**HAVUZ PROBLEMİ**

|  |
| --- |
| Abdulkadir GÜNDÜZ  PROJE 3 |

***Özet--*** ***Projede******, düğümler arasında maksimum akışı sağlayacak yolu bulan ve havuz dolduktan sonra kesilmesi gereken en az sayıda kenarı tespit eden uygulamayı yaptık.***

I. GİRİŞ

Proje gelişiminde programlama dili olarak “Java”, geliştirme ortamı olarak “Netbeans” kullanılmıştır.

Bu projede, literatürde azami akış ( maximum flow ) olarak geçen ve düğümler arasında akış kapasiteleri belirli bir şekildeki (graph) bir başlangıçtan bir hedefe en fazla akışın sağlandığı problemlere çözüm bulduk. Devamında ise akışın sistemden ​geçmemesi için literatürde min-cut olarak geçen yöntemi uyguladık.

II. YönteM

Algoritma şayet derin öncelikli arama (depth first search ,DFS) kullanıyorsa ford fulkerson olarak isimlendirilir. Şayet aynı algoritma bu arama işlemi sırasında sığ öncelikli arama (breadth first search, BFS) kullanırsa bu durumda da edmonds karp algoritması olarak isimlendirilir.

Biz uygulamada Ford-Fulkerson algoritmasını kullandık. Bu yöntemin sözde kodu oldukça kısadır;

initialize flow to 0

path = findAugmentingPath(G, s, t)

while path exists:

augment flow along path #This is purposefully ambiguous for now

G\_f = createResidualGraph()

path = findAugmentingPath(G\_f, s, t)

return flow

Akış artırımını daha ayrıntılı olarak açıklayan sözde kodun bir sürümü:

flow = 0

for each edge (u, v) in G:

flow(u, v) = 0

while there is a path, p, from s -> t in residual network G\_f:

residual\_capacity(p) = min(residual\_capacity(u, v) : for (u, v) in p)

flow = flow + residual\_capacity(p)

for each edge (u, v) in p:

if (u, v) is a forward edge:

flow(u, v) = flow(u, v) + residual\_capacity(p)

else:

flow(u, v) = flow(u, v) - residual\_capacity(p)

return flow

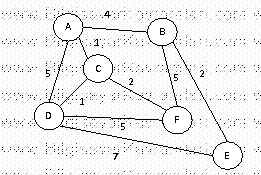
Ford-Fulkerson algoritmasının karmaşıklık analizi (Big O complexity analysis) = O(M\*f)

Burada M kenar sayısı ve f maksimum akış değeridir.

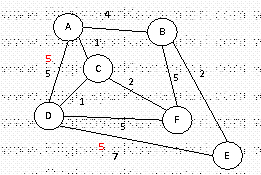
III. deneysel sonuçlar

İlk bulacağımız yol (path) A-D-E olsun bu durumda algoritma bu yol üzerindeki en küçük kapasiteyi bulmaya çalışır:

min(A-D,D-E) = min(5,7) = 5



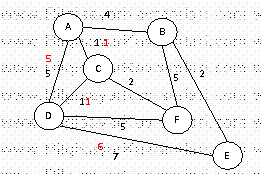
Bu bulunan sayı aslında bu yol üzerinden akabilecek maksimum değerdir. Dolayısıyla bu yoldan 5 kapasitesinde akış yapılmasına karar verilir ve şekil üzerinde bu durum işaretlenir:



Önceki şekilde buluna kırmızı sayılar, o yoldaki anlık akış miktarlarıdır. Ardından derin öncelikli arama işlemi devam eder ve örneğin A-C-D-E yolunu bulur.

min(A-C,C-D,D-E) = min(1,1,2) = 1

Yukarıda dikkat edilecek bir nokta, D-E aralığının şeklin” ilk halinde olan 7 olarak alınması yerine 2 olarak alınmasıdır. “şu anda kullanılıyor olması ve artık geriye 2 miktarında kapasite kalmasıdır.

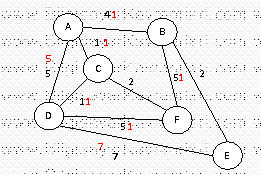


Bir sonraki derin öncelikli arama yolumuz A-C-F-D-E olsun. Bu durumda yol üzerindeki kapasite hesabı aşağıdaki şekilde olacaktır:

min(A-C,C-F,F-D,D-E) = min (0,2,5,7) =0

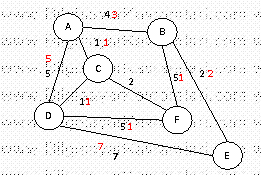
Görüldüğü üzere bu yol artık kullanılamaz çünkü bu yoldaki kapasite 0’dır. Bunun sebebi bir önceki adımda bulunan yolun, A-C arasındaki kapasiteyi sonuna kadar kullanmış olmasıdır.Sıradaki yolumuz A-B-F-D olsun:

min(A-B,B-F,F-D) = min( 4,5,5,1) = 1



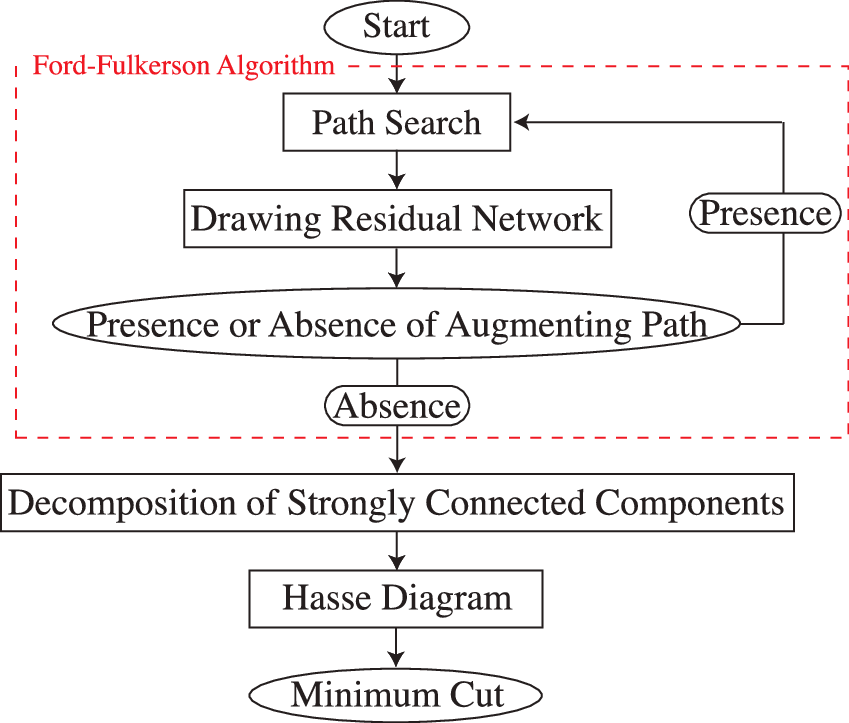
Son olarak yolumuz A-B-E olarak verilsin :

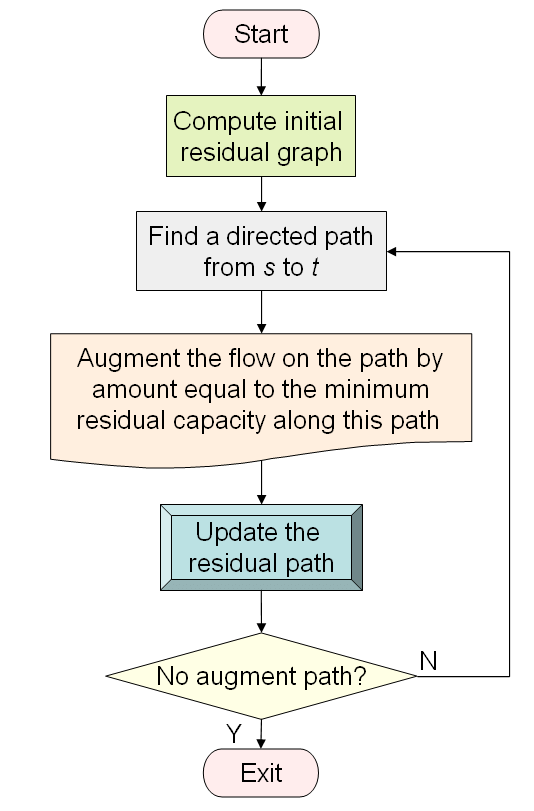
min(A-B,B-E) = min (3,2) = 2



Yukarıdaki şekilde bütün düğümler arasındaki akışlar gösterilmiştir. Gerçekten de graftaki E düğümüne (hedef düğüme) gidebilen yollar maksimum kapasite ile doldurulmuştur.

IV. akış diyagramı





V.SONUÇ

Arayüz üzerinden kullanıcıdan,

* Musluk sayısı (node/düğüm),
* Musluk arasında bağlantı bilgisini veren boru hattı (edge/kenar)
* Ve boru hatlarının kapasiteleri alıyoruz.

Alınan bilgilere göre bir​ graf yapısı dinamik​ olarak gösteren bir arayüz hazırladık. Projenizin çalışma adımları:

1. Kullanıcı havuzu doldurmaya başlayacak olan, başlangıç düğümünü seçer.

2. Geliştirdiğiniz algoritma havuzu maksimum kapasitede dolduracak yolu bulur.

3. Daha sonraki aşamada ise musluktan havuza su akmaması için kesilmesi gereken en az sayıda kenar tespiti yapılır.

Projede başlangıç düğümünden bitiş düğüme azami miktarda akış sağlayarak havuzu doldurduk ve sonrasında en az kenar akışı kesilerek (min-cut) akışı durdurduk.

KAYNAKLAR

1. <https://brilliant.org/wiki/ford-fulkerson-algorithm/> (Ford-Fulkerson Algoritması)
2. <http://bilgisayarkavramlari.sadievrenseker.com/2010/05/22/ford-fulkerson-algoritmasi/>
3. https://www.youtube.com/watch?v=fjT3WDKiAkI (Access date: 24 Kas 2016)
4. https://www.youtube.com/watch?v=LVrtVxNzXCU (19 Şub 2015)
5. https://www.youtube.com/watch?v=TzLuAjhfSrE (Nisan 20, 2017)
6. <https://www.youtube.com/watch?v=YGsmWKMMiYs&feature=youtu.be>
7. https://www.youtube.com/watch?v=YGsmWKMMiYs