**HAVUZ PROBLEMİ PROJESİ**

YAZILIM LABORATUVARI-II PROJESİ

Gözde ÖRGÜ – Büşra ERKAN

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

[gozde.orgu@gmail.com](mailto:gozde.orgu@gmail.com) [busraerkan39@gmail.com](mailto:busraerkan39@gmail.com)

**ÖZET**

*Havuz Problemi Projesi’nde bir graph üzerinde maksimum akışlı min-cut teoremi kullanmamız beklenmektedir. Amaç öğrencilerin algoritma çözümleme yapısını anlaması ve çözüm sağlayabilmesidir.*

**1.Giriş**

Projede teoremler ‘Havuz Problemi’ üzerinden uygulanmaktadır. Burada bizden beklenen düğümler arasındaki akış kapasiteleri belirli olan bir graph üzerinden başlangıç ve hedef noktalar arasındaki en fazla akışın sağlandığı problemleri azami akış(maksimum flow) ile çözmektir. Akışın sistemden geçmemesi için daha sonra min-cut yöntemini uygulamamız beklenmektedir.

Projede düğümler muslukları; düğümleri birbirine bağlayan kenarlar ise musluklardan geçen suyun kapasitesini belirtmektedir. Her kenar farklı kapasiteye sahip olabilir. Düğüm sayısını kullanıcılar tarafından belirlendiği için değişkenlik gösterebilir.

Musluk sayısı, bağlantı bilgileri, bağlantılar arası kapasite bilgileri, başlangıç ve bitiş düğümlerinin kullanıcıdan arayüz üzerinden alınması beklenmektedir.

**2.Temel Bilgiler**

Program C# dilinde yazılmış olup proje gelişiminde; Tümleşik Geliştirme Ortamı olarak “Visual Studio 2017” kullanılmıştır. Graph yapısı Form ekranı üzerinde gösterilmiştir.

**3.Yazılım Tasarımı**

**3.1 Algoritma**

Proje masaüstü uygulaması olarak oluşturulmuştur. Çalıştırıldığında bir form ekranı açılmaktadır. Form üzerinden kullanıcıdan musluk sayısı, bağlantı bilgileri, boru hatlarının kapasitesi, başlangıç ve bitiş noktaları seçilmektedir. Burada musluklar düğümleri, boru hatları ise düğümleri bağlayan kenarları ifade etmektedir. ‘Graph Çiz’ butonuna tıklandığında algoritma çalışmaya başlamaktadır. İlk olarak girilen bilgilere göre bir graph çizilir. Graph en fazla 20 düğüme kadar oluşturulabilmektedir.

Geliştirdiğimiz max-flow algoritması çizilen graphın temsil ettiği havuzu maksimum kapasitede dolduracak yolları bulmaktadır. Bulunan yollar görsel olarak dinamik bir şekilde form üzerinde gösterilmektedir. Kapasiteler üzerinde yapılan işlemler de graph üzerinde görülmektedir. Max-flow bu işlemler sonucunda hesaplanıp form ekranında gösterilir.

Bir sonraki aşama musluktan havuza su akmaması için kesilmesi gereken en az sayıda kenar tespitinin yapılması aşamasıdır. Bu noktada min-cut algoritması devreye girmektedir. Oluşturulan algoritmada kesilmesi gereken kenarların hangi noktalar arasında olduğu bulunmaktadır. Bulunan kenarlar da graph üzerinde belirtilmiştir.

Proje başarılı bir şekilde bütün isterleri vermektedir.

**3.2 Kullanılan Yöntem**

Projede azami akış ( maximum flow ) olarak geçen ve düğümler arasındaki akış kapasitelerinin belirli olduğu bir şekildeki (graph) başlangıç düğümünden hedef düğüme en fazla akışın sağlandığı problemleri çözmeniz beklenmekteydi. Burada düğümler muslukları temsil etmekte, boru hatları ise kenarları temsil etmekteydi. Maximum flow problemi için Ford-Fulkerson algoritması kullanılmıştır. Ford-Fulkerson çözüm için form ekranına çizilen graph üzerinde öncelikle hedef düğüme giden yolu bulur. Algoritma bu arama işlemi sırasında ‘Derin Öncelikli Arama(Depth First Search, DFS)’ kullanmıştır. Bu aramada ilk olarak belirtilen başlangıç düğümüne gidilir. Daha sonra bu düğümün gezilmemiş komşularından biri olan ve kapasitesi 0’dan büyük olan düğüm seçilerek oraya gidilir. Belirtilen hedef düğüme ulaşıldığında algoritma sonlandırılır.

Akışın sistemden geçmemesi için uygulanması gereken min-cut yönteminde ‘[Sığ Öncelikli Arama (Breadth First Search, BFS)](http://www.bilgisayarkavramlari.com/2010/05/22/sekillerde-sig-oncelikli-arama/)’ den yararlanılmıştır. Burada her düğüm için o düğümün müsait olan alt düğümlerine gidilir ve aynı işlem düğümler bitene kadar devam eder. Algoritmanın lineer bir şekilde ilerlemesi için kuyruk(queue) yapısı kullanılır. Min-cut işlemi için DFS(Depth First Search)’den de yararlanılmıştır.

**3.3 Karmaşıklık Analizi**

DFS için **daha önce gezilmiş bir düğüme tekrar gidilmemeli** kuralına göre her düğüm tam olarak 1 kez gezilmektedir. Düğüm sayısını N ile temsil edersek karmaşıklık **O(N)** olmaktadır. BFS de DFS’ye benzediğinden karmaşıklığı **O(N)** ‘dir.

Ford-Fulkerson algoritması karmaşıklığı **O(VE^{2})t**’dir. Burada V düğüm sayısı, E de kenar sayıdır.

Min-Cut probleminde düğüm sayısını N ile temsil edersek, karmaşıklık **O(V^2)** olmaktadır.

**3.4 Algoritma Kaba Kodu**

1. İlk artık graph oluştur.
2. Graphta başlangıç noktasından bitiş noktasına ulaşan yol olduğu sürece  
   aşağıdaki adımları yap:  
   **a)** Başlangıç noktasından bitiş noktasına olan yolu belirle.  
   **b)**Yol üzerindeki maksimum kapasiteyi(δ )belirle.  
   **c)** Yol üzerindeki akışı δ değerince arttır  
   **d)** Geri dönüş yolundaki akışı δ değerince azalt.
3. Maksimum akış probleminden sonra ortaya çıkan graphı dikkate al.
4. Graphı iki parçaya bölen kesiti tespit et ve graph üzerinde göster.

**3.5 Fonksiyonlar**

*public int minBulma(Stack<int> minKapasite)*

Bu fonksiyonda kapasite değerlerinin arasındaki en küçük değer bulunur ve return edilir. Fonksiyona kapasite bilgilerinin tutulduğu yığın gönderilmektedir.

*private bool bfs(int[,] GraphMC, int s,int t, int[] parent)*

Bu fonksiyonda “Breadth First Search(Sığ Öncelikli Arama)” algoritması uygulanmaktadır. GraphMC parametresi problemi çözeceğimiz graphın kopyasıdır; s parametresi başlangıç düğümünün indeksidir; t parametresi sonuncu düğümün indeksidir; parent dizisi parametresi ise kontrol edilen düğümün bir üstünde yer alan düğümdür. Return edilen değer sonuncu düğümün gezilip gezilmediği bilgisini vermektedir.

*private static void dfs2(int[,] GraphMC, int s, bool[] visited)*

Bu fonksiyonda “Depth First Search(Derin Öncelikli Arama)” algoritması uygulanmaktadır. GraphMC parametresi problemi çözeceğimiz graphın kopyasıdır; s parametresi başlangıç düğümünün indeksidir. Başlangıç düğümünden başlayarak 0’dan büyük olan ve gezilmeyen düğümleri gezer.

*public void minCut(int[,] graph, int bas, int son)*

Akışın sistemden geçmemesi için min-cut yönteminin uygulandığı fonksiyonudur. Uygun kenarlar bulunarak form ekranında gösterilmektedir. Gönderilen parametreler graph matrisi, başlangıç ve bitiş düğümleridir.

*public void dfs(int curNode)*

Bu fonksiyonda “Depth First Search(Derin Öncelikli Arama)” algoritması uygulanmaktadır. Maksimum akış problemini çözmek için kullandığımız algoritmada DFS’den yararlanılmıştır. curNode parametresi ile fonksiyona düğüm indeksi gönderilir.

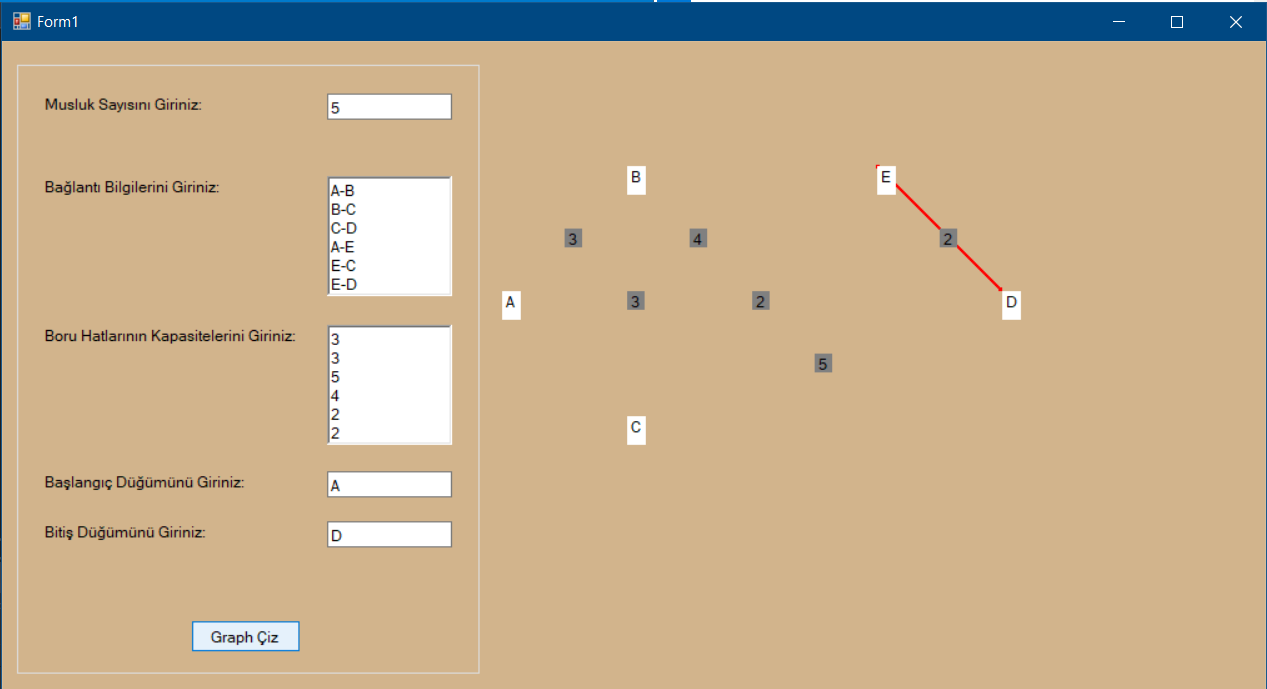
*private void button1\_Click(object sender, EventArgs e)*

Burada kullanıcının form ekranına girdiği verilerle graph çizdirilmektedir. Daha sonra grapha max-flow ve min-cut algoritmaları uygulanmaktadır.

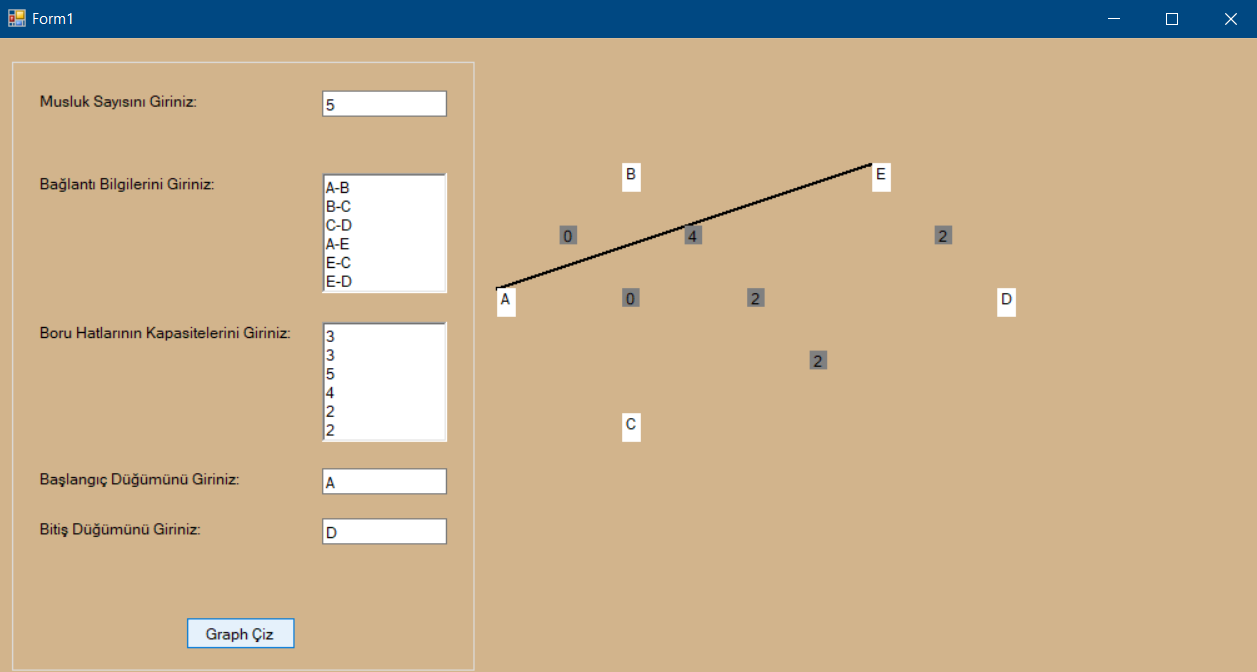
**4. Akış Şeması**



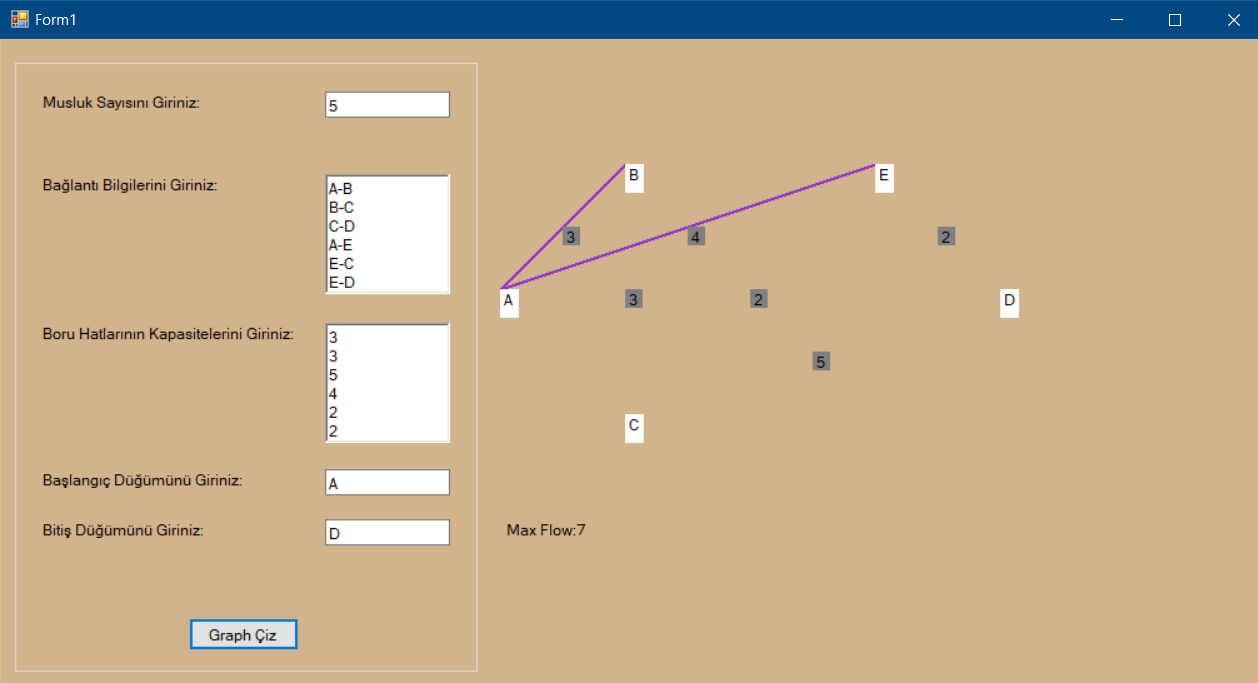
**5**.**Deneysel Sonuçlar**



(Girilen verilere göre graphın çizilmesi)



(Maximum flow probleminin çözülmesi)

****

(Min-cut probleminin çözülmesi)

**6.Kaynakça**

Maximum flow için :

* <http://bilgisayarkavramlari.sadievrenseker.com/2010/05/22/ford-fulkerson-algoritmasi/>
* <https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/>

DFS ve BFS için :

* <https://www.mobilhanem.com/graph-teorisi-dfs-algoritmasi/>
* <https://www.mobilhanem.com/graph-teorisi-bfs-algoritmasi/>

Min-cut için :

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