T.C.
MUĞLA SITKI KOÇMAN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İSTATİSTİK ANABİLİM DALI

GRAFLAR ÜZERİNDE EN KISA YOL ALGORİTMALARININ KARŞILAŞTIRILMASINA İLİŞKİN BİR UYGULAMA

YÜKSEK LİSANS TEZİ

SITKI CANSU

ŞUBAT 2017
MUĞLA

S. CANSU İSTAİTSTİK ANABİLİM DALI YÜKSEK LİSANS TEZİ MUĞLA 2017

T.C.
MUĞLA SITKI KOÇMAN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İSTATİSTİK ANABİLİM DALI

GRAFLAR ÜZERİNDE EN KISA YOL ALGORİTMALARININ KARŞILAŞTIRILMASINA İLİŞKİN BİR UYGULAMA

YÜKSEK LİSANS TEZİ

SITKI CANSU

ŞUBAT 2017
MUĞLA

MUGLA SITKI KOÇMAN ÜNİVERSİTESİ

Fen Bilimleri Enstitüsü

TEZ ONAYI

Sıtkı CANSU tarafından hazırlanan **GRAFLAR ÜZERİNDE EN KISA YOL ALGORİTMALARININ KARŞILAŞTIRILMASINA İLİŞKİN BİR UYGULAMA** başlıklı tezinin, 06/01/2016 tarihinde aşağıdaki jüri tarafından İstatistik Anabilim Dalı'nda yüksek lisans derecesi için gerekli şartları sağladığı oybirliği/~~oyçokluğu~~ ile kabul edilmiştir.

TEZ SINAV JURİSİ

Yrd. Doç. Dr.* Derya DOĞAN DURGUN — İmza:

Matematik Anabilim Dalı,
Celal Bayar Üniversitesi, Manisa

Yrd. Doç. Dr. Nida GÖKÇE** **(Danışman)** — İmza:

İstatistik Anabilim Dalı,
Muğla Sıtkı Koçman Üniversitesi, Muğla

Yrd.Doç. Dr. Nevin GÜLER DİNCER **(Üye)** — İmza:

İstatistik Anabilim Dalı,
Muğla Sıtkı Koçman Üniversitesi, Muğla

ANA BİLİM DALI BAŞKANLIĞI ONAYI

Doç. Dr. Dursun AYDIN — İmza:

İstatistik Ana Bilim Dalı Başkanı,
Muğla Sıtkı Koçman Üniversitesi, Muğla

Yrd. Doç. Dr. Nida GÖKÇE** **(Danışman)** — İmza:

İstatistik Anabilim Dalı,
Muğla Sıtkı Koçman Üniversitesi, Muğla

Savunma Tarihi: 06/01/2017

*Jüri Başkanının ismi birinci sırada verilecektir
**Danışmanın ismi ikinci sırada verilecektir

Tez çalışmalarım sırasında elde ettiğim ve sunduğum tüm sonuç, doküman, bilgi ve belgelerin tarafımdan bizzat ve bu tez çalışması kapsamında elde edildiğini; akademik ve bilimsel etik kurallarına uygun olduğunu beyan ederim. Ayrıca, akademik ve bilimsel etik kuralları gereği bu tez çalışması sırasında elde edilmemiş başkalarına ait tüm orijinal bilgi ve sonuçlara atıf yapıldığını da beyan ederim.

Sıtkı CANSU

…….. / …….. / 2017

(İmza)

ÖZET

GRAFLAR ÜZERİNDE EN KISA YOL ALGORİTMALARININ KARŞILAŞTIRILMASINA İLİŞKİN BİR UYGULAMA

Sıtkı CANSU

Yüksek Lisans Tezi

Fen Bilimleri Enstitüsü

İstatistik Anabilim Dalı

Danışman: Yrd. Doç. Dr. Nida GÖKÇE

Şubat 2017, 49 sayfa

Bu çalışmada en kısa yol problemi için geliştirilen algoritmaların performanslarının karşılaştırılması amaçlanmış ve bu karşılaştırma yapılırken web tabanlı bir arayüz kullanılmıştır. Kıyaslama yapılırken programlama dili olarak C#, Asp.NET ve SQL dilleri kullanılmıştır. İlgili algoritmalar programlama dilleriyle bilgisayar ortamına aktarılmış ve elde edilen bulgularla ileriye yönelik yapılacak çalışmalar için bir ön araştırma yapılmıştır.

Söz konusu algoritmalar bilgisayar diline aktarılırken nesne tabanlı programlama dili olarak C#, web arayüzlü olmasından dolayı Asp.NET ve verilerin bir database sisteminde saklı tutulması için Sql Server kullanılmıştır. Haritalama metodu olarak Google API ve Google Map'ten faydalanılmıştır. Yazılan kodların doğruluğunun sınanması için R programlama dili kullanılmıştır.

Literatürde yaygın olarak kullanılan en kısa yol algoritmalarından Bellman – Ford, Floyd ve Dijkstra algoritmaları incelenmiş ve karşılaştırmalar sonucunda Floyd algoritmasının programlanabilirlik açısından, Dijkstra algoritmasının ise hesaplama açısından daha elverişli olduğu saptanmıştır.

Anahtar Kelimeler: Graf Teorisi, Floyd Algoritması, Dijkstra Algoritması

ABSTRACT

AN APPLICATION FOR COMPARISON OF THE SHORTEST PATH ALGORITHMS ON GRAPH

Sıtkı CANSU

Master of Science (M.Sc.)

Graduate School of Natural and Applied Sciences

Department of Statistic

Supervisor: Yard. Doç. Dr. Nida GÖKÇE

February 2017, 49 pages

In this thesis, we have examined Bellman – Ford, Floyd and Dijkstra algorithms which is commonly used for the shortest – path algorithms in the literature. They purpose to compute the shortest path from each vertex to every other vertex. A web-based interface is used to compare the performances of the algorithms. These algorithms have been transferred to the computer environment by programming languages and a preliminary study has been carried out for future studies with the findings obtained.

While the algorithms transferring to the computer language, C# has been used for object-based programming, Asp.NET has been used because of its web interface and SQL Server has been used to store the data in a database. The mapping has been performed with Google API and Google Map. The correctness of written codes has been evaluated by using "igraph" package in R programming language.

As a result of the comparison, the Floyd algorithm has been found to be more programmable, and the Dijkstra algorithm is more efficient in terms of computing.

Key Words: Graph Theory, Floyd Algorithm, Dijkstra Algorithm

Sevgili Aileme

ÖNSÖZ

Yrd. Doç. Dr. Nida GÖKÇE'ye çalışmalarım sırasında göstermiş olduğu kolaylıklar ve bilimsel bir çalışmanın ve düşünmenin temellerini öğrettiği için teşekkürü bir borç bilirim.

Ayrıca çalışmamın tüm aşamalarında sabırla beni destekleyen değerli anne ve babama minettar olduğumu belirtmek isterim.

İÇİNDEKİLER

ÇİZELGELER DİZİNİ

ŞEKİLLER DİZİNİ

SEMBOLLER VE KISALTMALAR DİZİNİ

G	:	Graf
Δ(G)	:	Düğüm Derecesi
δ(G)	:	Grafın Minimum Derecesi
V	:	Düğümler Kümesi
E	:	Ayrıtlar Kümesi
e_1	:	Ayrıtın Gösterimi
v_1	:	Düğümün Gösterimi
S	:	Kaynak Matrisi
K	:	Komşuluk Matrisi
D_0	:	Uzaklık Matrisi
k	:	İterasyon Sayısı
D_k	:	Uzaklık Matrisinin k. İterasyonu
S_k	:	Kaynak Matrisinin k. İterasyonu
d_{ij}	:	i. Nokta İle j. Nokta Arasındaki Uzaklık
D	:	Bellman – Ford Algoritması İçin Mesafe
Pi	:	Bellman – Ford Algoritması İçin Gidilecek Yol
MYSQL	:	My Structured Query Language
MS-SQL	:	Microsoft Structed Query Language
ADO.NET	:	Activex Data Object
API	:	Application Programming Interface
HTML	:	Hypertext Markup Language
CSS	:	Cascading Style Sheets
ASP	:	Active Server Pages

1. GİRİŞ

Graf Teoremi, birimler arasındaki ilişkileri bir graf üzerinde modellemek için kullanılan matematiksel bir yaklaşımdır. Fen bilimleri, sosyal bilimler ve bilgi sistemleri gibi pek çok alanda karşılaşılan problemleri graflar yardımıyla ifade etmek mümkündür. Özellikle en kısa yol problemlerinin çözümünde önemli bir yere sahiptir. Graf teoremi birimlerin düğümlerle, birimler arasındaki ilişkilerin ise ayrıtlar ya da kenarlarla gösterildiği graflar üzerinden gerçekleştirilen tüm işlemleri genel bir başlık altında ele alır ve en kısa yol problemlerinin çözümünde birden fazla düğüm bulunan bir grafta en kısa yolu ya da en düşük maliyeti belirlemek amacıyla kullanılır.

Temeli 1736'da Leonhard Euler tarafından atılan bu kuram Floyd, Dijkstra ve Bellman – Ford gibi araştırmacılar tarafından zenginleştirilmiş ve pek çok uygulamada başarılı sonuçlar verdiği görülmüştür.

Bu algoritmalarla, yazılım ve navigasyon sistemleri üzerinde graf teorisi uygulamaları geliştirilmeye başlamıştır. Bilgisayar ortamında kullanılan elektronik haritalar ve haritalama programlarında en kısa yol belirlenirken sıklıkla bu algoritmalar kullanılmaktadır. Başlangıçta haritalamalar masaüstü programlarla yapılırken geliştirilen Javascript sayesinde günümüzde çoğunlukla web yazılımları tercih edilmektedir.

Bu çalışmada Floyd, Dijkstra ve Bellman-Ford algoritmaları web tabanlı bir uygulama kullanılarak karşılaştırılmıştır. Uygulamada C# ile birlikte Google API, Javascript, Html ve Css kodları da kullanılmıştır. Aynı zamanda yazılan programın doğruluğunu test etmek için R içerisinde "igraph" paketi kullanılmıştır.

Sözkonusu algoritmalar her ne kadar benzer amaca hizmet etseler de bazı yöntemsel farklılıklara sahiptirler. Çalışmanın mevcut çalışmalardan farkı söz konusu

algoritmaların Google API sayesinde Google Map içerisinde çalışma imkânı sunmasıdır.

2. LİTERATÜR

Graf Teoremi, bir porblemin çözümüne yönelik hazırlanan graf üzerindeki noktalar arasındaki ilişkileri inceleyen ve çoğunlukla düğümler arasındaki en kısa yolu en az maliyetle bulma amaçlayan teoremleri kapsayan matematiksel yaklaşımlar bütünüdür. Bu teoremler kullanılarak navigasyon sistemleri geliştirilmiş ve gerek iş dünyasında gerekse günlük hayatta bu sistemlerin kullanılması maliyet optimizasyonu ve zaman tasarrufu konusunda başarılı sonuçlar elde edilmiştir. Geniş bir uygulama alanına sahip olan en kısa yol algoritmalarını kullanım amaçlarına ve kullanım alanlarına göre farklılık göstermektedirler.

Geliştirilen algoritmalarla beraber teoremin sadece haritalarda değil biyoloji, matematik, tıp ve bilgisayar bilimlerinde de kullanılabileceği ortaya çıkmıştır. Kullanım alanlarının genişlemesi neticesinde yapılan çalışmalar gelişerek teorem robotik nesneler üzerinde uygulanmaya başlamıştır.

Algoritmaların bilgisayar diline geçirilmesiyle beraber işlemlerin hızlanması sağlanmış ve kullanıcı dostu arayüzlerin önemi biraz daha artmıştır. Tarım (2007), çalışmasında harita üzerinde manuel seçim yapılmaksızın Türkiye'deki iller arası en kısa yolu bulmak için web ara yüzlü bir uygulama geliştirmiştir. Bu uygulama içerisinde Google Api kullanılmış olup çözüm Dijkstra algoritmasıyla sağlanmıştır.

İnsansız hava araçları son yıllarda maliyet ve zamandan tasarruf sağlamak amacıyla birçok firma ve devlet tarafından tercih edilen yapay zekaya sahip makinelerdir. İnsansız hava aracıyla (UAV – Unmanned Aerial Vehicles) iki nokta arasında gidilebilecek en kısa yol belirlemesinde yine graf teorisi kullanılmıştır. Sathyaraj ve B. Moses (2008), çalışmasında Bellman – Ford, Dijkstra, Floyd – Warshall ve A* algoritmaları karşılaştırılmış ve insansız hava araçları için en uygun algoritmanın A* olduğu öngörülmüştür.

Graf teorisi sadece haritalar üzerinde değil bilgisayar ağlarında da kullanılmaktadır. Veri iletim hızının önemi giderek artmakta ve veriyi en kısa sürede iletme problemi insanları bu konuda çalışmaya yönlendirmektedir. Ağlardaki bu sıkıntılar protokol yöntemleriyle çözülmeye çalışılmış ve analizlerde graf teorileri kullanılmıştır. Dijkstra ve Bellman – Ford algoritmalarının karşılaştırılmasını inceleyen bir çalışma, veriyi en hızlı şekilde nasıl iletileceğini incelemiş ve sonuç olarak Dijkstra algoritmasının daha avantajlı olduğunu göstermiştir (Hanumanthappa vd., 2010).

Araç rotalama iş yerleri tarafından kullanılan ve iş yerlerine minimum maliyeti sağlayan bir tekniktir. Fakat ilerleyen teknolojiyle beraber bu teknik daha da geliştirilmiş ve gerçek zamanlı trafik bilgisi kullanılarak zaman ve maliyetten tasarrufa gidilmiştir (Bayzan vd., 2011). Trafik yoğunluğu yaşanan şehir merkezlerinde herhangi bir araç, oluşturulan yeni bir rota ile gitmek istediği yere daha hızlı ve daha az maliyetle varabilecektir.

En kısa yol problemlerinde genellikle bilinen parametreler üzerinden gidilmiş ve hesaplamalar bu duruma göre şekillenmiştir. Deng ve Chen (2012), çalışmasında bilinmeyen parametreler üzerinden işlem yepmak üzere Dijkstra algoritması kullanılmış ve karşılaşılan iki problem üzerinde durulmuştur. Bunlardan birisi iki kenarın eklenmesinin nasıl olacağı ve bir diğeri de bulanık sayılarla kenar uzunlukları temsil edilen iki ayrıtın arasındaki karşılaştırmanın nasıl yapılacağıdır (Deng vd.,2012).

Günlük yaşantımızın bir parçası olan robotlar, yine maliyet düşürmek ve zamandan tasarruf sağlamak için üzerinde çokça durulan keşiflerden biridir. Bu mekanik icatlar elektronik ortama geçtikten sonra bilgisayar dili ile üzerlerinde kontrol kurulmuştur. Böylelikle en kısa yol algoritmaları da dâhil olmak üzere birçok bilgisayar destekli deneye tabi tutulmuşlardır. Bu çalışmalardan birisinde de Floyd ve Dijkstra algoritmalarını hibrid bir mobil robota uygulanmştır. Uygulama C# diliyle yazılmış ve aynı zamanda uygulama yapılırken yapay sinir ağları ve genetik algoritmalarda kullanılmıştır (Lui vd., 2012).

Hart, 2013 çalışmasında, Dijkstra ve Bellman – Ford algoritmalarının karşılaştırılması sonucu 425 düğüme sahip bir grafın çözümünde Bellman – Ford

algoritmasının daha hızlı olduğu fakat 425 ve daha fazla genişliğe sahip graf çözümünde Dijkstra algoritmasının daha iyi sonuçlar verdiği görülmüştür (Hart, 2013).

Üretilen malların depolanması ve en kısa süre içerisinde stoklara erişim, büyük firmalar için müşteri memnuniyeti anlamında önem arz etmektedir. Bu nedenle üretilen malların depolanma ve depodan çıkarılma süreci için yapılan bir araştırmaya göre depo içerisinde kullanılan araçlar için bir AGV (Automated Guided Vehicles) üretilebilir ve bu araçta en kısa yol algoritması kullanılabilmektedir. Bilgisayar ortamında yapılan simülasyon sonucu yöntemin Dijkstra algoritmasıyla daha sağlıklı çalıştığı ortaya çıkmıştır (Shaikh vd.,2013).

Floyd algoritmasının daha kısa sürede sonuç verebilmesi için çeşitli yöntemler önerilmiştir. Bu tekniklerden bir tanesi de K – means kümeleme metodudur. Bu teknik Floyd algoritmasını iyileştirmeye yönelik yapılmıştır ve iterasyon sayısı $\theta(n^2m+n^3\log n)$ 'den $\theta(n^3+n^2\log n)$ düşürülmüştür. K – means kümeleme metodu için iki şekilde limitleme vardır. Bunlardan bir tanesi küme merkezleri belirlenirken yüksek ilişki kurulması gerekmektedir. Diğer limit ise küme sayısı önceden belirlenmiş olması gerekmektedir (Lui vd., 2013).

Bir trafik kazası oluştuğu zaman en kısa sürede olay yerine varmak hayat kurtarmakta önemli bir rol oynamaktadır. En kısa yol algoritmasını bulmak her zaman için bir çözüm değildir çünkü müdahale araçlarının zamanlamasını etkileyen pek çok çevre faktörü vardır. Bu problemin üzerine yapılan bir çalışmada kaza anında acil müdahale aracı için en kısa yolun bulunmasının yanı sıra gerçek zamanlı bir trafik gözleyici uygulama oluşturulmuştur (Kai vd., 2014).

Olası bir afet anında acil çıkış yolu belirleme problemi yine graf teorisiyle çözümlenmiş bir problemdir. Özellikle binalarda meydana gelen hasar sonucu kapanan acil çıkış yoluna alternatif bir güvenli yol oluşturmak graf teorisinde kullanılan algoritmalarla sağlanabilmektedir. Güvenli olmayan çıkış yolu yerine en kısa yol algoritmalarıyla alternatif yol üretilerek hızlı bir şekilde insanlar güvenli bölgeye ulaştırılmaktadır (Jeyhun vd., 2014).

2015 yılında yapılan bir çalışmada yılansı robotlarda yörünge analizi incelenmiştir (Yıldırım vd., 2015). Çalışma içerisinde yılansı robotların daha hızlı hareket edebilmeleri için akıllı nesneler algoritması geliştirilmiştir.

Trafik yoğunluğu sonucunda taksi kullanım oranı yükselmiş ve böylelikle yeni problemler doğmasına neden olmuştur. Yapılan çalışmada Graphic Information System (GIS) uygulaması geliştirilmiştir. Bu program gerçek zamanlı yol belirlemede taksicilere yol gösterici olmuş uzun zaman kendisini bekleyen müşterilere daha kısa süre içerisinde hizmet vermeyi amaçlamıştır. Uygulama, en kısa yol belirlenmesinde Dijkstra algoritmasını kullanmaktadır (Indrajaya vd., 2015).

Tren yolu, hem şehirlerarası hem de şehir içi ulaşımda önemini halen korumaktadır. Çoğunlukla zamandan tasarruf sağlamak amacıyla insanlar tren yolunu tercih etmektedir. Böylelikle maksimum faydayı sağlamak amacıyla en kısa yol algoritmaları tren yolu üzerine de kurgulanmıştır. Bu amaçla yapılan çalışmada geleneksel Dijkstra algoritması geliştirilmiş, böylelikle hibrid genetik algoritması ortaya çıkmıştır. Tren istasyonlarında meydana gelen yoğunluğun önüne geçilebilmek amacıyla alternatif bir yol belirleyen bu algoritma ilgili problemin ortadan kaldırılmasına yardımcı olmaktadır (Tong vd., 2015).

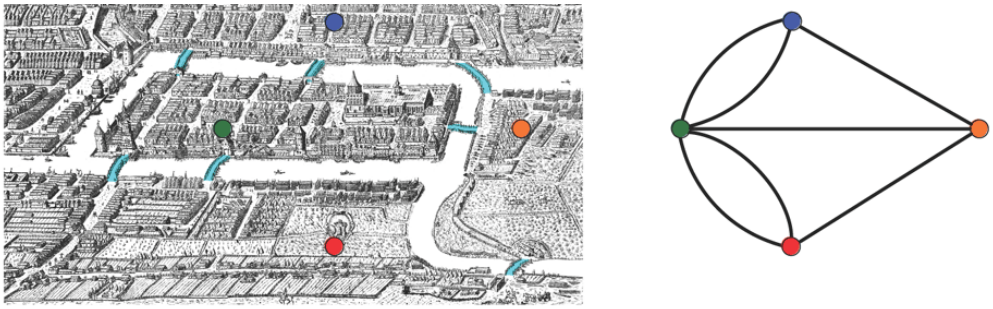
Bu çalışmaların kullanıcı dostu bir arayüze ihtiyaç duymasından dolayı tez içerisinde yazılan program tamamen mobil uyumlu ve responsive bir web sitesidir. Program içerisinde algoritmalar için veri tabanı kullanılmış böylelikle Floyd ve Dijkstra algoritmalarının arasındaki hesap zamanı farkının minimum seviyeye indirileceği düşünülmüştür.

3. GRAF TEORİSİNE GENEL BAKIŞ

Grafik kuramı 18. Yüzyılın ortalarından itibaren gerek matematiksel hesaplamaların modellenmesinde gerekse tıp alanında alanındaki araştırmacıların ilgi odağı olmuştur. Aynı zamanda Navigasyon sistemlerinin ve en kısa yol algoritmalarının temelini oluşturmaktadır. Bilgisayar bilimlerinin günlük hayattaki öneminin anlaşılmasıyla beraber bu algoritmaların önemi de artmaya başlamıştır.

1736 yılında Euler'in Königsberg kentindeki Eski ve Yeni Pregel nehirlerinin birleştiği bölgede şehri dört bölüme ayıran yedi köprüden (Şekil 3.1.) esinlenerek ortaya attığı bu teorem geliştirilerek günümüzde navigasyon sistemleri, tıp bilimi, mühendislik, fizik ve kimya gibi temel bilimlerde kullanılmaktadır.

Königsberg problemi, 'Pregel nehrinde birbirine ve kıyılarına bağlı iki ada arasındaki köprülerden bir ve yalnız bir defa geçmek koşulu ile bir tur yapılabilir mi?' sorusuna cevap arar. (Şekil 3.1.)



Şekil 3.1. Königsberg köprüsü (Cempau vd., 2011)

Königsberg köprüleri problemi her bir kara parçası bir düğüm ve söz konusu kara parçalarını birleştiren köprü bir ayrıt olarak ele alınarak basit bir grafa indirgenebilir. Euler böylesi bir graf üzerinden yaptığı çalışmalar neticesinde bu problemin

çözümünün olmadığını ispatlamış ve "Euler Teoremi"ni ortaya atmıştır. Bu teoreme göre düğümlerden birden fazla, kenarlardan sadece bir kere geçmek üzere oluşturulacak bir graf için geçerli kural bütün düğüm derecelerinin (Bir düğüme bağlı ayrıt sayısına düğüm derecesi denir (Caldwell,1995)) çift olmasıdır. Bir grafta Euler yolu olmasının başka bir şartı ise iki düğüm hariç olmak üzere diğer bütün düğümlerin derecelerinin çift olmasıdır (Tseng vd., 1998).

Tanım 1: G = { V, E} yönsüz bir graf olmak üzere, bütün düğümleri dolaşan bir yol varsa bu yol "Euler Yolu" olarak adlandırlır (Atallah,1984).

Euler Yolu'nun özellikleri

- G grafındaki bir Euler yolu G'nin tüm ayrıtlarını kenar olarak bir kere içeren kapalı bir yoldur.
- Kapalı bir Euler yolu bir Euler devresidir.
- Bir graf içerisinde en az bir Euler yolu barındırıyorsa bu graf bir Euler grafıdır.
- Euler yolu tüm ayrıtları sadece bir kez içerir fakat düğümlerden birden fazla geçilebilir.

Tanım 2: Çevrim içermeyen bağlı bir G grafı sadece ve sadece tüm düğüm dereceleri çift ise Euler grafıdır.

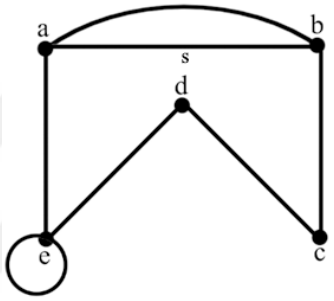
Tanım 3: Bağlı ve çevrim içermeyen Euler olmayan bir G grafında ancak ve ancak tam olarak iki dereceli düğüm var ise bir Euler yolu vardır.

Königsberg problemi bir graf üzerinde modellendiğinde düğüm derecelerinin hepsinin çift olmadığı görülmektedir. Böyle bir durumda Euler Teoremi gereğince Königsberg köprüsü probleminin çözümü yoktur.

Graf, bir problem, olay veya ifadenin düğüm ve çizgiler kullanılarak gösterilmiş şeklidir. Çözüm aranan problem, graf teorisi kullanılarak tanımlanabilirse, o problem için algoritmik bir durum elde edilmiş olunur. Bu sayede graf teorisine ait olan özellikler ve çözüm yöntemleri, söz konusu problemin çözümü için de kullanılabilir (Çölkesen, 2000)

Tanım 4: Sınırlı sayıdaki V düğümler kümesi ile yine sınırlı sayıdaki E ayrıtlar kümesinden oluşan (V, E) yapısına graf denir. Diğer bir deyişle bir graf G={V,E} kümelerinden oluşur. E={e_1, e_2, e_3,….., e_m} kümesinin elemanlarına ayrıt, V={v_1, v_2, v_3,….., v_n} kümesinin elemanlarına düğüm denir. V_1 ve v_2 düğümlerini birbirine bağlayan e_1 ayrıtının oluşturduğu görsele graf adı verilir (Saran, 2008)

Bir grafın derecesi içerisindeki düğümlerden derecesi en büyük olan değerle ifade edilir ve genellikle Δ(G) olarak gösterilir. G grafının minimum derecesi o grafa ait en küçük dereceli düğümün derecesidir δ(G) ile gösterilir (Cormen vd., 2001).

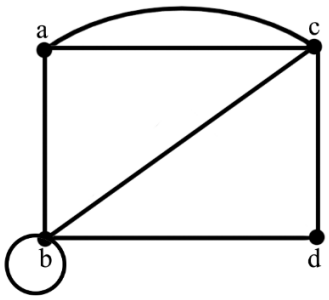


Şekil 3.2. Graf Örneği

Şekil 3.2.' de verilen G grafının düğümler kümesi V = {a, b, c, d, e } ve kenarlar kümesi E = {(a,b), (b,c), (c,d), (d,e),(e,a)}'dir.

Tanım 5: Basit bir G = {V, E} grafı, boş olmayan V düğümler kümesine ve düğümler kümesinin elemanlarını sıralı olma özelliğine bakmaksızın bağlayan veya ilişkilendiren E kenarlar kümesine sahiptir (Çölkesen, 2000).

Bir kenarın oluşabilmesi için en az iki tane düğüm olması gerekir. Şekil 3.2.' de V(a) ile V(b) arasındaki bağlantıyı s ayrıtı sağlamaktadır ve s = (a, b) şeklinde gösterilir. Eğer iki düğüm arasını bağlayan birden fazla ayrıt var ise bu ayrıtlara "paralel ayrıtlar" denir. G grafında V(a) ile V(b) düğümleri arasında iki tane ayrıt bulunmaktadır.

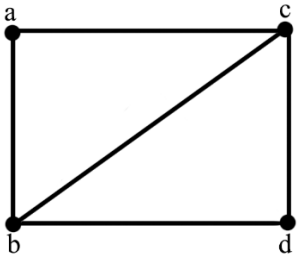


Şekil 3.3. Paralel Kenar ve Döngü

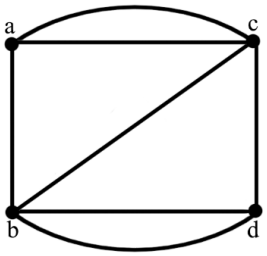
Şekil 3.3.' te verilen G grafında a ve c düğümlerine ait bir paralel kenar mevcuttur. V(b) düğümünde bir tane ayrıtın başlangıç ve bitiş düğümünün aynı olduğu görülmektedir, bu tür ayrıtlara "döngü" denilmektedir.

3.1. Graf Çeşitleri

Döngü ve paralel kenar içermeyen, yönsüz olan graflara "basit (simple) graf" denilmektedir (Şekil 3.4.). İki düğüm arasında birden çok ayrıt içeren graflara ise "çoklu (multi) graf" denilir (Şekil 3.5.).



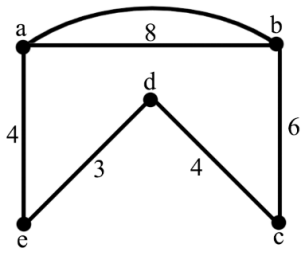
Şekil 3.4. Basit Graf



Şekil 3.5. Çoklu Graf

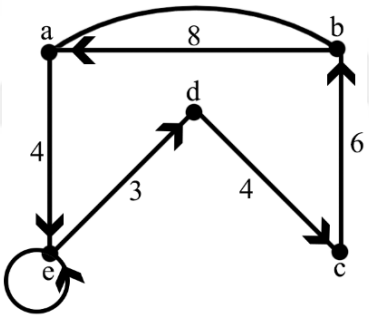
Düğümler arası ayrıt uzunluklarının arasında farklılık var ve bu farklılık da graf üzerinde gösterilmiş ise bu tip graflara "ağırlıklı graf" denir. (Şekil 3.6.)

Tanım 6 : G = (V,E) bir graf olmak üzere, V düğümlerine bağlı E ayrıtlarına ait her değere "ağırlık" denir ve ilgili graf da "ağırlıklı graf" olarak adlandırılır (Joyner vd., 2013).



Şekil 3.6. Ağırlıklı Graf

Bir G grafında düğümler ayrıtlarla ilişkilendirilmiş ve yön bildirir bir durumdaysa bu tür graflar yönlü graf olarak adlandırılmaktadır (Şekil 3.7.).



Şekil 3.7. Yönlü Graf

3.2. Komşuluk Matrisi

Düğümler arasındaki ilişkileri göstermek için kullanılan kare matrise komşuluk matrisi denir. Bu matrise ait satır ve sütunlardaki elemanlar ise iki düğüm arasındaki maliyetleri temsil etmektedir. Çizelge 3.1.' de 5 düğümlü basit bir yönsüz grafa ilişkin komşuluk matrisi örneği verilmiştir. Komşuluk matrisi yönsüz graflarda simetriktir, yönlü graflar simetrik olmayabilir.

Çizelge 3.1. Şekil 3.6.' nın Komşuluk Matrisine Dönüşümü

	A	**B**	**C**	**D**	**E**
A	0	8	∞	∞	4
B	8	0	6	∞	∞
C	∞	6	0	4	∞
D	∞	∞	4	0	3
E	4	∞	∞	3	0

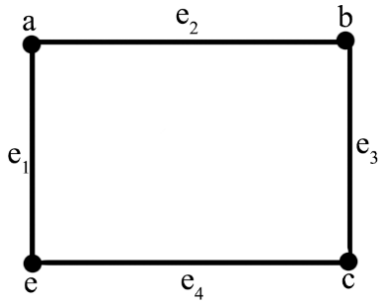
3.3. Çakışım Matrisi

Bir grafa ait çakışım matrisinde her satır bir düğüme her sütun da bir ayrıta karşılık gelir. Matrisin elemanları düğümler arasında bağlantı olup olmadığını göre sırasıyla 1 ve 0 değerlerini alır.

Tanım 7 : Bir G = (V, E) grafında V = $\{v_1, v_2, v_3 \ldots\ldots v_n\}$ ve E = $\{e_1, e_2, e_3 \ldots\ldots e_m\}$ olsun.

$$b_{ij} = \begin{cases} 1, v_i \text{ düğümü } e_j \text{ ayrıtı ile çakışıksa} \\ 0, \quad \text{aksi takdirde} \end{cases}$$

olmak üzere B = $[b_{ij}]_{n \times m}$ matrisine G'nin çakışım matrisi denir (Ulukan vd., 2015).



Şekil 3.8. Çakışım matrisine ilişkin bir graf örneği

Çizelge 3.2. Şekil 3.8.' deki grafiğin çakışım matrisine dönüşümü

	e_1	e_2	e_3	e_4
a	1	1	0	0
b	0	1	1	0
c	0	0	1	1
d	1	0	0	1

3.4. En Kısa Yol Algoritmaları

3.4.1. Floyd algoritması

Floyd Warshall algoritması 1962 yılında Robert Floyd tarafından önerilmiştir. Floyd algoritması önceden belirlenmiş bir başlangıç düğümünden graf üzerindeki diğer düğümlere giden en kısa yolu bulmayı amaçlayan bir algoritmadır (Floyd R. W., 1962) .

Graf üzerindeki bütün düğüm çiftleri arasındaki en kısa yol, algoritmanın sadece bir kere çalıştırılmasıyla elde edilir ve algoritma, negatif ağırlıklı graflar içinde kullanılabilir (Gajbhiye, 2013).

Tanım 8 : G bir graf ve V(G) boş olmayan sonlu bir küme olsun. V(G) içerisinde oluşturulan sıralı çift E(G) olmak üzere G = (V(G), E(G)) şeklinde ifade edilebilir ve G={E,V} olarak kısaltılabilir. Sonlu küme olan $V = \{v_1, v_2, v_3, \dots \dots v_n\}$ G'ye ait düğümleri, $E = \{e_1, e_2, e_3, \dots \dots e_m\}$ G'ye ait ayrıtları temsil etmek üzere ve E'deki her e_k elemanı ($1 \leq k \leq m$) V(G) içerisinde sıralı bir çift v_i ve v_j ($1 \leq i, j \leq m$) olsun. G = (V, E, W) ağırlıklandırılmış bir graf olsun. v_i ve v_j G grafında iki düğüm olmak üzere G'ye ait (v_i, v_j) minimum uzaklık, en kısa (v_i, v_j) yol olarak adlandırılır (Dachuan, 2010).

Floyd algoritması grafa ait bir komşuluk matrisi (K) ve kaynak matrisi (S) oluşturulmasına ihtiyaç duyar. Komşuluk matrisi, iki düğüm arasındaki mesafenin maliyetini göstermektedir ve n x n boyutunda bir kare matristir ($K_{n \times n}$). Satırlar başlangıç noktasını, sütunlar bitiş noktasını temsil etmektedir. Dolayısıyla komşuluk

matrisi aynı zamanda simetrik bir matristir. Seçilen düğümlerin başlangıç ve bitiş noktaları aynı ise matrisin ilgili elemanı "sıfır (0)" ile gösterilirken düğümler arasında bir yol yok ise "sonsuz (∞)" olarak ifade edilir. Kaynak matrisi ise başlangıç noktasından bitiş noktasına giderken izlemesi gereken yolları barındırır. Her iterasyon sonucunda bir değişim söz konusu olduğunda iterasyon numarası matriste ilgili hücre için değiştirilir.

Çizelge 3.3. Floyd algoritmasının işlem adımları (Çölkesen R., 2000)

```
Adım 1 : Komşuluk matrisi üzerindeki maliyetleri başlangıç maliyeti olarak
kabul et
Adım 2 : Rota matrisine boş anlamında değerler yerleştir. (ROTA = -1)
Adım 3 : Düğümlerin kendilerine ulaşım maliyetlerini sıfırla (UM[i][i]=0)
for (aradüğüm sayacı < düğüm sayısı) {
  for (satır sayacı < düğüm sayısı) {
    for (sütun sayacı < düğüm sayısı ) {
      if (ara düğüm üzerinden gitmek daha kısa ise) {
      aradüğümlü maliyeti en küçük maliyet kabul et;
      Rota bilgisini güncelle}
    }}}
```

Çizelge 3.4. Floyd algoritmasının pseudo kodları

```
for i = 1 to N
  for j = 1 to N
    "eğer i düğümünden j düğümüne bir ayrıt varsa"
      dist[0][i][j] = "i noktasından j noktasına olan mesafe     hücreye atılır"
    else
      dist[0][i][j] = INFINITY "eğer yoksa sonsuz değeri giriliyor"
```

Çizelge 3.4.' teki kodlarda N sayısı iterasyon, i satır sayısı, j sütun sayısını ifade eder. Buradan yola çıkarak komşuluk matrisi Çizelge 3.5.' teki gibi oluşturulur. Daha

sonra matris içerisinde her bir hücre gezilerek minimum maliyette çözüme ulaşılması hedeflenmektedir.

Çizelge 3.5. Floyd algoritmasının komşuluk matrisini oluşturma kodu

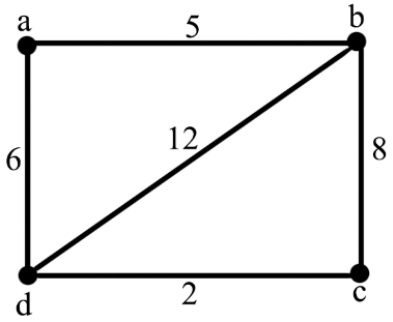
```
for k = 1 to N
  for i = 1 to N
    for j = 1 to N
      dist[k][i][j] = min(dist[k-1][i][j], dist[k-1][i][k] + dist[k-1][k][j])
```

Algoritmanın avantajları ve dezavantajları;

- Kodlama için kolay bir algoritmadır. İşlem sayısının azlığı ve döngülerdeki sadelik programcılar için tercih sebebi olmaktadır.
- Bütün düğümler için en kısa yol çözümünü vermesi de önemli bir avantajdır.
- Minimum çözüme ulaşırken iterasyon sayısını temel alması dezavantaj olarak görülmektedir. Bunun sebebi, 500 düğümü olan bir grafta optimum çözüme ulaşmak hedefleniyorsa algoritma 500 defa çalıştırılmalıdır ve bu da algoritmayı yavaşlamasına sebep olmaktadır

3.4.1.1. Floyd algoritması için bir örnek

Bu bölümde Floyd algoritmasının işlem adımlarının Şekil 3.9.' daki graf üzerinde basit bir uygulaması verilmektedir. Grafta her düğüm bir şehri ve her ağırlıklandırılmış ayrıt düğümler arası mesafeyi göstermektedir.



Şekil 3.9. Floyd algoritması için örnek grafik

Adım 1 :

Şekil 3.9.‘ da verilen graf, komşuluk matrisine (K_0) aktarılır. Bu matris düğümler arasındaki mesafeyi göstermektedir. Bir düğümden diğer düğüme gidilirken hangi noktadan geçileceğini belirlemek için S_0 matrisi oluşturulur. Birinci aşamada iterasyon sayısı 0’dır.

Eğer grafta paralel kenar varsa, bu kenarlardan ağırlıkça en küçük olanı seçilir ve diğeri yok sayılarak işlemlere devam edilir. Eğer graf k düğümden oluşuyorsa k x k boyutunda bir kare matris olması gerekmektedir.

Problemin çözümünde kullanılacak notasyonlar aşağıdaki gibidir;

$k,i,j \in N$ olmak üzere;

k: İterasyon sayısı

K_k: Komşuluk tablosunun k. İterasyonu

S_k: Kaynek tablosunun k. İterasyonu

d_{ij}: i. nokta ile j. nokta arasındaki uzaklığı gösterir olsun

Çizelge 3.6. Şekil 3.9’ daki örneğe ait (D_0) ve S_0 matrisleri

D_0	a	b	c	d	S_0	a	b	c	d
a	0	6	5	∞	**a**	0	2	3	4
b	6	0	12	7	**b**	1	0	3	4
c	5	12	0	8	**c**	1	2	0	4
d	∞	7	8	0	**d**	1	2	3	0

Adım 2 :

Tanımlamaların ardından birinci iterasyona geçilebilir. k = 1 durumunda D matrisindeki 1. satır 1. sütun kapatılır. Geriye kalan her bir hücre aşağıdaki şart ile kontrol edilir. Eğer şart sağlanıyorsa ilgili hücre güncellenir, sağlamıyorsa herhangi bir işlem yapılmadan devam edilir.

$$d_{ij} > d_{ik} + d_{kj} \tag{3.1}$$

i = 2, j = 3 ve k = 1 olmak üzere D matrisinde ilgili hücrelerin değerlerine bakılır.

$$d_{23} > d_{21} + d_{13} \gg 12 > 6 + 5 \tag{3.2}$$

3.2 numaralı denklemde şart sağlandığından dolayı d_{23} hücresi 11 değerini alır ve aynı işlem diğer hücredeki elemanlara da uygulanır. Diğer yandan ilgili değişiklik S matrisinde uygulanır ve D matrisinde güncellenen hücreye karşılık gelen S matrisine ait hücre o andaki iterasyon numarasıyla güncellenir. Bu durumda S_{23} hücresinin değeri 1 olacaktır. Yukarıdaki işlemler iterasyon sayısı düğüm sayısına eşit olana kadar sürdürülür. Böylelikle bütün düğümler için en kısa yol bulunmuş olur.

1.İterasyon

i = 2, j = 3, k = 1;

Çizelge 3.7. Şekil 3.9' daki örneğe ait birinci iterasyon tablosu

D_1	**a**	**b**	**c**	**d**	**S_1**	**a**	**b**	**c**	**d**
a	0	6	5	∞	**a**	0	2	2	4
b	6	0	12	7	**b**	1	0	2	4
c	5	12	0	8	**c**	1	2	0	4
d	∞	7	8	0	**d**	1	2	3	0

Çizelge 3.8. Çizelge 3.7' de verilen tablonun çözümü

D_1	**a**	**b**	**c**	**d**	**S_1**	**a**	**b**	**c**	**d**
a	0	6	5	∞	**a**	0	2	2	4
b	6	0	11	7	**b**	1	0	1	4
c	5	11	0	8	**c**	1	1	0	4
d	∞	7	8	0	**d**	1	4	3	0

$d_{23} > d_{21} + d_{13} \gg 12 > 6 + 5$ → Şart sağlanır ve hücrede değişiklik yapılır. Benzer şekilde diğer iterasyonlar için de aynı işlemler uygulanır ve optimum sonuca ulaşılır.

2. İterasyona ait tablo

Çizelge 3.9. Şekil 3.9' da verilen örneğe ait ikinci iterasyon tablosu

D_2	a	b	c	d	S_2	a	b	c	d
a	0	6	5	∞	**a**	0	2	2	4
b	6	0	11	7	**b**	1	0	1	4
c	5	11	0	8	**c**	1	1	0	4
d	∞	7	8	0	**d**	1	4	3	0

3. İterasyona ait tablo

Çizelge 3.10. Şekil 3.9.' da verilen örneğe ait üçüncü iterasyon tablosu

D_3	a	b	c	d	S_3	a	b	c	d
a	0	6	5	13	**a**	0	2	2	2
b	6	0	11	7	**b**	1	0	1	4
c	5	11	0	8	**c**	1	1	0	4
d	13	13	8	0	**d**	1	2	3	0

4. İterasyona ait tablo

Çizelge 3.11. Şekil 3.9.' da verilen örneğe ait dördüncü iterasyon tablosu

D_4	a	b	c	d	S_4	a	b	c	d
a	0	6	5	13	**a**	0	2	2	2
b	6	0	11	7	**b**	1	0	1	4
c	5	11	0	8	**c**	1	1	0	4
d	13	13	8	0	**d**	1	2	3	0

Sonuç olarak tablodan iki nokta arasındaki en kısa mesafeyi bulmak için öncelikle başlangıç noktası (a düğümü) ve bitiş noktası belirlenir (d düğümü). S_0 tablosuna bakıldığında "a" ve "d" düğümlerinin kesiştiği noktada 2 numaralı düğüm (b

düğümü) olduğunu görülmektedir. Bu bize a düğümünden d düğümüne gidebilmek için b düğümünden geçilmesi gerektiğini gösterir. Böylelikle "a" düğümünden "d" düğümüne olan en kısa yol "a → b → d" şeklindedir.

3.4.2. Dijkstra algoritması

Dijkstra algoritması bir başlangıç düğümünden bitiş düğümüne kadar en kısa yolu bulmayı amaçlayan bir algoritmadır. Algoritma günümüzde GPS ve Navigasyon sistemlerinde kullanılmakta ve birçok oyun uygulamasında tercih edilmektedir.

Dijkstra algoritması en kısa yolu bulurken Greedy yaklaşımını kullanır. Greedy yaklaşımı bir düğümden diğer düğüme giden olası en iyi çözümü araştırır. Yani başlangıç düğümünden bir sonraki düğüme geçişte olası düğümlerle arasında kalan mesafeyi minimum yapmayı amaçlar.

Dijkstra algoritmasının iterasyon sayısı O ($|V|^2+|E|$) = O ($|V|^2$) şeklinde ifade edilebilir (Bahar vd., 1997)

Çizelge 3.12. Dijkstra algoritmasının adımları (Bahar vd., 1997)

Adım 1. Bir uzaklık listesi, bir önceki düğüme ait dizin, ziyaret edilen ve seçilen düğümlerin listesi oluşturulur.
Adım 2. Başlangıç düğümü "sıfır" olarak, geriye kalan bütün düğümler "sonsuz" olarak işaretlenir.
Adım 3. Ziyaret edilen bütün düğümler "sıfır" bir önceki düğüme ait olan bütün liste "null" olarak belirlenir.
Adım 4. Geçerli düğüm başlangıç düğümü olarak işaretlenir ve başlangıç düğümü ziyaret edilmiş gibi kabul edilir.
Adım 5. Başlangıç düğümüne en kısa mesafede olan düğümü seçildikten sonra uzaklık ve bir önceki düğüme ait olan listeyi güncellenir
Adım 6. Bütün düğümler ziyaret edilene kadar 4. Adım tekrarlanır.

Çizelge 3.13. Dijkstra algoritmasının pseudo kodu (Gajbhiye, 2013)

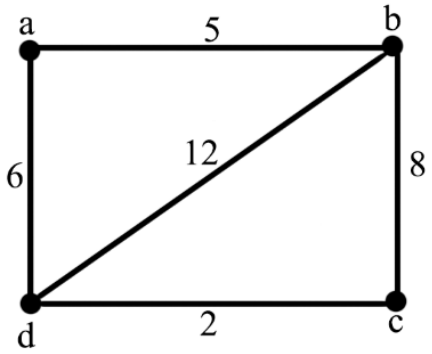
```
Dijkstra (G, w, s)
        d[s] = 0
        for each v ∈ V – {s}
            do d[v] = ∞
        S = ∅
        Q = V
        While Q ≠ ∅
            Do u = ExtractMin(Q)
            S = S ∪ {u}
            For each v ∈ adj {u}
                    Do if d[v] > d[u] + w(u, v)
                            Then d[v] = d[u] + w(u, v
```

Algoritmanın avantajları ve dezavantajları;

- Başlangıç düğümünden bitiş düğümüne olan en kısa yolun yanı sıra geriye kalan tüm düğümlere olan en kısa yollarında kolaylıkla bulabilir.
- Büyük ölçekli problemlerde Dijkstra algoritması hesaplama yükü açısından Floyd algoritmasına göre daha avantajlıdır.
- Programlama açısından daha zordur (Sathyaraj vd., 2008,).

3.4.2.1. Dijkstra algoritması için bir örnek

Graftaki (Şekil 3.10.) noktalar şehirleri noktalar arası çizgiler şehirlerarası yolları gösterir olsun. Graf, yönsüz bir graftır ve döngü içermemektedir. Dijkstra algoritmasında öncelikle yapılması gereken, başlangıç düğümü sıfır olmak üzere diğer bütün düğümleri sonsuz olarak işaretlemektir. Problem, bir tablo yardımıyla çözümlenecektir.



Şekil 3.10. Dijkstra algoritması için örnek grafik

Problemin amacı a düğümünden d düğümüne giden en kısa yolu bulmaktır. Grafa göre tablo aşağıdaki gibi olacaktır;

Çizelge 3.14. Şekil 3.10.' daki grafın ilk adım tablosu

	A	**B**	**C**	**D**
a	0	∞	∞	∞

Çizelge 3.14' te görüldüğü gibi bütün düğümlerin isimleri, sütunları oluşturulur ve başlangıç düğümü hariç diğer bütün düğümler sonsuz olarak işratelenir. Sonraki adımda satırdaki en küçük değer seçilir ve "min(mesafe, işaretli değer + graf üzerindeki mesafe)" formülü uygulanır. "a" düğümündeki değer en küçük değer olduğu için "sıfır" değeri ilk seçilen değer olacaktır. Şimdi diğer bütün düğümler için ilgili formül uygulanmalıdır.

Çizelge 3.15. Şekil 3.10.' daki "a" düğümü için çözümleme

Formül	**Min (TabloMesafesi, İşaretliDeğer + GrafMesafesi)**
A – B	Min(∞, 0 + 6) = 6
A – C	Min(∞, 0 + 5) = 5

"a" düğümü ile "d" düğümü arasında herhangi bir yol olmadığı için "sonsuz olarak kalacaktır ve tablo aşağıdaki şekle dönüşecektir.

Çizelge 3.16. Şekil 3.10.' daki "a" düğümü çözümlemesi sonucu oluşan tablo

	a	**b**	**c**	**d**
a	0	∞	∞	∞
c	0	6	5	∞

"a" düğümüne ait sıfır değerini daha önceden seçildiği için şimdi diğer düğümler arasında en küçük değer olan "5" değeri seçilir.

Çizelge 3.17. Şekil 3.10.' daki "c" düğümü için çözümleme

Formül	**Min (TabloMesafesi, İşaretliDeğer + GrafMesafesi)**
C – B	Min(6 ,5 + 12) = 6
C – D	Min(∞ , 5 + 4) = 9

"d" düğümü güncellenerek dokuz değerini alır ve işlemlere devam edilir.

Çizelge 3.18. Şekil 3.10.' daki "c" düğümü çözümlemesi sonucu oluşan tablo

	a	**b**	**c**	**d**
a	0	∞	∞	∞
c	0	6	5	∞
b	0	6	5	9

Önceden seçilmiş değerler haricindeki en küçük değeri seçilir ve tekrar formüle edilir.

Çizelge 3.19. Şekil 3.10.' daki "b" düğümü için çözümleme

Formül	**Min (TabloMesafesi, İşaretliDeğer + GrafMesafesi)**
B – D	Min(9 ,6 + 7) = 9

"d" düğümüne ait değerden daha küçük bir değer bulunamadığı için değer güncellenmez ve olduğu gibi bırakılır.

Çizelge 3.20. Şekil 3.10.' daki "b" düğümü çözümlemesi sonucu oluşan tablo

	a	**b**	**c**	**d**
a	0	∞	∞	∞
c	0	6	5	∞
b	0	6	5	9
d	0	6	5	9

Tablonun son hali yukarıdaki gibidir. Şimdiki adımda en kısa yol tablo üzerinden bulunmalıdır. Bunun için çizelge 3.21.' deki yöntem uygulanır;

Çizelge 3.21. Dijkstra algoritmasına ait yol izleme adımları

Adım 1 : Hedef düğüme ait satırı seç
Adım 2 : Bir satır yukarıya çık
Adım 3 : Eğer değer değişmişse ilgili satırdaki işaretlenen düğüme geç
Adım 4 : Kaynak düğüme ulaşıncaya kadar Adım 2 ve Adım 3'ü uygula.

Çizelge 3.22. Şekil 3.10.' daki örnek için yol bulma tablosu

	a	**b**	**c**	**d**
a	0	∞	∞	∞
c	0	6	5	∞
b	0	6	5	9
d	0	6	5	9

Çizelge 3.17' de öncelikle hedef düğüm olan "d" düğümüne ait satır alınır ve bir üst satıra bakılır. Değerde değişiklik olmadığı için bir üst değere bakılır. "c" düğümüne ait hücrede değişiklik olduğu için ilgili satırdaki son işaretlenen değer dikkate alınarak hedef düğüme ulaşılır. Böylelikle en kısa yol "A → C → D" şeklinde olur.

3.4.3. Bellman – Ford algoritması

Bellman – Ford algoritmasının temel çalışma mantığı, bir tek kaynak nokta yerine tüm noktaları birer kaynak gibi düşünerek bunlar üzerinden bağlantıları hesaplamaktır (Demirkol, 2003). Bellman – Ford algoritmasının işlem adımları Dijkstra algoritmasıyla aynıdır, aralarındaki tek fark Bellman – Ford Algoritmasının negatif ağırlıklı grafları da çözebilmesidir (Ulukan, 2015). Aynı zamanda Bellman – Ford algoritması yönlü graflar için tasarlanmıştır. Algoritma ilk olarak 1955 yılında Shimbel tarafından önerilmiştir. Daha sonra 1956 ve 1958 yılları arasında Richard Bellman ve Lester Ford tarafından yayınlanmıştır.

Çizelge 3.23. Bellman – Ford Algoritmasının İşlem Adımları

Adım 1: İlk adımda başlangıç düğümü atanır ve sıfır (0) olarak işaretlenir. Adım 2: Diğer bütün düğümlere gidiş maliyeti olarak sonsuz değeri verilir. Adım 3: Bu adımda en kısa mesafeler hesaplanır. Graftaki düğüm sayısının bir eksiği kadar (|V| - 1) aşağıdaki döngü çalıştırılır. Eğer mesafe[v] > mesafe[u] + ayrıt ağırlığı mesafe[v] = mesafe[u] + ayrıt ağırlığı

Çizelge 3.24. Bellman – Ford Algoritmasının Pseudo Kodları (Bahar vd., 1997)

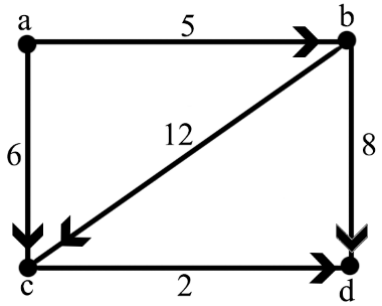
```
Bellman_Ford (A_G, s){
    S_s = Extract_Source (A_G, s);
    For (i=|V| - 1 ; i ≥ 0 ; i++){
        if (Negative_Else_Branch S_s)){
                "Print("Yol Negatif Çember İçermektedir" )
                Break;
        }
        w= SR_Matrix_Multiply (S,        A_G)
        if (S_S = = w) {
                Print ("s Kaynağından En Kısa Yol  Bulundu")
                Break;          }
               S_s = w;
                  }
            Return S_s;
         }
```

Algoritmanın avantajları ve dezavantajları;

- Negatif ağırlıklı graflar için geliştirilmiş olması algoritmanın başlıca avantajlarından birisidir.
- Dijkstra algoritmasına göre daha yavaş çalışır.
- Bellman – Ford algoritması Dijkstra algoritmasıyla aynı yaklaşımı kullanmaktadır. Bu en kısa yol algoritması $|V| - 1$ iterasyon sayısıyla çözüme ulaşabilir (Leiserson, 2009).

3.4.3.1. Bellman – Ford algoritması için bir örnek

Şekil 3.11' de Bellman Ford algoritmasının daha iyi anlaşılabilmesi için bir örnek verilmiştir. Örnekteki graf yönlü graftır ve her nokta şehirleri, noktalar arasındaki her çizgi şehirlerarasındaki yolları ifade etmektedir.



Şekil 3.11. Bellman – Ford algoritması için örnek grafik

Öncelikle başlangıç düğümü seçilir (a) ve başlangıç düğümü haricindeki bütün düğümler "sonsuz" olarak işaretlenir.

Çizelge 3.25. Şekil 3.11.' daki örneğin ilk adım tablosu

	a	**b**	**c**	**d**
d	0	∞	∞	∞
pi				

Çizelge 3.25.' te her sütun graftaki düğümleri, satırlardaki "d" düğümler arası mesafeyi "pi" ise bir noktadan diğerine giderken hangi düğümden geçileceğini göstermektedir. İlk olarak "A – B" arasındaki mesafeye bakılır. Mesafeler arasında aranan şart ise"min(mesafe[v] > mesafe[u] + ayrıt ağırlığı)" 'dır. Bu durumda ∞ > 0 + 5 şartına bakılacaktır. Sonuç olarak B düğümü 5 olarak güncellenecek ve B'ye ait "pi" değeri A olacaktır. Aynı işlem diğer düğümler içinde uygulanır.

Çizelge 2.26. Şekil 3.11.' daki örneğe ait birinci iterasyon tablosu

	a	**b**	**c**	**d**
d	0	5	8	13
pi		A	C	B

İkinci iterasyonda öncelikle muhtemel gidilecek yollara bakılarak düğümler arasındaki mesafe belirlenir ve elde edilen sayı bir önceki ağırlık değeriyle

toplanarak şartın sağlanıp sağlanmadığı kontrol edilir. Eğer ki şart sağlanıyorsa düğümün ağırlık değeri güncellenir, sağlamıyorsa olduğu gibi bırakılır.

Çizelge 3.27. Şekil 3.11' daki örneğe ait ikinci iterasyon tablosu

	a	**b**	**c**	**d**
d	0	5	6	8
pi		A	A	C

Aynı işlemler üçüncü iterasyonda da uygulanır ve sonuçta elde edilen tablo aşağıdaki gibi olur. Tablo okunurken "pi" değerleri üzerinden gidilir. Bu örneğimizde "d" düğümüne gitmek için öncelikle "c" noktasından geçilmesi gerekir ve en kısa yol "A → C → D" şeklindedir.

Çizelge 3.28. Şekil 3.11.' daki örneğe ait üçüncü iterasyon tablosu

	a	**b**	**c**	**d**
d	0	5	6	8
pi		A	A	C

4. PROBLEM VE ÇÖZÜM YÖNTEMİ

Çalışmanın bu kısmında yukarıda bahsedilen teknik ve algoritmaları uygulama üzerinde göstermek amacıyla bir graf oluşturulmuş ve graf üzerinde iki nokta arasındaki en kısa yol bulunmaya çalışılmıştır. Burada amaç, en kısa yol algoritmalarını programlanabilir ve maliyet açısından karşılaştırmaktır. Çalışmada Bellman Ford algoritmasına yer verilmemiştir. Bellman Ford algoritması Dijkstra algoritması ile benzer çalışma mantığına sahiptir ve negatif ağılrlıklı graflar için tercih edilmektedir. Pozitif ağırlıklı graflar üzerinde Dijkstra algoritması daha hızlı sonuç verdiği için bu çalışmada Dijkstra algoritması kullanılmıştır.

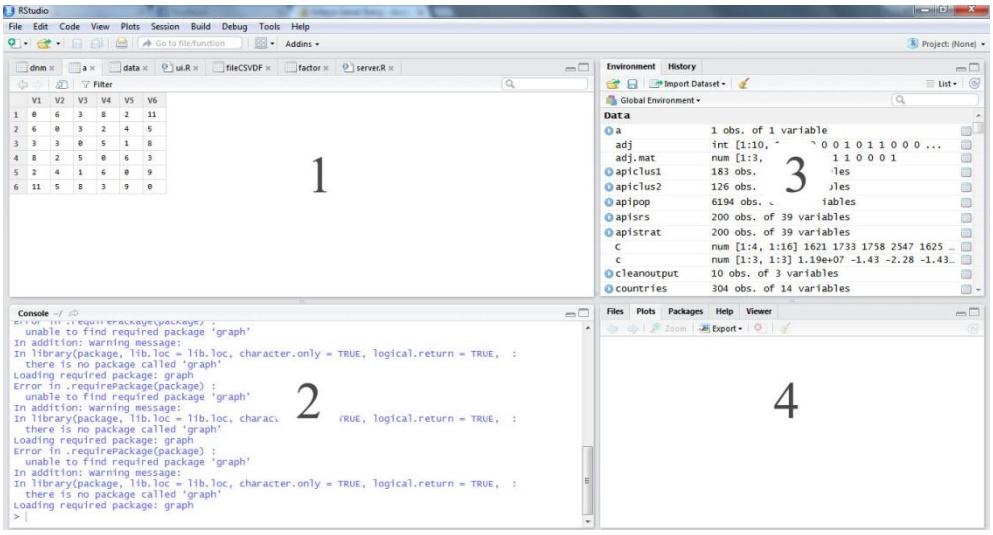
4.1. R Programlama

R programlama dili istatistiksel hesaplamalar için kullanılan ve açık kaynak kodlu bir editördür. Temel olarak S diline dayanmaktadır ve 1976 yılında Yeni Zellanda Auckland Üniversitesinden Ross Ihaka ve Robert Gentleman tarafından geliştirilmiştir.

R dilinin en belirgin özelliği kullanıcı veri görselleştirmek istediği zaman esnek yapılarda kodların ve paketlerin bulunmasıdır. Sytax yapısı olarak C diline benzemektedir. UNIX ile aynı anda geliştirilmeye başlanan R dilinin genel komut yapısı şu şekildedir;

Çıktı ← komut (girdiler ve argümanlar)

Veritabanı bağlantısı desteği sağlıyor olması kullanıcılar tarafından tercih edilmesinin en önemli nedenlerinden biridir. Şu ana kadar, arayüzünde bazı geliştirmeler yapılmış ve ortaya R Studio çıkmıştır.



Şekil 4.1. RStudio'nun kullanıcı ara yüzü

1. Veri setlerinin gösterildiği alan
2. Kod yazım bölümü
3. Değişkenlerin tutulduğu bölüm
4. Paket, grafik ve kütüphanelerin gösterildiği bölüm

Genel anlamda R programlama dili GNU'nun (GNU's Not Unix) bir parçası ve S programlama dilinin açık kaynak kodlu hali olmasından dolayı bazen GNU S olarak anılmaktadır.

Grafik teoremi için R programlama dilinde "igraph" paketi kullanılmaktadır. Bu paket bir grafa ait en kısa yolu bulmak için kullanıldığı gibi çeşitli grafik çıktılarıyla ve algoritma yapılarıyla kullanıcıya en iyi sonucu sunmaktadır.

4.2. Uygulamanın Yazılım Aşamaları

4.2.1. Veritabanı

Bir programda en iyi sonuçları elde edebilmek için yazılım dili ile veritabanının olabildiğince entegre çalışması gerekmektedir. Eğer bir PHP sayfası oluşturulmuşsa ilgili yazılım dili için en iyi sonucu verecek olan veritabanı MYSQL'dir. Yapılan

uygulama C# olmasından dolayı seçilen veritabanı MS-Sql olmuştur. Veritabanı versiyonu olarak MS-Sql Edition seçilmiştir. Bu versiyon çekirdek yapıda olup ücretsiz sürümdür.

Veritabanı için düğüm tablosu, mesafeler tablosu ve izlenen yol olmak üzere üç adet tablo kullanılmıştır. Tablolar, aralarında *"id"* sütunu ile ilişkilendirilmiştir. Bunun sebebi tablolar arası veri aktarımı yapılırken benzersiz bir kimlik ile verilere ulaşmaktır.

4.2.1.1. Düğüm tablosu

Bu tablo içerisinde düğümlere ait bilgiler ve Google Map koordinatları bulunmaktadır. Tablo aşağıdaki bilgilerden oluşmaktadır.

Column Name	Data Type	Allow Nulls
id	int	☐
isim	nvarchar(50)	☑
dno	nvarchar(50)	☑
hdnLat	nvarchar(50)	☑
hdnLng	nvarchar(50)	☑

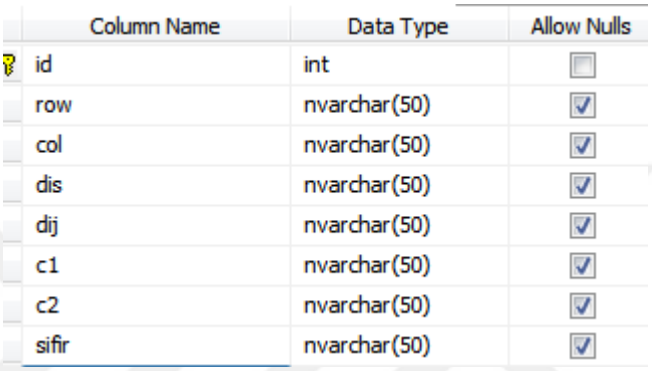
Şekil 4.2. Düğümlere ait MS-SQL tablosu

"id" alanı her düğüme ait bir numara tutmaktadır. Bu numaralar sistem tarafından otomatik olarak atanmaktadır. Tabloların birbiriyle ilişkilendirilmesi için bu alandan faydanılacağı için birincil anahtar "id" kısmına eklenir ve veri tipi integer (sayı) şeklindedir.

"dno" kısmında düğüm isimlerini içermektedir. "hdnLat" seçilen noktaya ait enlem, "hdnLng" ise boylam bilgilerini tutmaktadır. "id" alanı haricindeki her tablo sütunu için veri tipi "nvarchar(50)" seçilmiştir.

4.2.1.2. Mesafeler tablosu

Bu tablo içerisinde düğümler arası mesafeler, düğüm numaraları, Floyd algoritmasına ait başlangıç matrisi ve Dijkstra algoritmasına ait matris bulunmaktadır. Bu bilgiler gerekli hesaplamalar yapılarak veri tabanında saklanmaktadır ve iki nokta arasındaki en kısa yolun bir defa bulunup algoritmalar tekrar çalıştırılarak programın yavaşlaması engellenmektedir.



Column Name	Data Type	Allow Nulls
id	int	☐
row	nvarchar(50)	☑
col	nvarchar(50)	☑
dis	nvarchar(50)	☑
dij	nvarchar(50)	☑
c1	nvarchar(50)	☑
c2	nvarchar(50)	☑
sifir	nvarchar(50)	☑

Şekil 4.3. Grafik çözümüne ait MS-SQL tablosu

"id" numarası yine birincil anahtar olarak sistem tarafından atanmaktadır. "row" ve "col" alanları düğüm numaralarını tutmakta ve tablolar arası ilişkilendirmeyi kolaylaştırmaktadır. "dis" kısmı düğümler arasındaki mesafeyi barındırmaktadır ve veri tipi olarak string ayarlanmıştır. Veri o alandan çağırılırken gerekli dönüşüm yapılarak işlemlere devam edilmektedir. "dij" Dijkstra algoritmasına ait hesaplamaları, "sifir" ise Floyd algoritmasına ait başlangıç matrisini tutmaktadır.

4.2.1.3. İzlenen Yol

Asp.net'de sayfalar arası veri gönderilirken iki türlü yöntem izlenir. Bunlardan birisi "post yöntemi" iken diğeri "get yöntemi"dir. Fakat bu yöntemler geçici bellekte saklanmaktadır ve ileriki aşamalarda veri gerekli olduğunda programı tekrar çalıştırmak gereklidir. Bu işlem zaman alacağından izlenecek yolun veri tabanında saklanılması uygun bulunmuştur.

Column Name	Data Type	Allow Nulls
id	int	☐
dgm	nvarchar(50)	☑
lat	nvarchar(50)	☑
lng	nvarchar(50)	☑

Şekil 4.4. İzlenecek yola ait MS-SQL tablosu

Alan adları yukarıda görüldüğü gibi "dgm" düğüm numarasını, "lat" enlem değerini, "lng" ise boylam değerini tutmaktadır.

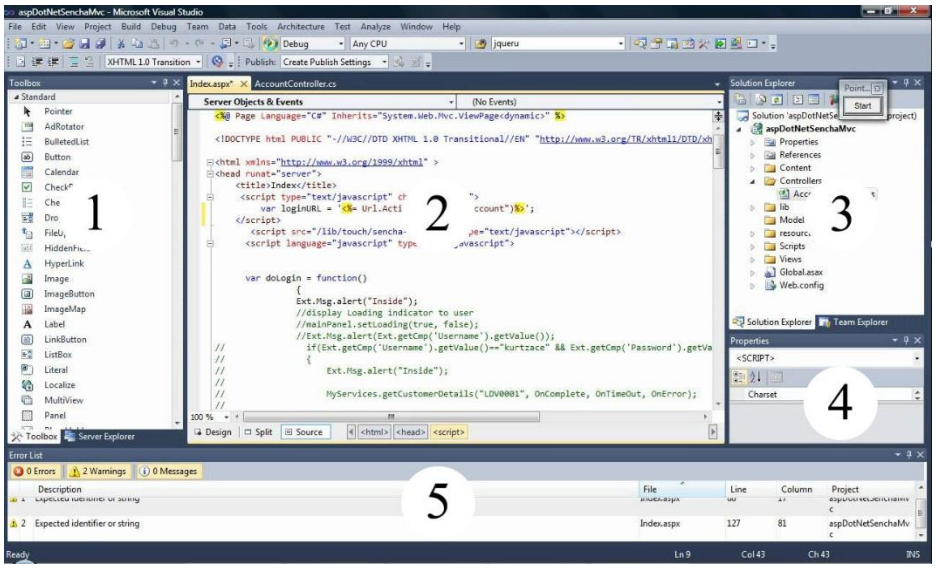
4.2.2. Yazılım aşaması

4.2.2.1. NET Platformu

Microsoft tarafından geliştirilen .Net Platformu hem masaüstü hem web yazılımları geliştirilmesi açısından tercih edilen bir editördür. Öncesinde dinamik yapıdaki bazı programlama dilleri (C#, C++…vb.) ayrı ayrı editörler ile yapılırken şimdi bu platform sayesinde bir araya toplanmış ve tek editör üzerinden işlem yapılmasına olanak sunmaktadır.

4.2.2.2. Visual Studio 2010

Visual studio yukarıda bahsedilen dilleri bir araya toplayan ve.NET platformunu da içerisine dahil etmektedir. Kod editörünün yanı sıra hata ayıklayıcı ve tasarım kısmından oluşmaktadır.



Şekil 4.5. Microsoft Visual Studio 2010 Kullanıcı Ara yüzü

1.Toolbox : tasarım sayfası açıldığı zaman sol tarafta kalan kısım visual studio içerisinde bulunan nesneleri (dropdrownbox,button,text,…vb.) barındırmaktadır. Bu nesneler sayesinde sayfa içerisinde olayların kontrolleri sağlanabilmekte ve her nesneye ayrı bir görev yada algoritma atanabilmektedir.

2.Design : Sayfa oluşturulduğu zaman karşımıza gelen pencere tasarım (design) kısmıdır. Bu kısımda eklenen resim, çerçeve,…vb. nesneler görüntülenmektedir.

3.Source : Tasarım kısmında eklenen nesnelere ait komut ve görevler bu kısımda verilebilir.

4.Solution Explorer : Sağ üst tarafta bulunan explorer kısmında oluşturulan sayfalar, css ve scriptler barındırılmaktadır. Bu kısımda sayfa ya da kodlar eklenebilir ve çıkarılabilir.

5.Properties : Nesnelere ait bütün özellikler bu kısımda saklanmaktadır. Örneğin bir butona ait yükseklik, ait olacağı css sınıf ya da arka plan rengi bu kısımdan değiştirilebilir.

4.2.2.3. Google Map ve API'leri

Google geçmişte ve şu anda en popüler arama motoru olmasının yanı sıra sunmuş olduğu Google Map, Google Earth, Earth Builder gibi servislerle uygulama yazılımcılarına birçok olanak sağlamıştır. Bu hizmetler https://developers.google.com/ isimli internet adresinde bir araya toplamıştır.

API içerisinde JavaScript'ten faydalanmış ve böylelikle nesnel programlamaya yatkın geliştiricilere kolaylık sağlamıştır. Uygulama içerisinde kullanılan HTML5 ve CSS dilleri ile web sitelerine görsellik kazandırılmıştır.

4.2.2.4. Bootstrap

Bootstrap teknolojisi geliştirilen web uygulamalarının mobil cihazlar üzerinde görsel anlamda daha etkili bir kullanıcı arayüzü oluşturmak için tercih edilen küçük scriptlerdir. Bu teknoloji arka plandaki işleri yapmak için Javascript tasarım kısmında ise CSS dilini kullanmaktadır. Web üzerinde kullanıma sunulan bu kütüphane açık kaynak kodludur ve twitter tarafından geliştirilmiştir.

Bootstrap kurulum için ayrı bir exe dosya içermemektedir. Bunun yerine css ve Javascript dosyalarını internet ortamından alınmasına olanak sağlamaktadır. Böylelikle html bir sayfanın içerisine basit kodlarla kurulumu yapılabilmektedir. Şu anda v3.3.6 versiyonu kullanılmakta olup sayfa içerisine kurulumu aşağıdaki kodlarla sağlanır.

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"><li
nk rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-
theme.min.css"><script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js">
</script>
```

Bu kodlarla beraber proje içerisine “css”, “js” ve “fonts” olmak üzere üç adet klasör dahil olmaktadır. Bu kodları kurmak için, html içerisinde “head” tagının kapatıldığı satırdan önce kurulum kodlarının kopyalanması gerekmektedir. Masaüstü işletim sistemlerine ait bütün tarayıcılarda desteklenen bootstrap, mobil işletim sistemleri içerisinde kullanılan tarayıcılar içerisinde bir tek opera tarafından desteklenmemektedir. Bunun sebebi daha hızlı kullanım sunmaya çalışan opera Javascript çalıştırma motorunu tarayıcı içerisinden kaldırmasıdır.

Gerekli kurulum kodları sayfaya eklendikten sonra mobil uyumluluğu aktif hale getirmek için meta taglar kullanılmaktadır. Meta taglar “head” etiketleri arasında bulunması gerekmektedir. İlgili komutlar aşağıdaki gibidir.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

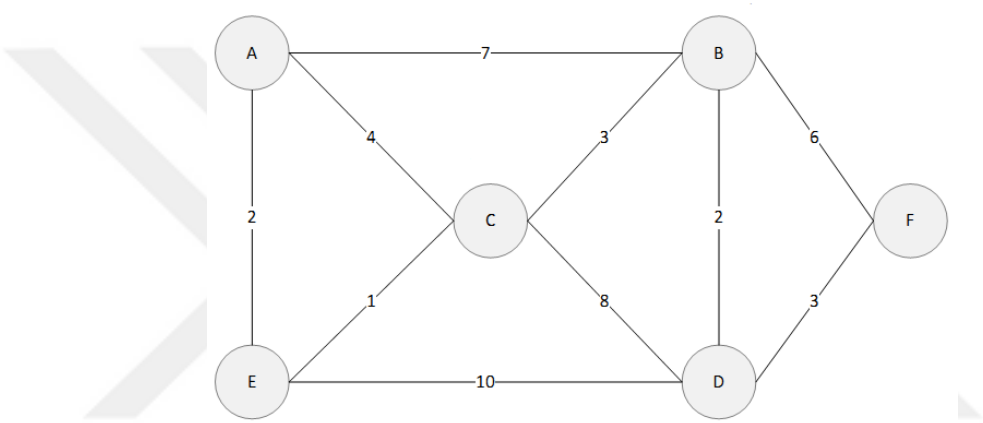
```
<metaname="viewport"content="width=device-width,initial-scale=1,maximum-
scale=1, user-scalable=no">
```

Bootstrap, html gövde yapısı içerisinde ızgara sistemi kullanmaktadır. Bu sistem oniki parçadan oluşmaktadır. HTML elementlerin içerisinde kullanmak için her nesneye sınıf tanımlaması yapılması gerekir. Sınıf tanımlamaları mobil, tablet ve masaüstü cihazların ekran çözünürlüğüne göre değişim göstermektedir. Örneğin div yapısını kullanacaksak *“class”* argümanına *“col-xs-12 col-sm-6 col-md-8”* tanımlaması yapılması gerekir.

```
<div class="row">
  <div class="col-xs-12 col-sm-6 col-md-8">.col-xs-12 .col-  sm-6 .col-md-8</div>
  <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
</div>
```

5. UYGULAMA

Aşağıdaki grafta iki nokta arasındaki en kısa yol bulunmak istenmektedir. Grafda düğümler şehirleri ve ayrıtlar şehirlerarasındaki yolu temsil etmektedir. Grafın ağırlıkları ise şehirlerarası mesafeyi göstermektedir.



Şekil 5.1. Uygulamaya ilişkin örnek

Öncelikle grafın veri tabanına kaydının kolaylaşması açısından ve Floyd algoritması için gerekli olan D matrisini oluşturulmalıdır.

Çizelge 5.1. Şekil 5.1' e ait grafın komşuluk matrisi

	A	**B**	**C**	**D**	**E**	**F**
A	0	7	4	∞	2	∞
B	7	0	3	2	∞	6
C	4	3	0	8	1	∞
D	∞	2	8	0	10	3
E	2	∞	1	10	0	∞
F	∞	6	∞	3	∞	0

5.1. Uygulamanın R Dilinde Çözümü

Öncelikle graf teoremi için optimum çözümü verecek olan *"igraph"* paketi ileriki aşamalarda kullanılmak üzere indirilmiştir. Bu işlem aşağıdaki komut kullanılarak gerçekleştirilmiştir.

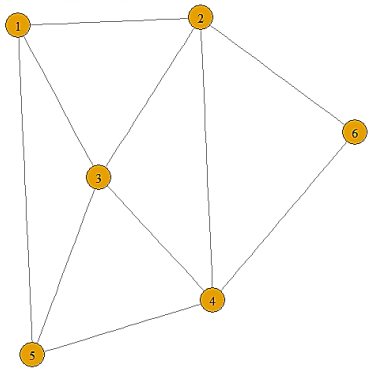
> library(igraph)

Library komutu kullanılmak istenen paketi program içerisine dahil eder ve paketi hazır hale getirir. Oluşturulacak grafik için "graf komutu kullanılır ve dizin oluşturmak için *c(...)* komutu uygulanır.

>gr←graph(edges=c(1,2,1,3,1,5,2,3,2,4,2,6,3,4,3,5,4,5,4,6),n=6,directed=F)

"graph" komutu içerisinde bulunan "n" argümanı düğüm sayısını, "directed" ise grafiğin yönlü ya da yönsüz olduğunu belirlemede kullanılır. Tanımlama yapılırken eğer düğümler için farklı isim kullanılmak istenirse c dizini içerisinde c("A","B","A","C") şeklinde tanımlanabilir. Gerekli tanımlamalar yapıldıktan sonra oluşan grafiği görmek için "plot" komutu uygulanır.

> plot(gr)



Şekil 5.2. Örneğin R Programında gösterimi

Çizilen grafiğin program içerisinde doğruluğunu kontrol etmek için *"E"* komutu kullanılır. *E (Edge - Ayrıt)* komutu kullanıcıya ayrıt sayısını ve düğümler arasındaki bağlantıları göstermektedir. Aynı zamanda grafiğe ait düğümleri görmek için *"V"* komutu kullanılır.

```
> E(gr)
  + 10/10 edges:
    [1] 1--2 1--3 1--5 2--3 2--4 2--6 3-- 4 3--5 4--5 4--6
> V(gr)
  + 6/6 vertices:
      [1] 1 2 3 4 5 6
```

Bu işlemden sonra grafiğe ait ağırlıklar programa tanıtılmalıdır. Bunun için öncelikle ağırlıklar yukarıda verilen ayrıt sırasına göre bir dizin olarak değişkene aktarılmalıdır.

```
>w=c(7,4,2,3,2,6,8,1,10,3)
> w
 [1]  7  4  2  3  2  6  8  1 10  3
```

Oluşturulan dizin grafiğin içerisine ağırlık olarak tanıtılmalıdır. Bunun için *"weight"* komutu kullanılır. Daha sonra verinin doğruluğunu kontrol etmek için yine *"E"* komutu kullanılabilir.

```
> E(gr)$weight = w
> E(gr)$weight
 [1]  7  4  2  3  2  6  8  1 10  3
```

Oluşturulan ağırlık artık matris içerisinde temsil edilmektedir ve grafik üzerinde de gözlemlenebilir. Matrisi görebilmek için değişken adının yanına köşeli parantez açılıp kapatılır.

```
>gr[]
 .  7  4  .  2  .
 7  .  3  2  .  6
 4  3  .  8  1  .
 .  2  8  .  10 3
 2  .  1  10 .  .
 .  6  .  3  .  .
```

En kısa yolu bulmak için *"shortest.paths"* komutu kullanılır. Böylelikle optimum çözüme ulaşılır.

```
> shortest.paths(gr)
```

0	6	3	8	2	11
6	0	3	2	4	5
3	3	0	5	1	8
8	2	5	0	6	3
2	4	1	6	0	9
11	5	8	3	9	0

"shortest.paths" komutu tek başına kullanılabileceği gibi birden fazla argüman ile birliktede kullanılabilir. Genel kullanım şekli aşağıdaki gibidir.

shortest.paths(graph, v=V(graph), to=V(graph),
mode = c("all", "out", "in"), weights = NULL ,
algorithm=c("automatic","unweighted","dijkstra","bellman-ford","johnson"))
Argümanlar;

Graph: Üzerinde çalışılan grafı temsil etmektedir.

V: Başlangıç düğümünü belirlemek için kullanılır.

To: Bitiş düğümünü belirlemek için kullanılır.

Mode: Eğer bu argüman "out" ise başlangıç düğümünden itibaren, "in" ise bitiş düğümünden sonra en kısa yolu bulması gerektiğini simgelemekte olup "all" ise varsayılan olarak belirlenmiştir. Fakat bu argüman yönsüz graflar için kullanılmamaktadır.

Weight: Graf içerisinde ağırlık barındırıyorsa "NULL" barındırmıyorsa "NA" olarak ayarlanmalıdır.

Algorithm: Bu argüman ile grafikte kullanılacak en kısa yolu belirleyen algoritma seçilmektedir. Varsayılan olarak uygulandığında bu argüman en hızlı ve uygun algoritmayı kendisi seçmektedir. Eğer graf içerisinde negatif ağırlık varsa ve 100 düğümden fazlaysa Jhonson algoritması kullanılmaktadır. Eğer bütün düğümler pozitif ise Dijkstra algoritması seçilmelidir.

Grafiği daha görsel hale getirmek için igraph paketi içerisinde birkaç komut mevcuttur. Bunlar “ecol” ve “vcol” komutlarıdır. Bu komutlarla beraber en kısa yol olarak belirlenen yol renklendirilmektedir. Yeni oluşan grafiği görmek için yine “plot” komutu kullanılır. Kullanılan argümanlarda oluşturulan renklerin atamaları mevcuttur.

```
> ecol <- rep("gray80", ecount(gr))
> ecol[unlist(news.path$epath)] <-     "orange"
> ew <- rep(2, ecount(gr))
> ew[unlist(news.path$epath)] <- 4
> vcol <- rep("gray40", vcount(gr))
> vcol[unlist(news.path$vpath)] <- "pink"
> plot(gr, vertex.color=vcol, edge.color=ecol,
+      edge.width=ew, edge.arrow.mode=0)
```

Düğümler arasındaki en kısa yolları görebilmek için *“all_shortest_paths”* komutu kullanılmaktadır. Buradaki amaç seçilen iki düğüm arasındaki minimum çözümü görebilmektir.

```
> all_shortest_paths(gr, from=V(gr), to = V(gr), mode = c("out", "all", "in"))
$res
$res[[1]]
+ 1/6 vertex:
[1] 1
$res[[2]]
+ 4/6 vertices:
[1] 1 5 3 2
$res[[3]]
+ 3/6 vertices:
```

[1] 1 5 3

$res[[4]]

+ 5/6 vertices:

[1] 1 5 3 2 4

$res[[5]]

+ 2/6 vertices:

[1] 1 5

$res[[6]]

+ 6/6 vertices:

[1] 1 5 3 2 4 6

$nrgeo

[1] 1 1 1 1 1 1

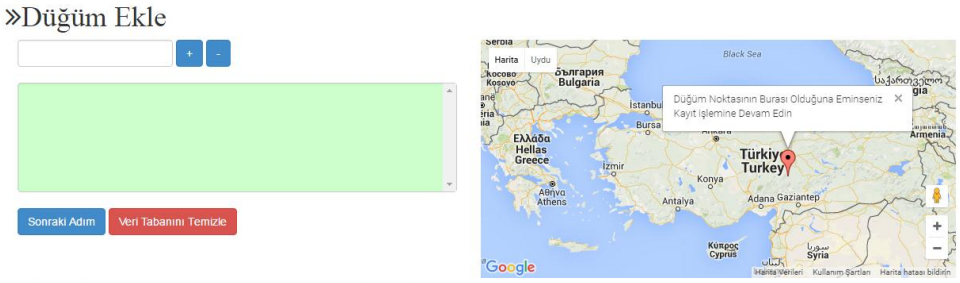
5.2. Uygulamanın Web Ortamında Çözümü

Graf yönsüz olduğu için uzaklık matrisi simetriktir. Verilerin kolaylıkla program içerisine dahil edebilmek için uygulama içerisinde farklı bir sayfa (excelm.aspx) oluşturulmuştur.

	A	B	C	D	E	F
1	0	7	4	9999	2	9999
2	7	0	3	2	9999	6
3	4	3	0	8	1	9999
4	9999	2	8	0	10	3
5	2	9999	1	10	0	9999
6	9999	6	9999	3	9999	0

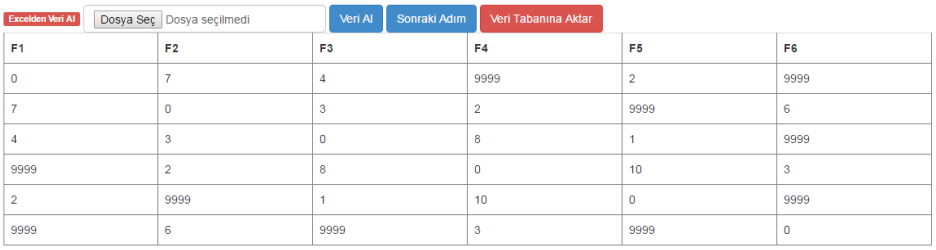
Şekil 5.3. Uygulamaya ait excel formatının gösterimi

Bu aşamada program çalıştırılır ve ilk adımda düğümler tanımlanır. Düğümler tanımlanırken sağ tarafta bulunan haritanın imleci hareket ettirilerek ilgili koordinatın üzerine gelinir. Düğüme verilecek isim sol üst tarafta bulunan "Düğüm Ekle" ibaresinin hemen alt tarafına girilebilir.



Şekil 5.4. Uygulamanın birinci adımı (step1.aspx)

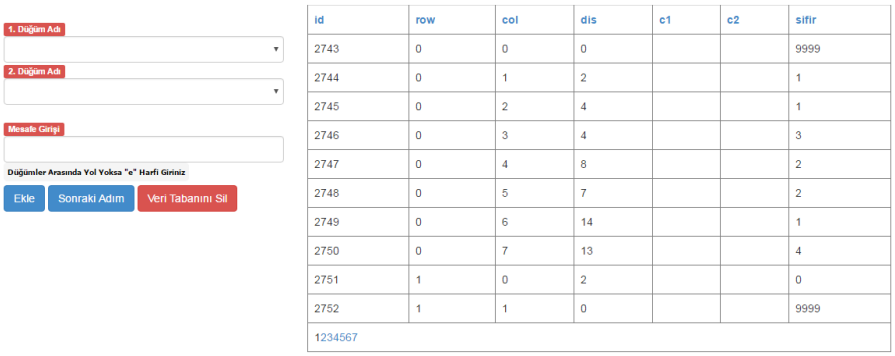
Şekil 5.4.' te görülen yeşil kutunun içerisinde düğüm isimlerinin listesi vardır ve buradan grafik içerisinde kaç adet düğüm olduğu görülebilmektedir. Düğüm isimleri girildikten sonra "Sonraki Adım" butonuna tıklanır ve diğer sayfaya geçilir.



F1	F2	F3	F4	F5	F6
0	7	4	9999	2	9999
7	0	3	2	9999	6
4	3	0	8	1	9999
9999	2	8	0	10	3
2	9999	1	10	0	9999
9999	6	9999	3	9999	0

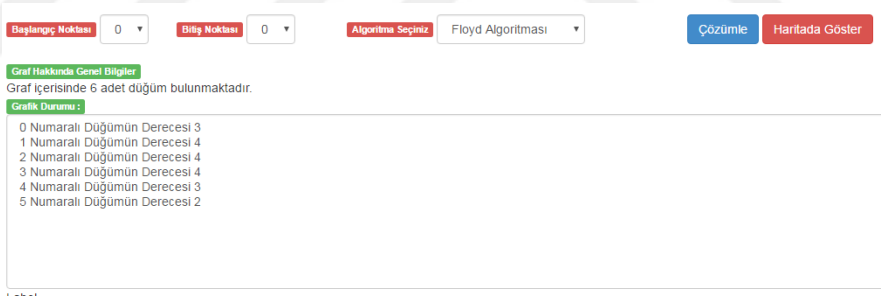
Şekil 5.5. Excel dosyası yükleme sayfası (excelm.aspx)

Excelm.aspx sayfası excelden veri aktarımına imkan sağlamaktadır. "Dosya Seç" butonuyla verilerin bulunduğu excel dosyası seçilir, "Veri Al" butonuyla seçilen veri görülebilir. Eğer verilerde bir problem yoksa "Veri Tabanına Aktar" butonuna tıklanarak veriyi MS SQL veri tabanı içerisine atılabilir. İşlemlere devam etmek için "Sonraki Adım" butonuna tıklanır.



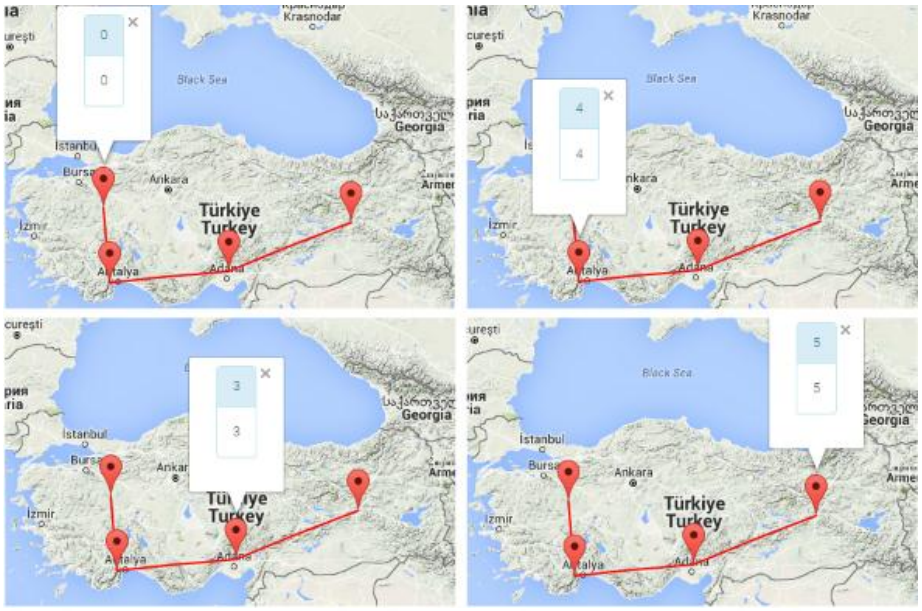
Şekil 5.6. Düğümler arası mesafelerin düzenlendiği sayfa (step2.aspx)

Şekil 5.6.' te veriler üzerinde değişiklik yapmak istenirse düğüm isimleri seçilir ve mesafe girişi yapılır. "Ekle" butonuna tıklandığında ilgili satır ve sütuna ait veri güncellenir. Eğer veri girişinde düzeltilemeyecek bir hata söz konusu ise "Veri Tabanını Sil" butonuna tıklayarak bütün veri tabanı silinebilir. Bu sayfadaki işlemler bitirildikten sonra algoritma çözümlerinin bulunduğu sayfaya geçmek için "sonraki adım" tuşuna basılır.

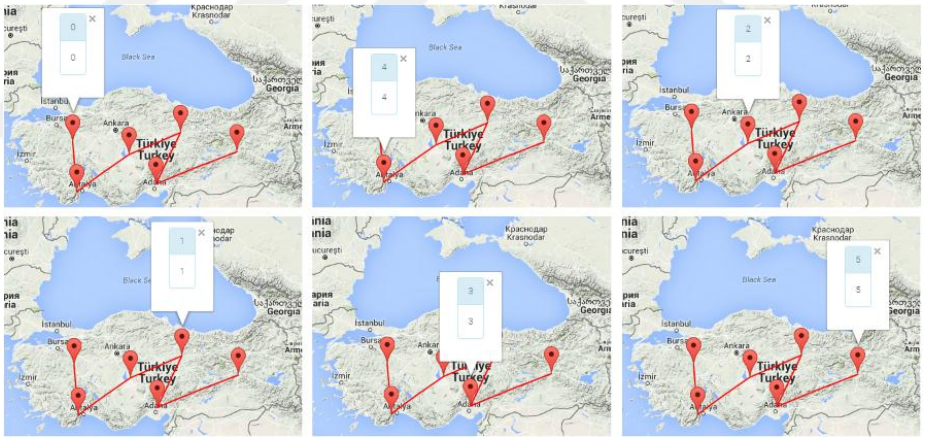


Şekil 5.7. Grafa ait bilgilerin ve algoritmaların bulunduğu sayfa (step3.aspx)

Şekil 5.6.' da başlangıç ve bitiş düğümleri seçilir. En sağ tarafta bulunan açılır kutudan algoritma tercihi yapılır ve grafiğin o algoritmaya göre çözümlenmesi istenir. Çözümleme yapıldıktan sonra "Haritada Göster" butonuna tıklanarak harita sayfasına geçilir.



Şekil 5.8. Örneğin Floyd Algoritması Çözümü



Şekil 5.9. Dijkstra Algoritması ile Çözümü

Şekil 5.8. ve Şekil 5.9.‘ da seçilen algoritmalara ait çözümler harita üzerinde gösterilmekte ve ilgili düğümler koordinatlarıyla beraber işaretlenmektedir. Aralarındaki kırmızı çizgi en kısa yolu göstermektedir.

6. SONUÇ

Navigasyon sistemleri özellikle 21. Yüzyıl sonlarında insanların dikkatini çekmiş, minimum maliyet ve en kısa yol problemleri için en çok tercih edilen uygulama alanlarından biri haline gelmiştir.

En kısa yol problemi, düğüm olarak simgelenen şehir ya da geçiş noktaları arasında mesafeleri gösteren bir grafta başlangıç düğümünden bitiş düğümüne toplam mesafe uzunluğu en kısa yolu bulma problemidir. Problemin çözümünde Floyd, Dijkstra ve Bellman – Ford algoritmaları tercih edilen algoritmaların birkaç tanesidir. Geliştirilen algoritmalar en kısa yol belirlenmesinde kendisine has yöntemler kullanarak çözüme ulaşmaya çalışmıştır.

Bu çalışma içerisinde en kısa yol probleminin çözümü için Floyd, Dijkstra ve Bellman – Ford algoritmaları kullanılmış ve algoritmalar arasındaki farklar incelenerek avantaj ve dezavantajları üzerinde durulmuştur. Aynı zamanda söz konusu algoritmalar bir uygulama içerisine test edilmiş ve doğruluğu R programlama ile karşılaştırılmıştır.

Yapılan karşılaştırmalar sonucu elde edilen bulgular şu şekildedir:

Floyd algoritması diğer algoritmalara göre programlanabilirliği daha kolay olduğu görülmüştür. Fakat daha büyük bir graf söz konusu olduğu durumlarda algoritmanın yapısı gereği oluşturulan iç içe döngüler programın yavaşlamasına neden olacağı öngörülmüştür.

Dijkstra algoritması ise büyük graflar için kullanılabilecek en hızlı ve en uygun algoritma olması sonucuna ulaşılmış fakat programlanabilirlik açısından diğer algoritmalara göre daha zor olduğu anlaşılmıştır. Dijkstra algoritmasının bir diğer dezavantajı ise negatif ağırlıklı grafları hesaplayamamasıdır.

Bellman – Ford algoritması yapı olarak Dijkstra algoritmasına benzer olduğu anlaşılmıştır. Bu algoritmanın avantajı ise negatif ağırlıklı graflar için çalışabiliyor olmasıdır. Negatif ağırlıklı graflar haritalama içermeyen maliyet hesaplamaları için kullanılmasından dolayı uygulama içerisine dahil edilmemiştir.

Uygulamada çalıştırılan Floyd ve Dijsktra algoritmaları, düğümler arası en kısa yolun belirlenmesinde iki farklı yol seçmiştir. Algoritmaların seçmiş oldukları yolun maliyeti 11 birim olup her iki algoritmada da aynıdır. Dijkstra algoritmasının Floyd algoritmasına göre daha fazla düğüm kullandığı görülmüştür. Bunun sebebi Floyd algoritmasının bütün düğümleri hesaplaması ve dolayısıyla en kısa yolu en az düğüm sayısıyla göstermesidir. Fakat Dijkstra algoritması Greedy yaklaşımı kullanmasından dolayı sonraki düğümler arası en kısa yolu bilemeyeceği için anlık en kısa yolu belirlediği gözlemlenmiştir.

Her ne kadar küçük graflarda en kısa yolu hesaplama zamanı aynı gibi görünsede aralarında mili saniyelik farklar oluşmaktadır. Daha büyük graflar söz konusu olduğu zaman Floyd algoritmasını yapısı gereği içiçe oluşan döngüler en kısa yol çözümünde daha fazla zaman alacaktır. Fakat Dijkstra algoritmasının kullandığı yukarıda bahsedilen yöntem gereği büyük graflarda hesaplamalar daha kısa sürede olacağı öngörülebilir.

Algoritmalar arasındaki en kısa yol hesaplamak için kullanılan zaman farkı veri tabanı kullanılarak en aza indirgenmeye çalışılmıştır.

KAYNAKLAR

Atallah M. (1984), Finding Euler Tours in Parallel, Journal of Computer and System Science, 29: 330 – 337

Bahar R. I., Frohm E. A., Gaona C. M., Hachtel G. D., Macii E., Pardo A., ve Somenzi F. (1993) Algebraic decision diagrams and their applications, *Formal Methods in System Design*, 10: 188–191.

Bayzan Ş. ve İnal M. (2011), Navigasyon Sistemi İçin Araç Güzergahı Belirlenmesinde Gerçek Zamanlı Trafik Bilgisi Kullanımı, *International Computer &* Instructional Technologies Symposium, 2011, Elazığ, 7.

Caldwell C. K. (1995), *Graph theory glossary*, University of Tennessee Department of mathematics and statistics.

Cormen T. H., Leiserson C. E., Rivest R. L. ve Clifford S. (2001), *Introduction To Algorithms*, Mıt Press And Mcgraw-Hill, 984.

Çölkesen R. (2000), *Veri Yapıları Ve Algoritmalar*, Papatya Yayıncılık, İstanbul, 346.

Demirkol Ö.E. ve Demirkol A. (2003) Dıjkstra Ve Bellman-Ford En Kısa Yol Algoritmalarının Karşılaştırılması, *Sau Fen Bilimleri Enstitüsü Dergısi* 7 .Cılt, 3.Sayı.

Deng Y. ve Chen Y. (2012), Fuzzy Dijkstra Algorithm For Shortest Path Problem Under Uncertain Environment, *Applied Soft Computing*, 12: 1231 – 1237.

Floyd W.R. (1962), Algorithm 97: Shortest Path, *CACM*, 5: 345.

Gajbhiye S. (2013), Optimal Power Flow Path Selection Using Different Shortest Path Algorithms, *IJIET*, 2:213 – 217.

Hanumanthappa J. ve Manjaiah D.H. (2010), *A Study On Contrast And Comparison Between Bellman-Ford Algorithm And Dijkstra's Algorithms*, National Conference on Wireless Networks, 2014, India, 9.

Hart C. (2013), *Graph Theory Topics in Computer Networking*, University of Houston-Downtown, Department Computer and Mathematical Sciences, Senior Project.

Indrajaya A, Affandi A, Pratomo I. (2015), Desıgn Of Geographıc Informatıon System For Trackıng And Routıng Usıng Dıjkstra Algorıthm For Publıc Transportatıon, ICWT, 1 – 4.

Jeyhun C., Ghang L. ve Seungjae L. (2014), An Automated Direction Setting Algorithm For A Smart Exit Sign, *Automation in Construction*, 139 – 148.

Joyner D., Nguyen M.V. ve Philips D. 2013, *Algorithmic Graph Theory And Sage,* Free Software Foundation, 327.

Kai N., Yao – Ting Z., Yue – Peng M. (2014) Shortest Path Analysis Based On Dijkstra's Algorithm İn Emergency Response System, *TELKOMNICA*, 12: 3476 – 3482.

Leiserson E., 2009, *Introduction to Algorithms*, 3. Baskı, The MIT Press, Boston, 1313.

Lui C., Zhou A., Du Q. ve Zhang G. (2013) Practical Path-Based Methods For Clustering Arbitrary Shaped Data Sets, *ICNC*, 962 – 966.

Lui H., Stoll N., Junginger S. ve Thurow K. (2012) A Floyd-Dijkstra Hybrid Application For Mobile Robot Path Planning İn Life Science Automation, *INTECH*, 10: 1 – 14.

Saran M.S. (2008), *Graf Teorisinin Bazı Mühendislik Uygulamaları,* Yüksek Lisans, Balıkesir Üniversitesi, Balıkesir, 60

Sathyaraj M., Jain L.C., Finn A. ve Drake S. (2008), Multiple Uavs Path Planning Algorithms: A Comparative Study, *Fuzzy Optimization And Decision Making*, 7: 257-267.

Shaikh A. ve Dhale A. (2013), Agv Routing Using Dijkstra's Algorithm – A Review, *IJSER,* 4: 1665 – 1670.

Tarım V. (2007), *Graf Teorisine Dayalı Web Arayüzlü Yol Problemi Uygulaması, Yüksek Lisans Tezi*, Beykent Üniversitesi, İstanbul, 6.

Tong L.,Nie L,Guo G, Li J.M. (2015), Study On Search Algorithm Of Passenger Travel Route İn Railway Trans-Port Network, *The Open Cybernetics & Systemics Journal*, 9: 2436 – 2444.

Tseng Y. C., Yang M. H. ve Juang T. Y. (1998), An Euler-Path-Based Multicasting Model For Wormhole-Routed Networks With Multi-Destination Capability, *International Conference On. Ieee*, 1998, 366-373.

Ulukan A., Korul H. (2015), Acil Telefon Merkezleri Modellemesinin Anadolu Üniversitesi Yerleşkesine Uygulanması, *EJOIR*, 3: 71 – 76.

Yıldırım Ş. ve Yasar E. (2015), Development Of An Obstacle-Avoidance Algorithm For Snake-Like Robots, *Measurement*, 68 – 73.

ÖZGEÇMİŞ

Kişisel Bilgiler

Ad Soyad : Sıtkı Cansu

Uyruk : T.C.

Doğum Yeri ve Tarihi : Beyşehir / 19.10.1985

Medeni Hali : Bekar

Telefon : 0 545 217 00 01

E-posta – Web Site : scansu@hotmail.com.tr – www.scansu.net

Eğitim

Alınan Derece	Aldığı Kurum/Üniversite	Mezuniyet Yılı
Lise	Beyşehir Anadolu Ticaret Meslek Lisesi	2005
Lisans	Muğla Sıtkı Koçman Üniversitesi	2013

İş Tecrübesi

Yıl	Yer	Pozisyon/Görev
2009	Entegre Yazılım	Web Tasarım
2010	Beyşehir Endüstri M.L.	Bilgisayar Öğrt.
2016	Tuğra Etüt Eğitim Merkezi	İngilizce Öğrt.

Yabancı Dil

İngilizce	Başlangıç	Orta	İleri
Yazma			X
Konuşma		X	
Anlama		X	
Okuma			X