(df, e.cap - e.flow)

     (Ve kenarları bu miktarda
güncelleyin.)

      for (e := pred[t]; e ≠ null; e :=
pred[e.s]) do
      e.flow  := e.flow + df
      e.rev.flow := e.rev.flow - df
      flow := flow + df

until pred[t] = null (i.e., artırım
yolu(augmenting path)bulunana kadar)

    return flow(akış)
```

6.2. Minimum Kesme

```
fonksiyon: MinCutPhase
(Graph G, Weights W, Vertex a):
    A <- {a}
    while A != V:
        A'ya sıkıca bağlı tepe noktası
        ekle.
        cut_of_the_phase depolayın ve
        son eklenen iki köşeyi
        birleştirerek G'yi daraltın.

minimum = INF
fonksiyon: MinCut
(Graph G, Weights W, Vertex a):
    while |V| > 1:
        MinCutPhase(G,W,a)
        if cut_of_the_phase < minimum:
          minimum = cut_of_the_phase
    return minimum
```

7. Akış Şeması



8. Kaynakça

- https://www.geeksforgeeks.org/max-flow-problem-introduction/
- https://www.geeksforgeeks.org/minimum-cut-in-a-directed-graph/
- https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/
- https://brilliant.org/wiki/max-flow-min-cut-algorithm/
- https://en.wikipedia.org/wiki/Edmonds%E2%80%93Karp_algorithm
- http://www.yazilimmutfagi.com/index.php/2015/01/15/java-frame-ders-1-gorsel-sinif-gui-olusturma/
- https://www.javatpoint.com/java-swing
- http://www1.cs.columbia.edu/~bert/courses/3137/hw3_files/GraphDraw.java
- https://www.javatpoint.com/Graphics-in-swing
- https://tutorialspoint.dev/data-structure/graph-data-structure/minimum-cut-in-a-directed-graph
- http://www.mathcs.emory.edu/~cheung/Courses/323/Syllabus/NetFlow/max-flow-min-cut.html
- https://downey.io/blog/max-flow-ford-fulkerson-algorithm-explanation/
- https://stackoverflow.com/questions/36681289/displaying-jlabel-matrix
- https://stackoverflow.com/questions/11745391/java-how-to-flash-jpanel-window

- https://stackoverflow.com/questions/5752307/how-to-retrieve-value-from-jtextfield-in-java-swing
- https://stackoverflow.com/questions/5064393/using-loop-to-get-values-from-jtextfields
- https://stackoverflow.com/questions/2832472/how-to-return-2-values-from-a-java-method
- https://www.hackerearth.com/practice/algorithms/graphs/min-cut/tutorial/
- http://www.mathcs.emory.edu/~cheung/Courses/323/Syllabus/NetFlow/max-flow-min-cut.html