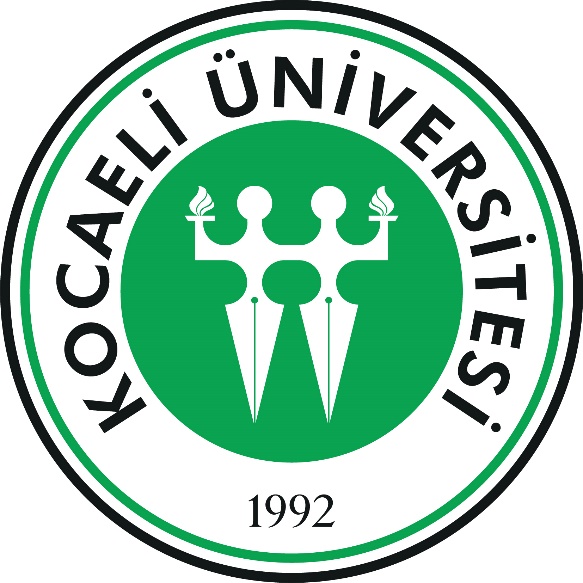
****

**T.C.**

**Kocaeli Üniversitesi**

**Teknoloji Fakültesi**

**Bilişim Sistemleri Mühendisliği**

**PROJE TAKİP SİSTEMİ**

**Feyyaz Can Köse**

**Mert Bulut**

**Gizem Coşkun**

**Mehmet Ali Çelik**

**Özgür Aslan**

**2021-2022 BAHAR DÖNEMİ YAZLIM LABORATUVARI-II DERSİ**

**PROJE - II**

**DANIŞMAN-ÖĞRETİM GÖREVLİSİ**

**Dr. Öğr. Üyesi Önder YAKUT**

**Öğr. Gör. Adnan Sondaş**

**Masaüstü Uygulaması Geliştirme**

**Gezgin Satıcı Problemini Dijkstra Algoritmasını Kullanarak Çözme**

**Desktop Application Development**

**Solving the Traveling Salesman Problem**

**Using Dijkstra's Algorithm**

Feyyaz Can Köse, Mert Bulut, Gizem Coşkun,

Mehmet Ali Çelik, Özgür Aslan

[gizemcsk1@gmail.com](mailto:gizemcsk1@gmail.com) , mail[@mertbulut.com](mailto:mrtebulute@gmail.com) , --, --, --

Bilişim Sistemleri Mühendisliği

Kocaeli Üniversitesi

**ÖZET**

Bir masaüstü uygulamasının nasıl ve hangi adımlarla geliştirileceğinden ve gezgin satıcı problemini dijkstra algoritmasını kullanarak nasıl çözebileceğimizden bahsedilmiştir. Öncelikle bir fikir üretmek, o fikir hakında araştırmalar yapmak, araştırmaların tartışılması, bir karar verildikten sonra tasarım aşamasına geçilmesi, daha sonra yazılım ve en son ise test aşamalarından oluşur. Hangi yöntemi, dillerin kullanıldığından bahsedilmiştir. Masaüstü uygulaması geliştirmek için kullanabileceğimiz C#, Python, Ruby, Java, gibi programlama dilleri bunlara örnektir.

Anahtar kelimeler: Masaüstü Uygulama Geliştirme, Desktop Application, C#, Dijkstra Algoritması

**ABSTRACT**

It is mentioned how and in which steps the application will be developed on the web platform. First of all, it consists of producing an idea, doing research on that idea, discussing the research, moving to the design phase after a decision is made, then the software and finally testing phases. Information about which development environments and which programming languages can be used to develop applications on the web platform is given. Programming languages such as C#, Visual Basic, PHP, Python, Ruby, Java, XML that we can use to develop web applications are examples of these. Based on various statistical findings as a result of research, it has been explained which direction the trends are in web applications development and where the progress can be.

Keywords: Desktop Application Development, Desktop Application, C#, Dijkstra Algorithm

**GİRİŞ**

Masaüstü uygulama geliştirme, bilgisayarlarınız ya da dizüstü bilgisayarlarınız ile standart bir şekilde çalışacak olan yazılımların geliştirilmesi sürecidir. Masaüstü uygulama geliştirmek isteyen birçok insan hangi dili kullanmalıyım, nereden başlamalıyım ve nerede nasıl geliştirebilirim soruları ile karmaşık bir durumda oluyor.

Bu raporda ve araştırmalar sonucunda masaütü uygulama geliştirmenin ne olduğunu, nasıl yapılabileceği, hangi geliştirme ortamı ve dillerin kullanılabileceğinden ve dijkstra algoritması ile en kısa yoldan bir problemi nasıl, hangi adımlarla çözebileceğinden bahsedilmiştir.

**1. Masaüstü Uygulama Geliştirme**

Masaüstü uygulaması geliştirme, yazılımın belirli cihazlar ve işletim sistemleri için geliştirildiği mobil uygulama geliştirmeye çok benzer. Bu durumda, masaüstü geliştirme, bilgisayarlarda çalışmak üzere oluşturulmuş yazılımı ifade eder. MacOS, Windows ve Linux gibi işletim sistemlerinde yerel olarak çalışan ve internete bağlı olması gerekmeyen yazılım uygulamalarıdır.

Hem bireysel olarak hem de firma bünyesinde sıklıkla kullanılan masaüstü uygulamaları, ihtiyaç duyulan gelişmelerin yürütülebilmesine olanak sağlan işlemlerin bütünüdür.

İstenilen program detayları verilerek bu özelliklere sahip olan yazılım uygulamaların geliştirilmesi günümüz teknoloji dünyasında artık çok kolaydır. Böylece hem kişisel hem de kurumsal anlamda işlemlerin hızlanması ve kolaylaştırılması mümkündür.

**1.1.Masaüstü Uygulama Geliştirmenin Avantajları**

* İş akışı büyük oranda hızlanır.
* Ürün stok takibi ve ürünlerin kontrolü kolaylıkla sağlanır.
* Muhasebe, müşteri cari bilgileri ve ön hesapları tutularak muhafaza edilir. Böylece iş akışı hızlanır.
* İnsan kontrolündeki hataları minimum düzeye indirir.
* İşletmelerin yönetimi kolaylaşır ve zaman sarfiyatı engellenir.
* İşletmedeki performans en üst düzeye çıkar ve verimlilik artar.

**2.UYGULAMA GELİŞTİRME AŞAMALARI**

**2.1. FİKİR ÜRETMEK**

Başladığımız hangi proje olursa olsun yapılması gereken en önemli aşama FİKİR üretme aşamasıdır. Dolayısıyla bir masaüstü uygulaması geliştirmek için geliştiricilerin birincil aşaması önce düşünmek ve bir fikir bulmaktır. Geliştirilmesi istenen uygulamaların sadece bir sorunu çözmesi gerekmez. İnsanların hayatlarına sadece eğlence katmak amacı ile de bir uygulama geliştirilebilir. Üretilen fikirler kişiye ait, özgün olması önemli bir kriterdir. daha önceden yapılmış bir uygulamanın birebirini yapmak çok da faydalı olmaz. Var olan uygulamalara yeni özellikler getirilebilir, kısacası özgün olmak önemlidir.

**2.1. WRITING**

Bu aşamada geliştirilmek istenen uygulama ile alakalı düşünülen, akla gelen neredeyse her şeyi daha sonradan toparlamak amacı ile bir kağıda veya not alabileceği herhangi bir yere yazmasıdır. Akla gelen “her şeyin” yazılması sonraki aşamalarda daha mantıklı ve güzel bir karar verilmesini sağlar.

**2.2. ARAŞTIRMA**

Bir uygulamanın belki de en önemli aşamalarından biri de doğru bir şekilde araştırma yapmaktır. İleride yaşanabilecek olası hatalardan kaçınmak için başlangıçta doğru ve detaylı bir araştırma yapmak önemlidir. Eğer bu aşama doğru bir şekilde veya hiç yapılmazsa ileride geri dönüşü zor ve zararı çok hatalarla karşılaşılabilir.

Bu aşamada; benzer uygulamaları kullananlar, kullanan kitlenin değerlendirmesi, avantajları ve dezavantajları gibi birçok konuda detaylı araştırma yapmak gerekir.

**2.3. KESİNLEŞTİRMEK**

**2.3.1. Uygulama Hedef Kitlesi**

Bir uygulama geliştirilirken, uygulamaya ait hedef kitlesinin kimler olduğunu belirlemek önemli aşamalardan biridir çünkü hedef kitle bilinde uygulamanın yapım ve gelişme aşamasında hedef kitleye göre değişimler ve geliştirmeler yapmak daha doğru olacaktır. Uygulama ile sunulan faydaların, hizmetlerin, ihtiyaçları nasıl ve ne ölçüde karşıladığını göz önünde bulundurmak oldukça büyük bir önem taşır. Ve tüm bunlar ne kadar doğru ve yeterli şekilde karşılanırsa kitle ve uygulama arasındaki bağ arttırılır.

**2.3.2. Gelir Modeli**

Bir uygulmadan kazanç sağlamak için birçok yöntem vardır. Bu yöntemlerden hangisinin geliştirilen uygulama için en iyisi olduğunu belirlemek için pazarda bulunan alternatif uygulamalarda hangi gelir modellerinin kullanıldığı incelenmelidir. Ayrıca, uygulamayı pazara sürdükten sonra hangi aşamada para kazanmaya başlanılacağı da düşünülmesi gerekir.

**Uygulamadan para kazanma yöntemleri:**

* Reklamlar: Günümüzde uygulamaların birçoğu reklam içeriyor ve gün geçtikçe reklam ile para kazanmak, uygulamalara giriş ile ücretten kazanılan miktara yaklaşıyor. Reklamlar ile kazanç sayesinde kullanıcılara uygulamayı kullanmaları için ücretsiz bir deneyim sağlayabilirsiniz. Ama uygulamanıza reklamlar eklerken bunu çok dengeli bir şekilde yapmalısınız. Birkaç dakikada bir araya reklam girerse, sayfanın tamamını reklamlar kaplarsa kullanıcılar bundan sıkılacak ve kullanmaktan kaçınacaktır. İkinci maddede de söylediğim gibi dengeyi kurmak önemlidir. Bu yüzden abartılmadan etkili olacak sayıda ve düzeyde reklamlar verilmelidir.
* Ücretli indirme: En az tercih edilen gelir modeli ücretli indirme/kullanmadır. Çünkü bir uygulamaya hedef kitlenin ücret vermesi ve yüksek rağbet görmesi için uygulamanın kullanıcılara çok iyi bir şekilde tanıtılması ve kullanıcıları iyi bir tanıtım sayesinde ikna etmesi gereklidir. Eğer bu yapılmazsa kullanıcı ücret vermekten çekinecektir ve beklenen gelir sağlanamayacaktır. Ayrıca ücretli uygulamaların korsan olarak dağıtılması en büyük sorunlardan birini teşkil eder.
* Uygulama içi satın alma: Ücretli indirme yerine tercih edilebilecek bir diğer gelir modeli uygulama için satın almadır. Bu gelir modeliyle uygulama ücretsiz sunulur ve kullanıcının deneme imkanı olur. Fakat uygulama içinde ücret ile alınabilecek premium özellikler ile istenilen gelir sağlanabilir. Ancak uygulamanın içerisinde kullanıcı taraflı cazip olmayan fiyatlar olması veya gerçek anlamda etkileyici olmayan içerikleri kullanıcıyı ekstra ücret ödeme konusunda geri çekecektir. Netflix, BluTV, Youtube Premium bunlara örnek olarak verilebilir
* Sosyal Medya Kullanımı: Sosyal medya kullanımı artık çok yaygın olduğu için sosyal medyayı kullanarak uygulamanızı pazarlamanız uygulamanızın viral olma şansını arttırır. Sosyal medyada paylaşacağınız içeriklerin kalitesi çok önemlidir bu sizin pazarlamada başarınızı arttırır. Uygulamanız için sosyal medyada insanların beğeni ve yorum yapacağı içerikler üretmeniz de etkileşiminizi ve uygulamanızın daha fazla insana ulaşmasını arttıracaktır.
* Sponsorluk: Uygulamanıza sponsorluk anlaşması yapmanız gelirinizin artmasında çok önemli bir artış sağlayabilir. Uygulamanızın tasarımından tanıtımına kadar birçok şeyde sponsor yardımcı olabilir.

**2.4 UYGULAMAYI TASARLAMAK**

Tasarlama aşamasında uygulamaya hem görsel hem işitsel bir şekil kazandırılır. Bu aşamalar aşağıdaki gibidir:

* Tasarım Araştırması : Bu adım kullanıcıyı direkt yoldan etkileyeceği için en önemli aşamalardan biridir. Tasarım yapımı geliştiricinin isteğine bağlı olarak isterse hazır isterse kendi tasarımı olacak şekilde bir başlangıç seçmelidir. Seçimine karar verdikten sonra varsa halihazırda bulunan uygulamaların tasarımlarını, kullanıcı ara yüzlerini, logolarını vs incelemelidir. Uygulamada var olması gereken içeriğin ise tasarım yapım aşamasına geçmeden hazırlanmış olmalıdır. Çünkü içeriği belli olmayan bir uygulamanın tasarımın yapılması ileride istenmeyen değişikliklerin yapmasına yol açabilir. Bu da hem zaman hem maddi olarak bir kayıba yol açar.
* Akış Şeması (Flowchart) : “Akış şeması: web uygulamanın içindeki ekranlar arasındaki geçişleri, hangi ekrandan hangi ekrana geçileceğini ve uygulama içindeki diğer navigasyonel fikirlerin sürecinin her bir aşamasını (ara yüzünün) oklar ile birbirlerine bağlanarak tasarım süreci başlamadan önce görsel olarak düzenlenmesidir.”
* Digital branding : Akış şemasının ardından uygulama için bir logo tasarlamak tasarımın resmi bir hal almasını sağlayacaktır. Bu logo, uygulamanın açık ya da kapalı olduğu her an onu simgeleyecek ana bir görsel olacağı için uygulamaya özel ve uygun bir logo tasarlanmalıdır. Logo tasarlandıktan sonra görsel içerik de logo tasarımına uygun olmalıdır. Sayfadaki düzeni, renk uyumunu ve bütünlüğü sağlamak uygulama için önemli bir kriterdir.
* Wireframing: Geliştirilen mobil uygulama tasarımının ana planıdır. Bir diğer deyişle tasarımın görsel bir iskeletidir. Burada wireframe, elle çizilebileceği gibi artık dijital ortamda da çeşitli araçlar ile hazırlanabilir. Wireframe, tasarımın yapım sürecinde geliştiricilere bir yol gösterir. Kurgulanan yapıya göre nasıl hareket edilmesi gerektiğini gösterir. Bu yönünden dolayı aynı zamanda tasarımınızı test etmenizi de sağlar.
* Arayüz detaylarının tasarımı: Kullanıcı Arayüzünün, daha önce birbirleriyle genel hatlarıyla etkileşimleri belirlenen (wireframe ile) görsel elemanların gerçeğe dönüştürülmesi ve tasarlanması aşamasıdır. Bu tasarımı, daha önce tasarlanmış uygulamanın ana logosuna göre yapmak gerekir çünkü daha önce de belirttiğim gibi sayfanın bütünlüğü, sadeliği ve özeninin bozulmaması çok önemlidir. Bu tasarım yapılırken kullanılabilecek, tasarımı hazır bir şekilde yapabilmemiz için birtakım siteler mevcuttur fakat bu durum özgün tasarımlar hazırlanmasının önüne geçer. Arayüzü tasarlarken renkleri, şekilleri, çerçeveleri gibi tüm görsel detayları iyi düşünmek ve bir bütünlüğe uyarak tasarımı yapmak gerekir. Bu süreçte tutarlı bir tasarım yapmak uygulamanın geliştirilme sürecinde daha az sorun ile çalışmasını sağlar.

**2.5. Uygulama Geliştirme Yaklaşımını Belirlemek (Kullanılacak teknolojileri seçmek)**

Tasarım tamamlandıktan sonra asıl işlemlerin yapıldığı, kısım yazılım aşamasıdır. Uygulama kullanımı süresince yapılacak her işlemin kullanıcıyı yormadan, sıkmadan olması gereklidir. Bunu yapabilmek için iyi bir algoritma yapısı kurulmuş olması en önemli şarttır. Uygulama yapım sürecinde doğru teknolojileri ve ideyi seçmek de önemlidir.

Masaüstü uygulaması geliştirirken yazdığımız kodlar ikiye bölünür. Bir taraf tarayıcının anlayacağı görselliğin olduğu ve biz buna client-side yani frontend darken diğeri ise server tarafında database gibi işlemleri yaptığımız taraftır. Aralarında request ve response bağlantısı vardır. Yani bir taraf istekte bulunurken diğer taraf bu isteğe bir cevap verir.

**2.5.2. Client-Side Programlama (Frontend)**

Client, türkçesi müşteri tarafı anlamına gelmektedir. Sayfaları server’da request eder ve cevapları kullanıcı tarafında gösterir. Kodlar sunucuda çalışmaz direkt browser tarafından yorumlanır. Yapılan işlemlerde sunucu ile bağlantı kurulmaz ve müşterinin bilgisayarında ekrana basılır.

2.5.2.1. Client-Side Hangi Dillerde Yazılır?

* JavaScript
* HTML
* CSS
* Ajax
* Flash
* JQuery veya TreeJS

Avantajları:

* Sunucuya fazla yük binmez
* Editörlerin birtakım özelliklerini değiştirmek için server gereksiz yere meşgul edilmez. Basit işlemler client tarayıcısında kolayca halledilir.

Dezavantajları:

* Her browser her script dilini desteklemez. Dolayısıyla bazı browserler’da çalışırken sorun yaratabilir.
* Sayfayı incele denildiğinde kaynak kodlar görülür bu da güvenli değildir ve risk taşır.
* Kaynak kodlar Client-side taraflı yazıldıysa kodları inceleye birisi proje ile alakalı her türlü bilgiye kolayca ulaşabilir. (şifre,tablo adları vs.)

**2.5.3. Server-Side Programlama (Backend)**

Yazılan kodlar Client tarafında browser’a gitmez. Kodlar sunucu üzerinde çalışır ve sonuçlar client tarafına gönderilir. Sunucu üzerinde çalışan kodlar direkt HTML çıktısı olarak browser’a gönderildiğinden dolayı her browser’da sorunsuz çalışır.

Avantajları:

* Kodlar sunucudan istemciye gönderilmediği için, kodlara bakmak isteyen biri tarafından kodlar görülmez. Güvenlidir.
* Kodlar sunucuda çalıştırıldığı için browser’a olan bağımlılık ortadan kalkar.

Dezavantajları:

* Kodlar tamamen sunucuda çalışacağı için sunucuya çok iş düşer. Hem maddi hem de site performans açısından istenilen durumda olmaz.
* Kodlar sunucuda çalıştırıldığı için iyi sunucu özellikleri olması gerekir. (RAM, CPU,HDD)

2.5.3.1. Server-Side Hangi Dillerde Yazılır?

* C, C#, C++, VB
* Java, Perl, Python, Ruby
* ASP
* PHP
* Shell

**2.5.4. Masaütüstü Uygulama Geliştirme Frameworkleri**

WPF: “Windows Presentation Foundation, masaüstü istemci uygulamaları oluşturan bir kullanıcı arabirimi çerçevesidir.”

UWP: “UWP, masaüstü uygulama geliştirme için bir başka önemli çerçevedir. Aynı zamanda, .NET platformunun popülaritesi için oldukça itibarlı bir çerçevedir. Bunun nedeni, geliştiricilerin platformlar arası masaüstü uygulamaları oluşturmasına izin vermesidir.”

Cocoa: “Masaüstü işletim sistemi macOS için Apple'ın yerel nesne yönelimli programlama ve uygulama programlama arayüzüdür.”

Electron Js: “ HTML ve CSS ile masaüstü uygulamalar oluşturmamızı sağlayan bir frameworktür.”

Swing: “Java Koleksiyon Yapısı (JFC- Java Collections Framework) nın bir üyesidir ve tamamen Java dili ile yazılmıştır. **Swing**, bütün platformlarda aynı şekilde davranabilen ve görünümü değişmeyen (AWT bileşenleri, Windows'ta farklı, MacOS'da farklı görülebiliyordu) arayüzler sunar ve platform bağımsızdır.”

**2.6. UYGULAMAYI TEST ETMEK**

Kullanıcı Deneyimi (UX-User Experience)'ni en iyi hale getirebilmek adına yazılan kodları ve tüm arayüzü, kullanıcı gözüyle incelemek ve defalarca test etmek gerekir. [1][19]

**3. Uygulama Geliştirme Ortamları**

“Programcıların bilgisayar programı yazarken kullandığı farklı programlama dillerini birleştirmelerini sağlar.” Bilgisayara yükleyebileceğimiz bir yazılımdır.

IDE’ler kaynak kodlarını düzenleme, çalıştırma, dosyaları çalıştırma ve hataları ayıklama gibi özelliklere sahiptir.

* Kaynak Kodu Düzenleme

Kod yazmak, programlamanın en temel ve önemli parçasıdır. Satırlarca kod yazıyoruz ve ideler bize daha doğru, düzenli kod yazma imkanı sunuyor. Kodları otomatik tamamlama, vurgulama gibi özellikleri işlerimizi kolaylaştıryor.

* Otomatik Tamamlama

Ide’de var olan programlama dili ile yazdığınızda ideler ne yazmak istediğini tahmin edebilir ve kodu kısayollar ile tamamlamanız için seçenekler sunar.

* Hata Ayıklama

Bir kodu hatalı yazmak, hata yapmak, eksik yazmak gibi birçok sorun karşılaşılşabilecek en normal şeylerden biridir. Bu konuda ide’ler programcıların yardımına koşar ve kodlardaki hataları kolayca bulmalarında olanak sağlayan hataları ayıklama araçları sağlar. Ayrıca kodlama yaparken programcıya yol gösterici niteliğinde olan ipuçları sağlar.

**3.1.Visual Studio**

En popüler ve en iyi IDE’lerden ve web geliştirme seçeneklerinden biridir. Kod yazarken yazdıklarınızı tahmin eder ve otomatik tamamlar. Bunların ötesinde ekibinizle canlı olarak birbirinizle bağlantı kurarak iletişim halinde olabilirsiniz. Azure desteğinin yanı sıra web, mobil, uygulama ve oyun geliştirme, [C](https://tr.wikipedia.org/wiki/C_(programlama_dili))/[C++](https://tr.wikipedia.org/wiki/C%2B%2B) (Görsel yoluyla C++), [VB.NET](https://tr.wikipedia.org/wiki/Visual_Basic_.NET) (Visual Basic .NET üzerinden), [C#](https://tr.wikipedia.org/wiki/C_Sharp) (Visual C# ile), ve [F#](https://tr.wikipedia.org/wiki/F_Sharp) (Visual Studio 2010 itibarıyla) içermektedir.

Avantajları:

* Kod düzenleme kolaylığı
* Kodlamaya hazır projeler oluşturabilme
* Dinamik siteler hazırlayabilme
* Çok fazla eklenti ile özelleştirebilme rahatlığı
* Özelleştirilmiş snippet'lar kullanabilme
* Kolay tuş kombinasyonları ile kodlara rahat bir şekilde ulaşabilme ve kodlar üzerinde düzenleme yapabilme
* Arayüzü özelleştirebilme

**3.3. PhpStorm**

“PhpStorm çapraz platform bir PHP tümleşik geliştirme ortamıdır (IDE).İlk olarak 2009 yılında yayınlanmıştır ve altyapı olarak yine JetBrains firması tarafından geliştirilmiştir.”

Web geliştirme için en iyi IDE’lerden biridir denilebilir. PHPStorm daha çok PHP’ye odaklanmıştır ancak PHP dışında da dilleri desteklemektedir. HTML, CSS, Sass, Less, JavaScript dahil olmak üzere daha fazla dil örnek olarak verilebilir. Kod analizi, hata ayıklama, debugger desteği gibi özelliklere sahiptir.

Artıları

* Zengin bir kod editörü ve otomatik kod tamamlama özelliklerine sahiptir.
* Yerel veya uzak olarak çalıştırılabilir.
* Windows, Mac OS X ve Linux’ta çalışabilme özelliği vardır.
* Kodların kaybolmasını engellemek adına kodları kaydetmeden önce geliştiricilere karşılaştırma özelliği sunar.
* Git desteği sunar.

## 3.4.InteliJ IDEA

IntelliJ IDEA, Jet Brains tarafından geliştirilen bir başka IDE’dir. Bu IDE, ücretsiz versiyon olarak Community Sürümünü, ücretli ve extra özelliklere sahip sürüm olarak ise Ultimate versiyonunu öne sürmektedir.

Özellikleri

* Geniş veritabanı editörü ve UML tasarımcısı.
* Çoklu yapı sistemlerini desteklemesi.
* Test çalıştırıcısı kullanıcı arabirimi.
* Sürüm Düzenleyicisi, günlük dosyalarını içerir ve zaman çizelgesini hazırlar.
* Çoğu uygulama sunucusu için dağıtım ve hata ayıklama araçları.
* AIR Mobile, Android ve iOS cihazlarını destekler.
* Yeni başlayanlar için zorlayıcı bir IDE olabilir.

**4. MASAÜSTÜ UYGULAMA GELİŞTİRME DİLLERİ**

**4.3. Java**

“Java ilk olarak Sun Microsystems tarafından 1995 yılında piyasaya sürülen bir programlama dili ve bilgi işlem platformudur. Java neredeyse her türdeki ağ uygulamalarının temelini oluşturarak gömülü ve mobil uygulamalar, oyunlar, Web tabanlı içerik ve kurumsal yazılım geliştirme ve dağıtımı için küresel standarttır. Dünya genelinde 9 milyonu aşan geliştiriciyle Java, heyecan verici uygulamaları ve hizmetleri verimli bir şekilde geliştirilmesine ve dağıtılmasına olanak tanır.”

Birçok Java geliştiricisi ile özel bir topluluk oluşmuştur (Java Community) ve bu topluluk Java’yı daha çok geliştirmiş, genişletmiş, iyileştirmiş, test etmiş ve onaylamıştır. Uygulamaları karma bir şekilde çeşitli platformlarda çalışabilir hale getirerek daha fazla hizmet sunulmasını sağlamışlardır.

Avantajları

* Java’nın yazımı kolaydır, okunaklı ve kolay öğrenilebilir bir yapısı vardır.
* Günümüzdeki tüm işletim sistemleri Java'yı desteklemektedir.
* Java’nın en önemli özelliklerinden biri de farklı ortamlarda java derleyicisi olduğu sürece çalışılabilir yani taşınabilir olmasıdır.
* Java nesne yönelimli bir programlama dilidir ve yapı olarak C++ özelliklerine benzerdir ve java’da ilerlemek isteyen bir insanın OOP’yi iyi anlaması gerekli ve çok önemlidir.
* Yüksek performanslıdır ve derlenirken çok hızlı çalışır.
* Dinamiktir.
* Çok çeşitli kütüphanelere ve frameworklere sahiptir.
* Web tabanlı bir dildir.

Dezavantajları

* Java'nın en büyük dezavantajı, C/C++ dillerine göre çok yavaş olması ve java yorumlayıcısına ihtiyaç duymasıdır.
* ARM işlemcileri bu noktada sorunu çözebilmektedirler.

**4.4.Python**

**“Python**, [nesne yönelimli](https://tr.wikipedia.org/wiki/Nesne_y%C3%B6nelimli), [yorumlamalı](https://tr.wikipedia.org/wiki/Yorumlanan_programlama_dili), birimsel (modüler) ve etkileşimli [yüksek seviyeli](https://tr.wikipedia.org/w/index.php?title=Y%C3%BCksek_seviyeli_programlama_dili&action=edit&redlink=1) bir [programlama dilidir](https://tr.wikipedia.org/wiki/Programlama_dilleri).”

Başlangçta sadece gereksiz programlar için tamamlayıcı bir dil olarak nitelendirilirken son birkaç yılda altyapı yönetimi ve veri analizinde birinci sınıf bir programalama dili olmuştur.

Avantajları

* Basit ve sade bir dil olması nedeniyle kolay da öğrenilebilir bil dil olma özelliği taşır.
* Açık kaynak kodludur.
* Tüm platformlarda sorunsuz çalışabilir.
* Nesne yönelimli programlama olma özelliği taşımasının yanında prosedür yönelimli programlamayı da destekler. C++, C ya da Java gibi dillere kıyasla güçlü ama basit bir programlama dilir.
* Kütüphane bakımından çok geniştir.

Dezavantajları

* Kodların satır satır yüklenmesinden dolayı python yavaş bir dildir ve bu yüzden de kodu derlerken daha fazla iş yapması gerekir.
* Kullandığı bellek miktarı çok büyüktür ve ne yazık ki bellek açısından verimsizdir.
* Projede veritabanı kullanıldığında diğer teknolojilere göre ilkel ve az gelişmiş kalıyor.

Python ile neler yapılabilir?

* Machine Learning
* Web uygulamaları (Django)
* Rest API oluşturma
* Veri Analizi
* Masaüstü uygulamaları vb.

**Gezgin Satıcı Problemi**

Gün içinde yapmamız gereken, gitmemiz gereken birçok şey oluyor ve zaman bizim için çok önemli. Evden çıkıp okula, işe, markete, spora vs. gitmek ve en sonunda tekrar eve dönmek gibi tüm bunları yaparken en kısa yolu bulup hem zamandan hem maddi açıdan çok kaybetmemiz lazım. İşte bu olay gezgin satıcı problemi ile ilgilidir.

Özetlemek istenirse;

Bir kurye var;

Bu kurye, yemekleri {\displaystyle n\,}şehirde siparişleri sipariş verilen yere götürmek istiyor;

Öte yandan, mantıklı bir şekilde, bu kurye bu siparişleri mümkün olan en kısa şekilde ve her bir konuma maksimum bir kere uğrayarak siparişleri dağıtmak istiyor.

Problemin amacı da bu kuryeye mantıklı en kısa yolu sunmaktır.

Bu problemi ise şu şekilde çözebiliriz;

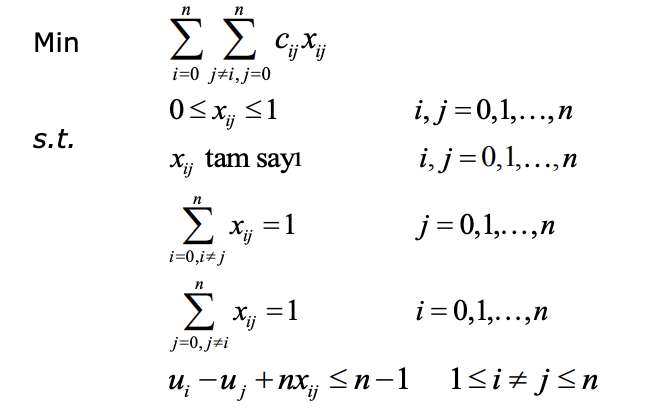
Başlangıçta gidebileceği n tane farklı konum var.

İlk konuma giden n-1 değişik yol var.

İkinci konıuma giden n-2 değişik yol var.

Dolayısıyla kuryenin n! değişik yol arasından seçim hakkı olacaktır.

**Gezgin Satıcı Problemi Matematiksel Modeli**



GSP’nin matematiksel modelinde kullanılan ij c , i. şehirden j. şehire olan uzaklığı (mesafe) ifade etmektedir. ij x ise i. şehirden j. şehire gidilmesi durumunda 1, diğer durumlarda 0 değerini alan bir parametredir. GSP modelinin amacı ij c , ij x çarpım toplamlarının minimize edilmesidir. Modelde n, şehir sayısını ifade ederken, u parametresi ise eksiksiz tam bir tur yapılması için koşulları sağlamak üzere modele eklenen yapay bir değişkendir.

**Gezgin Satıcı Problemi Kullanım Alanları**

• GSM operatörlerinin baz istasyonlarının yerleşim yerlerinin belirlenmesi,

• Birçok ulaşım ve lojistik uygulamaları,

• Malzeme akış sistem tasarımı,

• Araç rotalama problemleri,

• Depolardaki vinç güzergâhlarının programlaması,

• Stok alanındaki malzeme toplama problemleri,

• Uçaklar için havaalanı rotalaması,

• Elektronik devre tasarımı

• Röportaj zamanlama,

• Matbaa zamanlama,

• Bilgisayar kablolama,

• Ekip planlama,

• Misyon planlama,

• Navigasyon uydu sisteminin ölçme ağlarının tasarımı,

• Sipariş toplama.

**Dijkstra Algoritması**

Dijkstra Algoritması düğümler arasındaki en kısa yolu bulmak için kullanılmaktadır. Bu algoritmanın en önemli ve temel amacı bir düğümden başka bir düğüme giderken hem kolay ve kısa hem de en ucuz maliyetle nasıl gidileceğini hesaplamaktır.

**Dijkstra Algoritmasının Zayıf Yönü**

Algoritma ne yazık ki eksi (-) değer taşıyan bir kenar bulunması halinde başarılı bir şekilde çalışmaz. Çünkü eksi değerdeki kenarın sürekli olarak mevcut durumdan daha iyi bir sonuç üretmesi ve algoritmanın hiçbir zaman için kararlı hale gelememesidir.

**Dijkstra Algoritması Sözde Kod**

1. Başlangıç düğümü belirle.
2. Başlangıçta başlangıç düğümü dışındaki bütün düğümlere sonsuz değeri ata.
3. Başlangıç düğümünün komşusu olan düğümleri dolaş.
4. Dolaşılan değerlerin ulaşım mesafesini güncelle.
5. Güncellenen düğümlerin komşularını güncelle. Bütün düğümler güncellenene kadar bu işleme devam et.

**Dijkstra Algoritması Kodu**

class GFG {

static int V = 9;

int minDistance(int[] dist,bool[] sptSet) {

int min = int.MaxValue, min\_index = -1;

for (int v = 0; v < V; v++)

if (sptSet[v] == false && dist[v] <= min){

min = dist[v];

min\_index = v;}

return min\_index; }

void printSolution(int[] dist){

Console.Write("Vertex \t\t Distance "

+ "from Source\n");

for (int i = 0; i < V; i++)

Console.Write(i + " \t\t " + dist[i] + "\n");}

void dijkstra(int[, ] graph, int src){

int[] dist = new int[V];

bool[] sptSet = new bool[V];

for (int i = 0; i < V; i++) {

dist[i] = int.MaxValue;

sptSet[i] = false;}

dist[src] = 0;

for (int count = 0; count < V - 1; count++) {

int u = minDistance(dist, sptSet);

sptSet[u] = true;

for (int v = 0; v < V; v++)

if (!sptSet[v] && graph[u, v] != 0 && dist[u] != int.MaxValue && dist[u] + graph[u, v] < dist[v])

dist[v] = dist[u] + graph[u, v]; }

printSolution(dist);

public static void Main(){

int[, ] graph = new int[, ] { { 0, 4, 0, 0, 0, 0, 0, 8, 0 },

{ 4, 0, 8, 0, 0, 0, 0, 11, 0 },

{ 0, 8, 0, 7, 0, 4, 0, 0, 2 },

{ 0, 0, 7, 0, 9, 14, 0, 0, 0 },

{ 0, 0, 0, 9, 0, 10, 0, 0, 0 },

{ 0, 0, 4, 14, 10, 0, 2, 0, 0 },

{ 0, 0, 0, 0, 0, 2, 0, 1, 6 },

{ 8, 11, 0, 0, 0, 0, 1, 0, 7 },

{ 0, 0, 2, 0, 0, 0, 6, 7, 0 } };

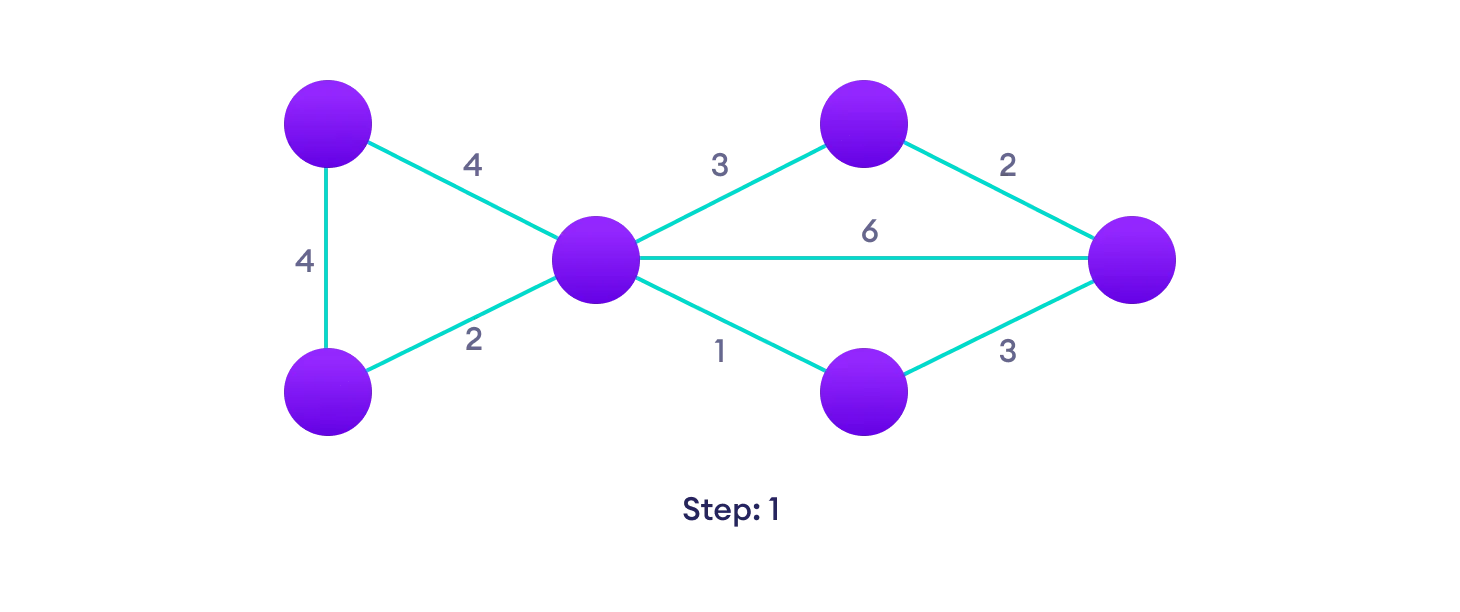
GFG t = new GFG();

t.dijkstra(graph, 0);

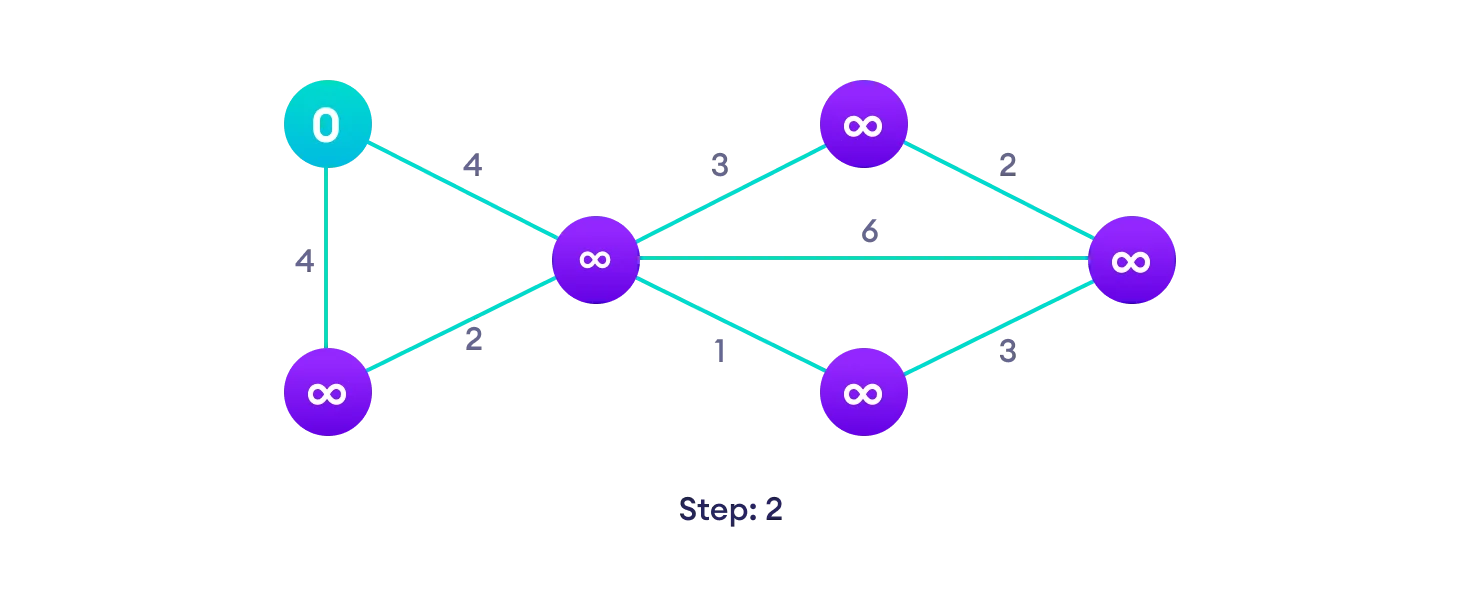
}

}

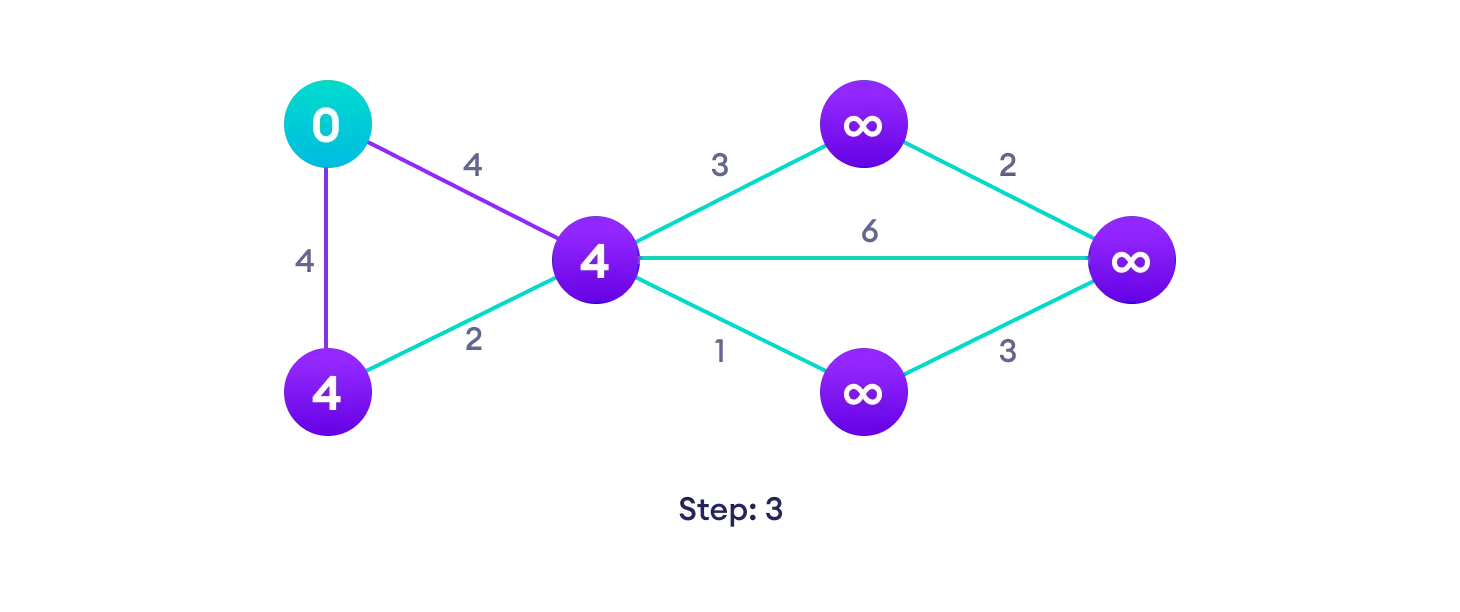
**Dijkstra Algoritması Çalışma Mantığı**



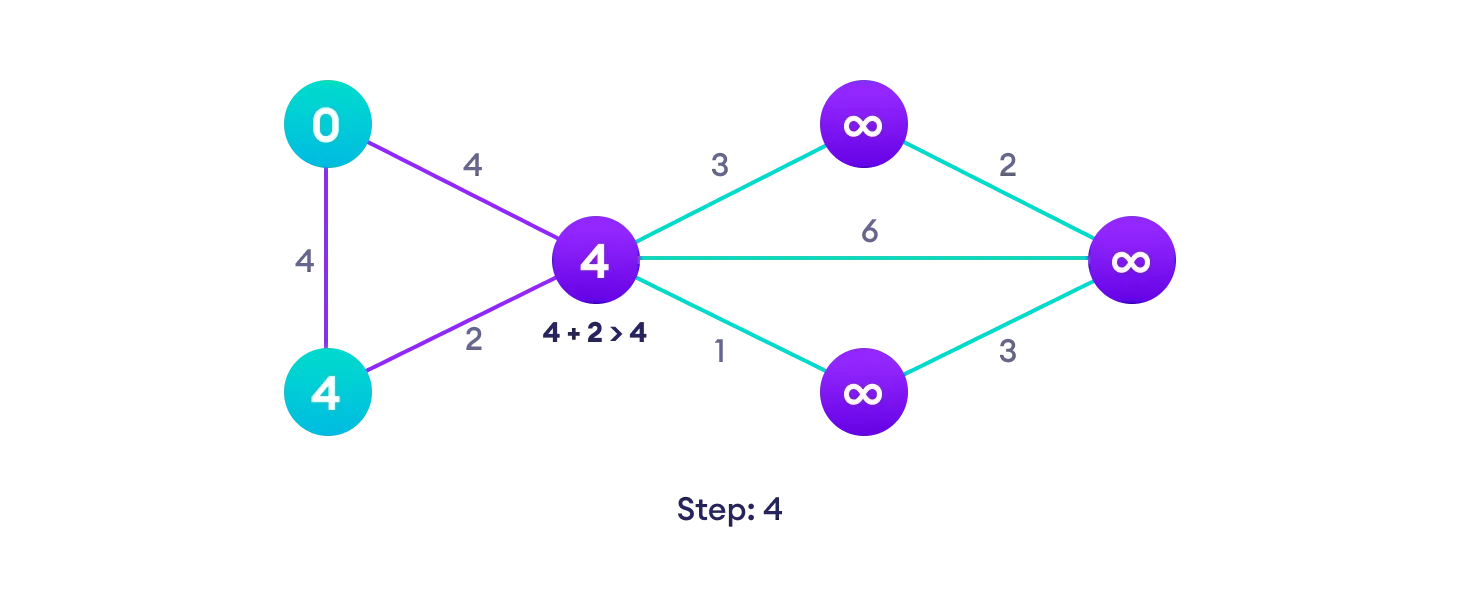
Öncelikle bir başlangıç düğümü seçiyoruz ve bu seçtiğimiz düğüm 0 oluyor.

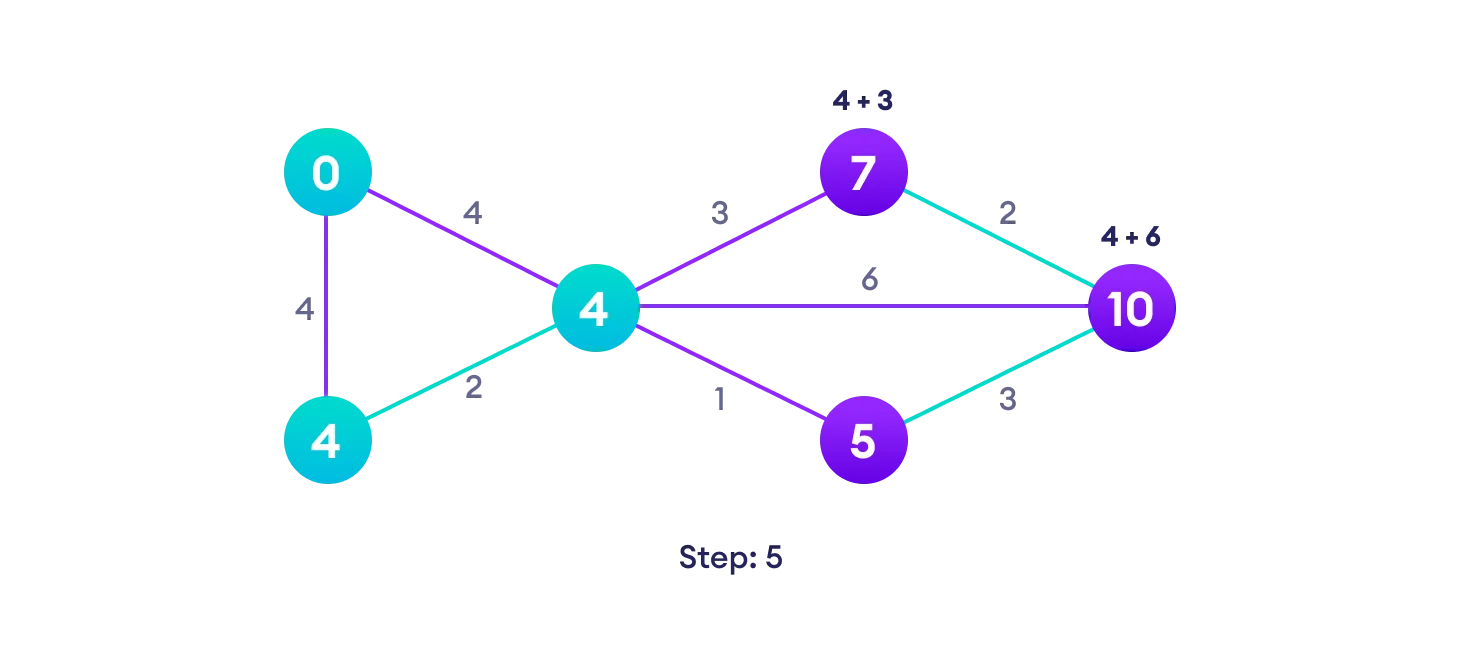


Başlangıç dışındaki tüm düğümlere sonsuz değeri atılıyor.

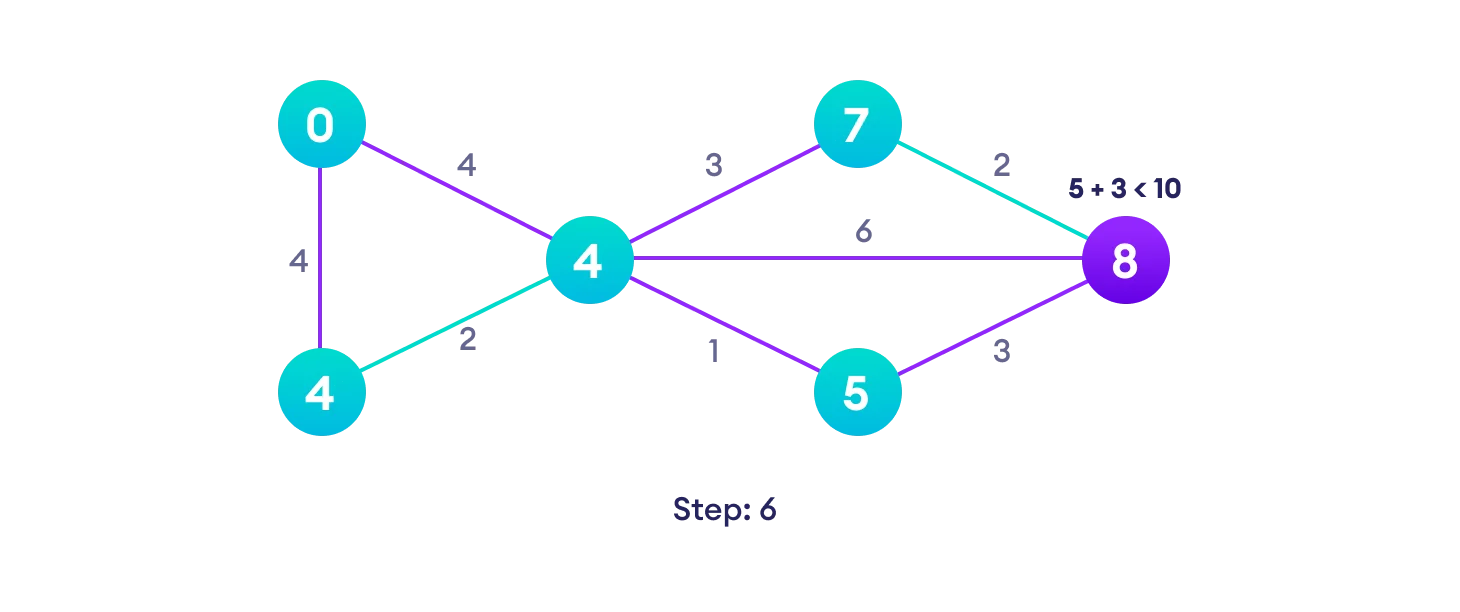


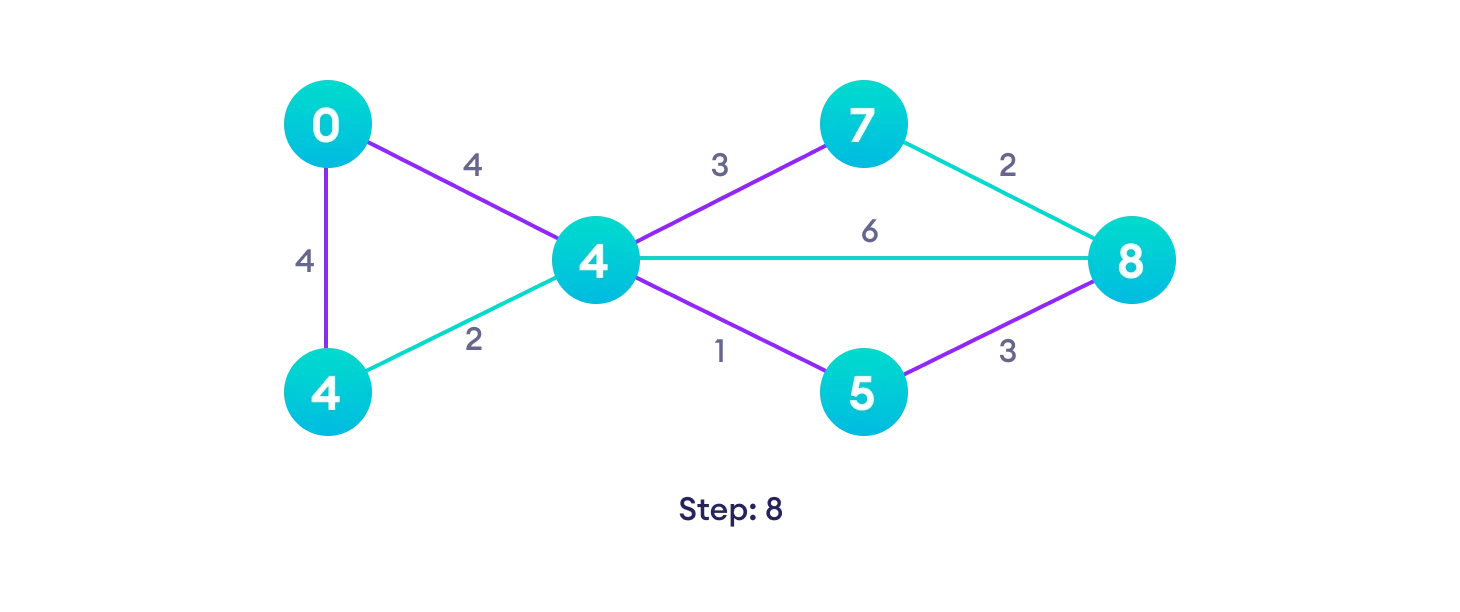
Daha sonra sırayla her düğümün komşusu güncellenerek ilerleniyor. Tüm düğümler güncellenene ve herhangi ekstra bir düğüm eklenmeyene kadar bu işlem yapılmaya devam edilir.



Eğer komşu yolun uzunluğu yeni yolun uzunluğundan küçükse onu güncellemiyoruz. 

Zaten ziyaret edilen yolları güncellemekten kaçınıyoruz.

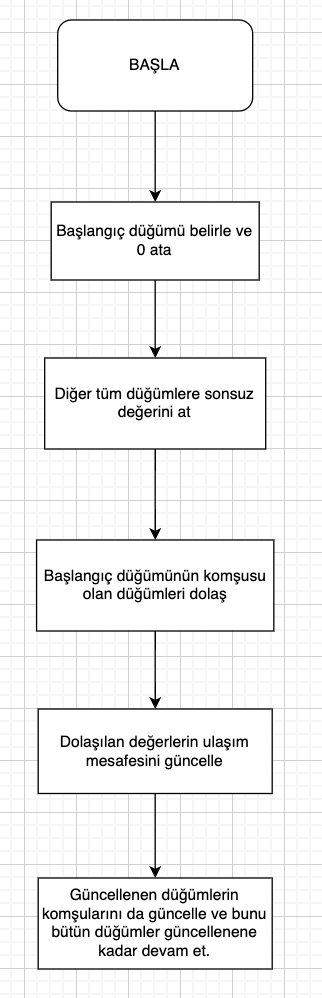


Her yinelemeden sonra en az yol uzunluğuna sahip ziyaret edilmeyen tepe noktasını seçeriz. Yani 7’den önce 5’i seçiyoruz. 

Tüm düğümler ziyaret edilene ve güncellenene kadar bu işleme devam ediyoruz.

Dijkstra Algoritmasının çalışma mantığı kısaca böyledir.

**Dijksta Algoritması Akış Diyagramı**

****

**Literatür Taraması**

Dijkstra'nın algoritması (adını keşfinden alan E.W. Dijkstra), bir grafikteki bir noktadan (kaynak) bir hedefe giden en kısa yolu bulma problemini çözer. Belli bir kaynaktan bir grafikteki tüm noktalara giden en kısa yolların aynı anda bulunabileceği ortaya çıktı, bu nedenle bu soruna bazen tek kaynaklı en kısa yol sorunu denir. Bu makale, basit ve anlaşılması kolay örnekler ve çizimler yardımıyla Dijkstra Algoritmasının altında yatan kavramları anlamanıza yardımcı olacaktır.

Gezgin satıcı problemi (GSP) fikri, belirli sayıda şehirde bir tur bulmak, ziyaret etmek her şehir tam olarak bir kez ve başlangıca dönüş . Bu turun uzunluğunun en aza indirildiği şehir. Gezgin satıcı probleminin ilk örneği 1759'da sorunu taşınmak olan Euler'dendi. bir satranç tahtasındaki her pozisyona tam olarak bir şövalye bir Zamanlar. Gezgin satıcı ilk kez bir kitapta ün kazandı

Alman satıcı BF Voigt tarafından 1832'de yazılmıştır. nasıl başarılı bir gezgin satıcı olunur. O bu isimle olmasa da GSP'den bahseder. mümkün olduğunca çok yeri kapsamasını önermek herhangi bir yeri iki kez ziyaret etmeden en çok bir tur planlamasının önemli bir yönü. Bu GSP'nin matematikteki kökenleri gerçekten kesin olarak bildiğimiz tek şey, bunun gerçekleştiği 1931 civarında.

Standart veya simetrik gezici satıcı problem matematiksel olarak şu şekilde ifade edilebilir: Ağırlıklı bir grafik verildiğinde G = (V,E) burada ağırlık i ve j düğümleri arasındaki kenarda cij negatif olmayan bir değerdir, minimum toplam maliyet. Şu anda garanti edilen bilinen tek yöntem gezgin satıcı problemini en iyi şekilde çözmek herhangi bir boyut, olası her turu numaralandırarak ve

en düşük maliyetle tur aranıyor. Her biri olası tur 123'ün bir permütasyonudur. . . n, n nerede şehir sayısı, dolayısıyla tur sayısı n!. n büyüdüğünde, imkansız hale gelir polinom zamanında her turun maliyetini bulur.

**Yazılım Mimarileri**

1. **Sunum Katmanı**

Bu bir projenin en üst düzeyidir. Ürünü inceleme, satın alma, alışveriş sepeti içeriği gibi hizmetler ile ilgili bilgiler görüntülenir.

Kısacası yazılımın, kullanıcıların doğrudan bir web sayfası gibi erişebileceği katmandır.

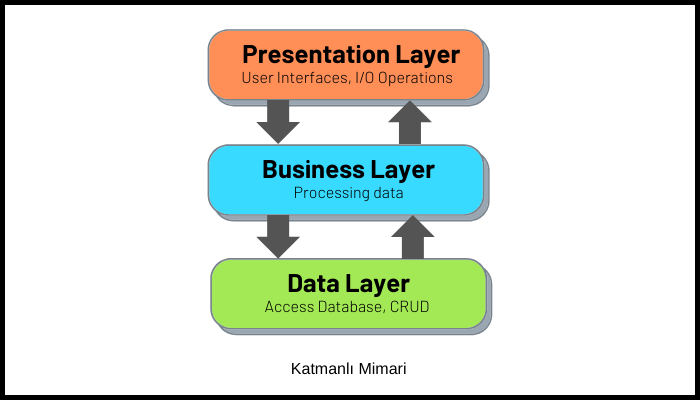
1. **İş Katmanı**

Diğer adıyla uygulama katmanıdır. Uygulamanın işlevselliğini kontrol eder. Sistemin çekirdek işlevselliğini uygular ve ilgili işletme mantığını kapsar.

1. **Veri Katmanı**

Veritabanı sunucuları, dosya paylaşımları vb. kapsayan, verileri açığa vuran kısacası verilere erişimin sağlandığı katmandır. İyileştirilebilir, ölçeklendirilebilir ve sürdürülebilirlik karşılığında veri merkezli uygulamalar için oldukça iyi sonuçlar alabileceğiniz bir mimari katmandır.

Genel olarak bu 3 mimarinin görevini toplayarak anlatacak olursak veri katmanı tarafından veritabanı kullanılıp işlemler yapıldıktan sonra iş katmanına gönderilir ve burada veri manipülasyonları yapıldıktan sonra sunum katmanına gönderilir. Sunum katmanında ise sadece verilerin gösterilmesini sağlayan kodlar yazılmalıdır. Yani sunumda veritabanı ile ilgili hiçbir işlem olmamalıdır.

****

**Big O Notasyonu Nedir?**

**Zaman**; hepimiz için çok önemli bir kavramdır ve herkes zamanını verimli ve boş şeylerle çok kaybetmeden geçirmek ister. Buraada devreye Big-O giriyor ve bunun sayesinde i sahip olduğumuz algoritmanın performansını değerlendirebilmemizi, algoritmamızı diğer algoritmalarla kıyaslayabilmemizi sağlıyor. Big-O notasyonu bir algoritmayı analiz etmede kullanılan en temel araçlardan bir tanesidir.

Kısacası Big-O notasyonu **kodunuzun karmaşıklığını açıklar.**

Dijkstra'nın en kısa yol algoritması O(ElogV)'dir, burada: V köşe sayısıdır E toplam kenar sayısıdır.

Algoritmanın O(VElogV) ise, burada: V köşe sayısıdır E, tek bir düğüme eklenen maksimum kenar sayısıdır. E’yi N olarak yeniden adlandırıyoruz. Yani bir analiz O(ElogV) iken diğeri O(VnlogV). Her ikisi de doğrudur ve aslında E = O(VN).

Aradaki fark, ElogV'nin daha sıkı bir tahmin olmasıdır.

1. Using Binary Heap: O((V+E)logV)

2. Using Fibonacci heap: O((VlogV+E))

3. Using Binomial Heap: O((V+E)logV)

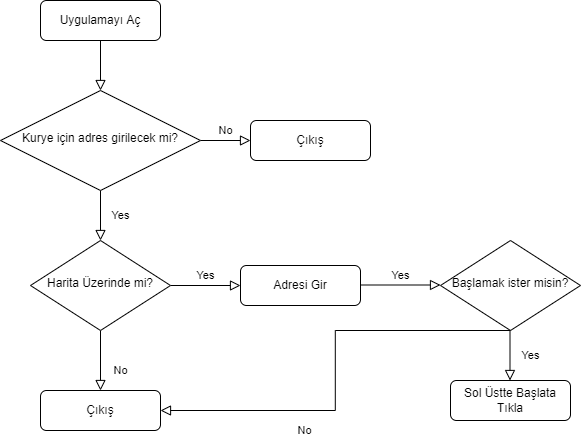
4. Using Array: O(V2+E)

**Djikistra Algoritması Big O Notasyon Hesabı**

* V : toplam köşe sayısı
* E : toplam kenar sayısı

İşlem **:** O(VElogV)

**-Blok Diyagramı-**



**13.Proje Grup Üyelerine Ne Kattı?**

**Gizem Coşkun**

Projede grup çalışmasının iyi olduğunu bir kere daha anlamış oldum. Takıldığım birçok noktada takım arkadaşlarımdan aldığım destek hatalarımı daha kolay çözmemi sağladı ve tüm bu süreç bana farklı bir bakış açısı kazandırdı. Diğer yandan böyle büyük bir projede çalışmak bana teknik anlamda çok fazla şey kattı. Daha önce diğer dillerde görüp anlamadığım bazı noktaların mantığını bu projede daha iyi anladım. Ayrıca Github kullanıyor olmamız proje kontrolünü çok kolaylaştırdı ve komut kullanımlarımızı pratikleştirdi. Gezgin satıcı problemi ve Djikistra hakkında bilgi sahibi oldum.

**Mert Bulut**

Proje c# Windows geliştirmeyi daha fazla tanımamı sağladı. Daha önce c# ile web projeleri geliştirdim fakat ilk defa Windows projesinde çalıştım. Ayrıca takım çalışmasının önemini bu projeyle birlikte çok iyi pekiştirdim. Projemizi git üzerinden takip ettiğimiz için birbirimize olan bağımlılığımız oldukça yüksekti. Merge olmamak için birbirimizin görevlerini sabırla bekledik. Takım çalışması diğer bütün isterlerden daha önemli bence. İyi bir takımla güzel bir proje çıkarttık. Fakat forma Google mapi gmap ile entegre etmek konusunda sıkıntı yaşadık. Entegre edince ise harita üstte kaldı ve işaretlerimiz gözükmedi.

**Feyyaz Can Köse**

Büyük bir projede klasör yapısının nasıl olması gerektiği, yapıların nasıl kurulması gerektiği takım çalışmasının ne denli önemli olduğunu anlamış oldum.

**Mehmet Ali Çelik**

Büyük bir projede çalışma tecrübesi edindim. Ekip olarak çalışmanın faydalarını gördüm. Github üzerinden aynı proje üzerinde birden fazla kişi çalışmayı öğrendim. Bir masaüstü uygulamasının nasıl geliştirildiğini öğrendim. Gezgin satıcı problemi hakkında araştırmalarımız sonucu bilgi sahibi oldum. Grup çalışmasının önemini anladım.

**Özgür Aslan**

Grup çalışmasının mantığını öğrenmiş oldum. Projenin Github'tan takibini yapmayı öğrendim. Projenin geliştirilme aşamasında uyulması gereken standartları öğrendim. Disiplin ve sorumluluk konusunda kendimi geliştirmiş oldum. Normalde belki de önem veya öncelik vermeyeceğim şeylere önem ve öncelik vermem gerekti çünkü grubumdaki diğer arkadaşlarıma karşı sorumluluğum vardı. Genel olarak faydalı ve güzel bir süreç geçirdim.

**KAYNAKÇA**

[1] https://bilgisayarkavramlari.com/2010/05/13/dijkstra-algoritmasi-2/

[2] https://dergipark.org.tr/tr/download/article-file/668518

[3]http://acikerisim.pau.edu.tr/xmlui/bitstream/handle/11499/1256/%C5%9Eahin%20Bayzan.pdf?sequence=1&isAllowed=y

[4]https://www.researchgate.net/publication/341371861\_Literature\_Review\_on\_Travelling\_Salesman\_Problem

[5]https://www.researchgate.net/publication/357219001\_Algorithms\_and\_Optimization\_Techniques\_for\_Solving\_TSP

[7] https://kutuphane.dogus.edu.tr/mvt/pdf.php

[8]https://www.researchgate.net/publication/273264449\_Understanding\_Dijkstra\_Algorithm

[9] https://erkancomez.com.tr/dijkstra-algoritmasi/

[10]<https://ijirt.org/master/publishedpaper/IJIRT101672_PAPER.pdf>

[11] https://www.scnsoft.com/application/desktop-app-development

[12] https://www.webtures.com/tr/blog/sozluk/masaustu-uygulamalari/

[13]https://www.technopat.net/sosyal/konu/masauestue-uygulama-gelistirmek.453497/

[14]https://stackoverflow.com/questions/26547816/understanding-time-complexity-calculation-for-dijkstra-algorithm