HAVUZ PROBLEMİ

Onur Kuş, Abdülbaki Bayraktar

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

[onurkus58@gmail.com](mailto:onurkus58@gmail.com), 61a.baki61@gmail.com

Özet

*Yazılım Laboratuvarı 2 dersi üçüncü projesinde Kesikli Matematik, Nesneye Yönelik Programlama ve Veri Yapıları ve Algoritmalar derslerinde edinilen bilgilerin uygulamaya geçirilmesi amaçlanmıştır.*

# Giriş

Uygulama da boru hattındaki su akışını "Maximum Flow" ile azamı akışı modellenmiştir. Boru hattındaki en az kesilebilen borular "Minimum Cut" ile asgari kesiti modellenmiştir. "Maximum Flow" un "Ford-Folkerson" algoritması kullanılmıştır. "Ford-Fulkerson" algoritması, çözüm için öncelikle hedef düğüme giden yolu bulur. Algoritma bu arama işlemi sırasında şayet derin öncelikli arama (depth first search, DFS) kullanıyorsa "Ford-Fulkerson" olarak isimlendirilir.

"Ford-Fulkerson" algoritmasını daha ayrıntılı anlatacak olursak ilk olarak derin öncelikli arama işlemi gerçekleştirilir. İlk bulunacak path içindeki en küçük kapasiteyi bulmaya çalışır. Aslında bu bulunan sayı belirtilen path üzerinden akabilecek maksimum değerdir. Daha sonrasında yoldaki anlık akış miktarları Artık Grafa aktarılır.

Ardından derin öncelikli arama devam eder. Artık grafa akatrılan kısım grafın normal halinden çıkarılır. Ardından kalan path üzerindenki en küçük kapasiteyi bulmaya çalışır.

Bu işlem belirtilen başlangıç ve bitiş düğümleri arasındaki yol bitene kadar devam eder. İşlem sonucunda ise sonuç olarak başlangıç ve bitiş yolundan akabilecek azami akış yani "Maximum Flow" değeri ve path döndürülür. "Minimum Cut" ise bir grafı, iki parçaya bölen kesitin, en az kenarı kesmesi durumudur. Yani en az kesilebilen kenardan kesilmiştir. Bu kesilme dışındaki bütün kesitler bu kesitten daha yüksek veya aynı boyuttadır.

# Temel Bilgiler

Proje gelişiminde; Programlama dili olarak "Java" kullanılmıştır. Arayüz tasarımı kısmında "Swing" kütüphanesi kullanılmıştır. Geliştirme ortamı olarak "Netbeans" kullanılmıştır.

# Tasarım

Havuz Problemi Uygulaması projesi programlanma aşamaları, altta belirtilen başlıklar altında açıklanmıştır.

## Arayüz

1. **Graph Arayüzü**

*Graph Bilgileri altında aşağıdaki girişler beklenir.*

***-Düğüm sayısı:*** Kullanıcıdan düğüm sayısının (Integer) girişini bekler.

***-Başlangıç Düğümü:*** Kullanıcıdan başlangıç nodunu harf (String) cinsinden girişini bekler.

***-Bitiş Düğümü:*** Kullanıcıdan bitiş nodunu harf (String) cinsinden girşini bekler.

*Boru Hattı Bilgileri altında aşağıdaki girişler beklenir.*

***-Başlangıç Düğümü:*** Boru hattını temsil eden düğümlerin her biri için başlangıç düğümü bilgilerini harf (String) cinsinden girişi beklenir.

***-Bitiş Düğümü:*** Boru hattını temsil eden düğümlerin her biri için bitiş düğümü bilgilerini harf (String) cinsinden girişi beklenir.

***-Kapasite:*** Boru hattını temsil eden düğümlerin her biri için kapasite (Integer) bilgilerinin girişini beklenir.

***Görüntüle:*** Girilen bilgiler doğrultusunda oluşan düğümleri gösteren arayüze geçişi sağlar.

1. **Animasyon Arayüzü**

Açılan ara yüzde önceki ara yüzde bilgileri alınmış düğümleri Swing kütüphanesi yardımıyla çizilmiş hali görüntülenir. Boru hattı bilgileri kısmında önceki ara yüzde kullanıcından bilgileri alınmış düğümlerin başlangıç düğümü, bitiş düğümü ve kapasite bilgileri listelenir.

***-Max Flow Butonu:*** Bilgileri kullanıcıdan alınan düğümlerle gösterilen boru hattının Ford-Fulkerson algoritması yardımıyla "Max Flow" gösterilir. Butonu ilk tıklandığında butonun sağında "Max Flow"un yol sayısı,butonun sol tarafında "Max Flow"un Ford-Fulkerson algoritması yardımıyla hesaplanan sayısal değeri listelenir.

Yol sayısına bağlı olarak butona her tıklandığında farklı bir "Max Flow" yolu daha önceden çizilmiş halinde farklı bir renk ile gösterilir.

***-Min Cut Butonu:*** Bilgileri kullanıcıdan alınan düğümlerle gösterilen boru hattının "Min Cut" gösterilir. Butona her basıldığında "Min Cut" algoritması tarafından kesilmesi gereken yollar düğümün daha önceden çizilmiş halinde farklı bir renk ile gösterilir.

***-Orjinal Graph Butonu:*** Bilgileri kullanıcıdan alınan düğümlerle gösterilen boru hattının kullanıcıdan bilgileri alınmış halini gösterilir. Butonu her basıldığında ilk hali gösterilir.

***-Yeni Havuz Problemi Butonu***: Bu butona basıldığında kullanıcıdan alınmış bütün veriler sıfırlanır ve yeni veriler girebilmesi için ilk açılan ara yüze yönlendirilir.

## Fonksiyonlar

* **noktaCiz()**

Girilen koordinatlara göre düğümlerin ekrana basar.

* **cizgiCiz()**

Girilen koordinatlara göre kenarların ekrana basar.

* **kordinatlariSifirlama()**

Koordinatları sıfırlar.

* **maxflowButonActionPerformed()**

Butonuna basıldığında Max-Flow işlemlerini gerçekleştirir.

* **mincutButonActionPerformed()**

Butonuna basıldığında Min-Cut işlemlerini gerçekleştirir

* **orginalgraphButonActionPerformed()**

Butonuna basıldığında graphın orijinal halini ekrana basar.

* **yenihavuzproblemiButonActionPerformed()**

Butonuna basıldığında yeni havuz problemi oluşturmak için Graph Arayüzü açar**.**

# Kazanımlar

* **Graph Yapısı**

Graph yapısının proje kullanımında yeterli tecrübemiz yoktu. Bu proje sayesinde Graph yapısını uygulamada işleyişini daha iyi anlamış olduk.

* **Max-Flow (Ford-Fulkerson) Algoritması**

Max-Flow algoritmasının nasıl çalıştığı ve koda nasıl uyarlandığını tecrübe etmiş olduk.

* **Min-Cut Algoritması**

Min-Cut algoritmasının nasıl çalıştığı ve koda nasıl uyarlandığını tecrübe etmiş olduk.

* **Swing Kütüphanesi**

Swing kütüphanesini daha farklı kullanımlarını uygulayarak tecrübe etmiş olduk.

# Kaba Kod ve Karmaşıklık (Big-O)

## Karmaşıklık (Big-O)

Artık graphda (**Residual Graph)** açık bir yol olduğu sürece, yoldaki minimum kapasitelerden en azını gönderir. Algoritmada ancak tüm ağırlıklar rasyonel olduğunda sonlandırılır. Aksi takdirde algoritmanın maksimum değeri bulunmaması mümkündür. Algoritma sona ererse, maksimum değeri bulunur.

E: Kenar sayısı, max\_flow: Kenarlar arası maksimum geçiş

 Algoritma karmaşıklığı: O (max\_flow \* E)

## Kaba Kod (Pseudo Code)

***Ford-Fulkerson Algoritması:***

**BAŞLA**

(İlk akışta max\_flow 0 olarak ayarlanır.)

**While** (Artık graph içinde başlangıç düğümden(s) bitiş düğümüne(t) bir path\_flow varsa)

*{max\_flow=path\_flow+ max\_flow}*

**RETURN** (max\_flow)

# Akış Diyagramı

# Kaynakça

1. <https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/>
2. <https://www.geeksforgeeks.org/minimum-cut-in-a-directed-graph/>
3. <https://www.it-swarm.dev/tr/java/jpanelde-nasil-cizilir-swinggrafik-java/972804061/>
4. <http://www.yazilimmutfagi.com/index.php/2010/07/12/java-grafik-kutuphanesi/>
5. <https://en.wikipedia.org/wiki/Ford%E2%80%93Fulkerson_algorithm>
6. <https://www.quora.com/What-is-the-time-complexity-of-Ford-Fulkersons-algorithm>
7. <https://brilliant.org/wiki/ford-fulkerson-algorithm/>
8. <https://cp-algorithms.com/graph/edmonds_karp.html>
9. <https://www.muhendisbeyinler.net/ford-fulkerson-algoritmasi/>
10. <https://discuss.codechef.com/t/problem-in-understanding-time-complexity-of-ford-fulkerson-algorithm-for-maximum-flow-problem-plz-help/18675>