YAZILIM LABORATUVARI I

PROJE 1

KARGO DAĞITIM SİSTEMİ

KAAN HIRÇIN, ALPER TALHA KÜÇÜK

160202032 – 180202034
Bilgisayar Mühendisliği Bölümü (İÖ)
Kocaeli Üniversitesi

1.Problem Tanımı

Projede akıllı kargo dağıtım sistemi yapan bir masaüstü uygulaması geliştirilmesi bekleniyor. Biz bunu C# ile gerçekleştirdik.

Yapılan Araştırmalar

C# projesine Google API entegre etmek, Google Maps API Google tarafından hazırlanmış olan haritaları ve harita veri tabanlarının farklı uygulamalarda da kullanılabilmesini mümkün hale getirmektedir. Yani Google Haritalar'ı sayfalarına entegre etmek, Google Haritalar'dan veri almak, basit kullanım veya kapsamlı kişiselleştirmelere olanak sağlayan bir Google ürünüdür. Harita tabanlı farklı amaçlı çözümlerde kullanılan Google Maps Developer Console'da bulunan hazır kodlardır. Tamamen açık kaynak olarak sunulan bu ara yüzler, kendi işletmeleri için ya da piyasaya sunmak için uygulama geliştirmek isteyenlere temin edilmektedir. Google bu API'lar için aylık 200 dolara kadar ücretsiz kullanım hakkı tanımaktadır. Yüksek hacimli kullanımlar için Google Partner aracılığı ile bu servisleri çok daha uygun şartlarda kullanıcılara sağlamaktadır.

Google Maps Api Çeşitleri Google tarafında kullanıcılara sunulan API servislerinin çok farklı kullanım alanları bulunmaktadır. Bunlardan bazılarını şu şekilde sıralamak mümkündür: •**Direction API**: Rota optimizasyonu ve anlık rotalama olanağı sağlamaktadır. Google Maps API arasında en sık tercih edilenlerden biri olan Direction API kullanarak rota hazırlarken birden fazla ulaşım aracını baz almak da mümkündür. Ayrıca canlı trafik bilgileri de rota hesaplamasına entegre edilebilmektedir. •**Distance Matrix API**: Başlangıç ve bitiş noktalarınız arasındaki mesafe ve zaman ölçümünü matrisler üzerinden yapabilmek için kullanılmaktadır. Distance Matrix API kullanarak 25 başlangıç ve 25 bitiş noktası tayin etme olanağı bulunmaktadır. •**Geocoding API**: Bir adrese dair bilgileri koordinata çevirme ya da bir koordinatı adrese çevirme işlemleri yapılmaktadır. Bu sayede adrese dayalı büyük hacimli verilerin dağılımı harita üzerinde görüntülenebilmektedir.

Google Maps Api Çalışma Şekli Google Maps API için iki farklı çalışma yöntemi bulunmaktadır. Bunlardan ilki bireysel olarak online erişim yöntemi, diğeri ise Google partnerleri aracılığı ile erişimdir. Google Maps API'ları internette açık kaynak kodlu olarak bulunmaktadır. Hazır kodlar bir API Key yardımıyla sorgulanabilir haldedir.

Projemizde kullandığımız bir kod parçasını örnek olarak gösterip açıklayacak olursak **,**

```
public Form1()
    {
        InitializeComponent();
GMapProviders.GoogleMap.ApiKey =
@"AIzaSyDjMyeORm1t3nK01oklJfUc6LiF
yySQYnM";
    }
```

Görülen kodda, Google üzerinden aldığımız api key'in projemize entegrasyonunu gerçekleştirdik.

Bulut Veritabanı Kullanımı, Google Cloud Platform, GCP maliyet yönetimi, veri yönetimi ve video teslimi ile yapay zeka ve makine öğrenimi araçları dahil olmak üzere bir dizi bilgi işlem hizmeti sunar. Google Bulut Platformu, Google'ın Google Arama, Gmail ve Google Fotoğraflar gibi son

kullanıcı ürünleri için dahili olarak kullandığı bulut altyapısı üzerinde çalışan bilgi işlem, ağ iletişimi, depolama, büyük veri, makine öğrenimi ve yönetim hizmetlerinden oluşan bir koleksiyondur.

Günümüzde **bulut bilişim**, donanım ve yazılım ürünlerinin uzaktan (veri merkezlerinde) ve uygun ölçekte bir arada var olmasına izin verir. Bu ürünler birlikte belirli hizmetler sunmak için çalışır.

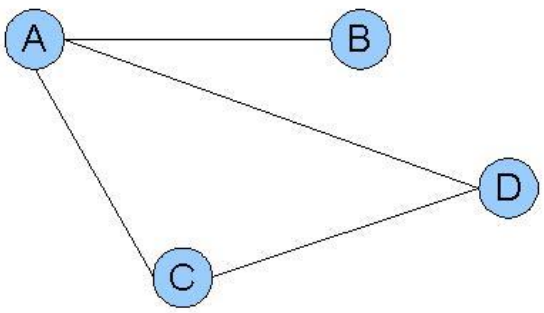
Kullanıcılar genellikle ihtiyaç duydukları araçlara web tabanlı bir arayüz üzerinden erişebilir, bunları yönetebilir ve kullanabilir. Aynısı **Google Cloud Platform hizmetleri** için de geçerlidir.

Kullanıcılar, Google Cloud Platform ile çalışırken hizmet erişilebilirliğinin yanı sıra esneklik de kazanır. Her hizmet “müşterinin isteği doğrultusunda” sunulur ve kullanıcıların ihtiyaç duydukları altyapıyı oluşturmak için çeşitli kaynaklardan yararlanmasına olanak tanır.

Graf , Bilgisayar dünyasında bulunan ve gerçek hayatta çeşitli sebeplerle karşılaşılan yapıları temsil amacıyla kullanılan şekillerdir. Örneğin bir bilgisayar ağını, karakenarları haritasını veya bir karar ağacını graflar kullanarak temsil etmek mümkündür.

Bilgisayar bilimleri çeşitli uygulamalarda karşılaşılan bu yapıları ifade etmek için çeşitli matematiksel ve görsel yöntemlerden faydalanır. Buna göre bir grafta bulunan varlıklar düğümler ile ifade edilmekte, bu varlıklar arasındaki ilişkiler ise graftaki kenarlar ile ifade edilmektedir.

Grafları kenarların yönlü olup olmamasına göre, yönlü graflar ve yönsüz graflar olarak ikiye ayırmak mümkündür. Ayrıca kenarların değer almasına göre değerli graflar veya değersiz graflar isimleri verilebilir.



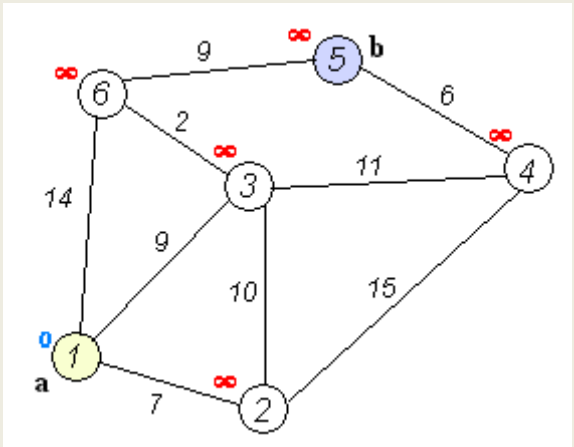
Örneğin yukarıdaki grafta 4 düğümden ve 4 kenardan oluşan bir graf gösterilmektedir. Bu grafı G= ({A,B,C,D} , {(A,B),(A,C),(C,D),(A,D)}) şeklinde ifade etmek mümkündür. Dolayısıyla graflar G= (V,E) şeklinde yanı düğümler ve kenarlar şeklinde yazılmaktadır.

Dijkstra algoritması , bir bir algoritma bulmak için kısa yolları arasındaki düğüm A grafiği, örneğin, temsil edebilir, yol ağları. Bu tarafından tasarlandı bilgisayar bilimcisi Edsger Dijkstra 1956 yılında ve üç yıl sonra yayınlandı.

Algoritma birçok değişkende mevcuttur; Dijkstra'nın orijinal varyant iki düğüm arasındaki en kısa yolu buldum, ancak daha yaygın bir varyantı "kaynak" düğüm olarak tek bir düğüm giderir ve üreten, grafikte tüm diğer düğümlere kaynaktan kısa yolları bulur en kısa yol ağacı.

Grafikteki belirli bir kaynak düğüm için, algoritma bu düğüm ile diğerleri arasındaki en kısa yolu bulur.[3] Hedef düğüme en kısa yol belirlendikten sonra algoritmayı durdurarak tek bir düğümden tek bir hedef düğüme en kısa yolları bulmak için de kullanılabilir. Örneğin, grafiğin düğümleri şehirleri temsil ediyorsa ve kenar yol maliyetleri doğrudan bir

yolla bağlı şehirler çiftleri arasındaki sürüş mesafelerini temsil ediyorsa, Dijkstra algoritması bir şehir ve diğer tüm şehirler arasındaki en kısa rotayı bulmak için kullanılabilir. Sonuç olarak, en kısa yol algoritması, ağ yönlendirme protokollerinde , en önemlisi IS-IS ve Açık Kısa Yol Yol İlkinde (OSPF) yaygın olarak kullanılır .

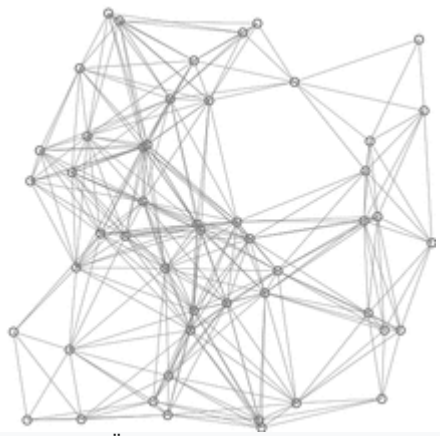


A ve *b* arasındaki en kısa yolu bulmak için Dijkstra algoritması . En düşük mesafeyle ziyaret edilmemiş köşeyi seçer, keşfedilmemiş her

komşuya olan mesafeyi hesaplar ve komşunun mesafesini küçükse güncelleştirir.).	
Sınıf	Arama algoritması
Veri yapısı	Grafik
En kötü durum performansı	O(|E| + |V| log |V|)

Dijkstra'nın orijinal algoritması kullanmak değildir min-öncelik sırası ve çalışır zaman . Bu algoritma fikri Leyzorek ve ark. 1957. Uygulama, bir Fibonacci yığın tarafından uygulanan ve en az bir öncelik sırasına dayanılarak yürütülüyor (nerede Kenar sayısıdır) Fredman & Tarjan 1984'ten kaynaklanmaktadır . Bu asimptotik bilinen en hızlı tek kaynak en kısa yol algoritması keyfi için yönlendirilmiş grafikler sınırsız negatif olmayan ağırlıklar ile. Bununla birlikte, özel durumlar (sınırlanmış / tamsayılı ağırlıklar, yönlendirilmiş asiklik grafikleri vb.) Gerçekten Uzmanlaşmış türevlerde ayrıntılı olarak geliştirilebilir .

Bazı alanlarda, özellikle yapay zeka, Dijkstra'nın algoritması veya bunun bir varyantı, **tekdüze** maliyetli **arama** olarak bilinir ve daha iyi ilk arama fikrinin bir örneği olarak formüle edilir.



Dijkstra'nın Öklid uzaklıklarına dayalı algoritmasının bir demosu. Kırmızı çizgiler, ör., *U* ve prev [*u*] ' yı bağlayan en kısa yoldur.

2.Tasarım

Akış şeması

Algoritma:

Başlangıç olduğumuz düğüme **ilk düğüm olsun** . ***Y*** **düğümünün mesafesinin başlangıçtaki düğümden *Y'ye*** olan **uzaklığı olmasına** izin verin. Dijkstra algoritması bazı başlangıç mesafesi değerlerini atayacak ve onları adım adım geliştirmeye çalışacaktır.

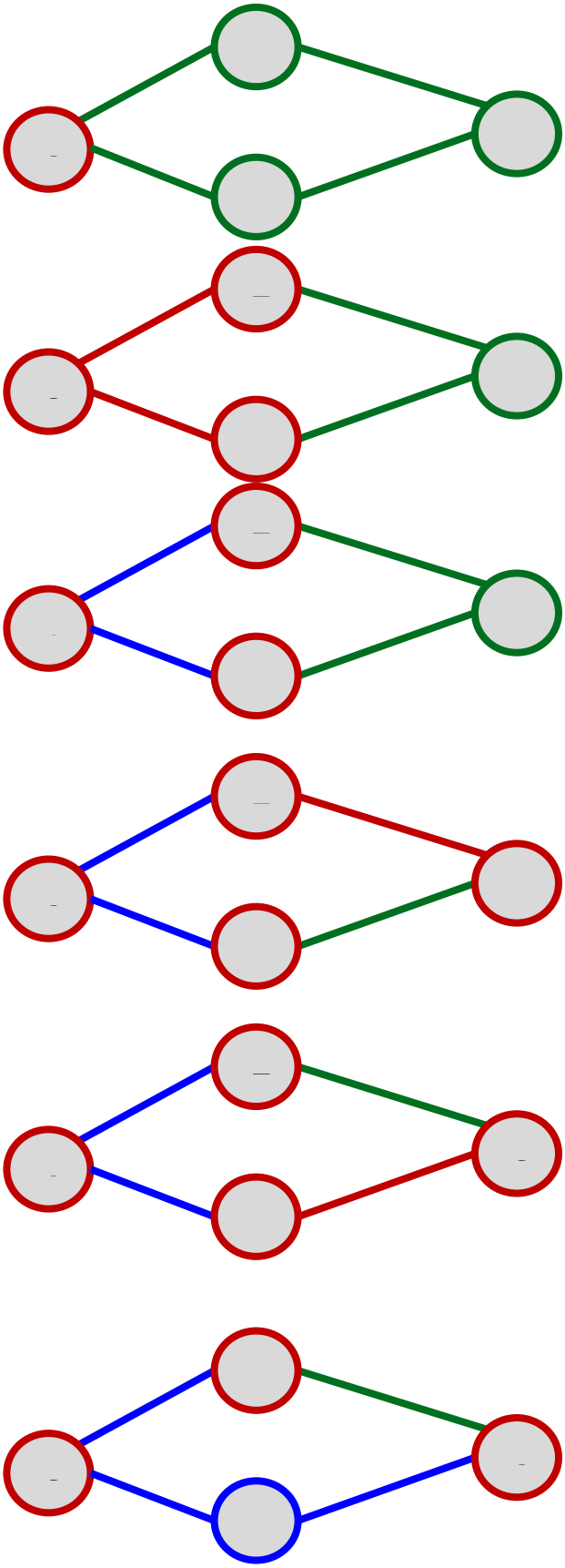
1. Her düğüme geçici bir mesafe değeri atayın: ilk düğümümüz için sıfıra ve diğer tüm düğümler için sonsuza ayarlayın.
2. İlk düğümü geçerli olarak ayarlayın. Tüm diğer düğümleri ziyaret edilmemiş olarak işaretleyin. *Ziyaret edilmemiş küme* diye adlandırılmayan tüm düğümlerin bir kümesi *oluşturun* .
3. Mevcut düğüm için, ziyaret edilmemiş tüm komşularını göz önünde bulundurun ve geçici *uzaklıklarını* hesaplayın. Yeni hesaplanan *geçici* uzaklığı ile atanan değere *karşılaştırın* ve küçük olanı atayın. Örneğin, mevcut düğüm *A* 6'lık bir mesafe ile işaretlenmişse ve onu bir komşu *B'ye* bağlayan kenarın uzunluğu 2 ise, *B'ye* (*A* yoluyla) mesafesi 6 + 2 = 8 olacaktır. B önceden önceden belirlenmişse 8'den büyük bir mesafe işaretlenmişse, 8'e değiştirin. Aksi halde geçerli değeri saklayın.
4. Şu anki düğümün tüm komşularını göz önünde bulundurduğumuzda, geçerli düğümü ziyaret edilmiş olarak *işaretleyin* ve ziyaret *edilmemiş kümeden* kaldırın . Ziyaret edilen bir düğüm hiçbir zaman tekrar kontrol edilmez.
5. Hedef düğüm ziyaret edildi olarak işaretlenmişse (iki belirli düğüm arasında bir rota planlarken) veya ziyaret edilmemiş kümedeki düğümler arasındaki en küçük geçici mesafe sonsuz *olduğunda* (tam bir geçiş planlarken; ilk düğüm arasında hiçbir bağlantı olmadığında gerçekleşirse) Ve kalan istenmeyen düğümler), sonra durdurun. Algoritma tamamlandı.
6. Aksi takdirde, en küçük geçici olmayan mesafe ile işaretlenmiş olan ziyaret edilmemiş düğümü seçin, onu yeni "geçerli düğüm" olarak ayarlayın ve 3. adıma dönün.

Sözde Kod

```
1   fonksiyon Dijkstra ( Grafik , kaynak ):
 2
 3 köşe kümesi Q'yu oluştur
 4
```

```
 5       her köşe v de Grafik :
// Başlatma
 6 dist [ v ] ← INFINITY
// v kaynaktan Bilinmeyen mesafe
 7 prev [ v ] ← tanımlanmamış
kaynak optimal yolu // Önceki düğüm
 8 eklenti v için Q
// All Başlangıçtaki Q düğümleri
(istenmeyen düğümler)
 9
10 dist [ kaynak ] ← 0
// kaynaktan kaynağa uzaklık
11
12       iken  Q boş değil:
13           u içinde köşe ← Q min dist
ile [u]     En az mesafe // Düğüm ilk
seçilecektir
14 çıkarın u dan Q
15
16           , u'nun her komşusu için v
:            // burada v hala Q'da
bulunur.
17               alt ← dist [ u ] +
length ( u , v )
18               if  alt <dist [ v ]:
// v'ye kısa bir yol
bulunmuştur 19 dist [ v ] ← alt
20 prev [ v ] ← u
21
22       return dist [], önceki []
```

Aşağıda dijikstra algoritmasının şekilli örneği verilmiştir:



Yazılım mimarisi

```
1   fonksiyon Dijkstra ( Grafik , kaynak ):
2 dist [ kaynak ] ← 0
// başlatma
3
4 köşe kümesi Q'yu oluştur
5
6       her köşe v içinde Grafik :
7           eğer  v ≠ kaynak
8 dist [ v ] ← INFINITY
// v kaynaktan Bilinmeyen mesafe
9 prev [ v ] ← tanımlanmamış
v // Öncül
10
11          S .add_with_priority ( h , Dist [ h ])
12
13
14      iken  Q boş değil:
// Ana döngü
15          u ← Q .extract_min () // Her komşu v için u'nun en iyi
16          noktasını çıkarın ve döndürün                    :
// yalnızca v hala hâlâ Q
17'de               haldedir alt ← dist [ u ] + Uzunluk ( u , v )
18              if  alt <dist [ v ]
19 dist [ v ] ← alt
20 ön [ v ] ← u
21                  Q .decrease_priority ( v , alt )
22
23      return dist [], önceki []
```

Referanslar

- Fredman, Michael Lawrence ; Tarjan, Robert E. (1984). Fibonacci yığınları ve geliştirilmiş ağ optimizasyon algoritmalarındaki kullanımları . Bilgisayar Biliminin Temelleri Üzerine 25. Yıllık Sempozyum. IEEE . S. 338-346. doi : 10,1109 / SFCS.1984.715934 .
- https://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html
- https://www.muhendisbeyinler.net/graf-teorisi-nedir/
- Arama, Robert B. (1969). "Algoritma 360: Topolojik sıralamaya sahip en kısa yol ormanı [H]". ACM'nin iletişimi . **12** (11): 632-633. doi : 10,1145 / 363.269,363610 .
- web.karabuk.edu.tr/ismail.karas/759/Sunu1_esas.pdf
- http://www.vogella.com/tutorials/JavaAlgorithmsDijkstra/article.html
- https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
- Cormen, Thomas H .; Leiserson, Charles E .; Rivest, Ronald L .; Stein, Clifford (2001). "Bölüm 24.3: Dijkstra'nın algoritması". Algoritmalara Giriş (İkinci baskı). MIT Press ve McGraw-Hill . Pp. 595-601. ISBN 0-262-03293-7
- veriyapilarivealgoritmalarcplusplus.blogspot.com/2015/05/graf-veri-modeli.html
- http://bilgisayarkavramlari.sadievrenseker.com/2010/05/13/dijkstra-algoritmasi-2/
- Fredman, Michael Lawrence ; Tarjan, Robert E. (1987). "Fibonacci yığınları ve geliştirilmiş ağ optimizasyon algoritmalarındaki kullanımları" . Bilgisayar Mühendisliği Derneği Dergisi . **34** (3): 596-615. doi : 10,1145 / 28.869,28874 .
- Knuth, D.E. (1977). "A Generalization of Dijkstra's Algorithm". Information Processing Letters. **6** (1): 1–5. doi:10.1016/0020-0190(77)90002-3.