

BMB2006

VERİ YAPILARI

Doç. Dr. Murtaza CİCİOĞLU
Bursa Uludağ Üniversitesi
Bilgisayar Mühendisliği Bölümü

Ders Bilgileri

- Ders Günü ve Saati: Perşembe – 13:00
- İletişim: murtazacicioglu@uludag.edu.tr
- Dersin web sayfası : Microsoft Teams
- Dersin asist. : Fatih KAYA

Ders Bilgileri

- Değerlendirme:
- Ara Sınav (Vize) → %10
- Lab Uygulamaları -> %15
- Proje → %15
- Final Sınavı: %60

- Geç teslimler kabul edilmeyecektir.

Ders Bilgileri

- **Kaynaklar:**

- Veri Yapıları ve Algoritmalar, Dr. Rifat ÇÖLKESEN, Papatya yayıncılık,
- C ve Java ile Veri Yapılarına Giriş, Olcay Taner Yıldız, Boğaziçi Üniversitesi Yayınevi
- C/C++ İle Veri Yapıları Ve Çözümlü Uygulamalar, Muhammed Fatih Adak, Nejat Yumuşak, Seçkin Yayıncılık
- Data Structures and Problem Solving Using Java, Mark Allen Weiss, Pearson International Edition
- Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivers, Clifford Stein, The MIT Press
- internet

Dersin gereksinimleri

- Bu derste NTP dillerinden birisi (C#, Java, C++) ve yordamsal dillerden birisini (C, Pascal) bildiği varsayılmıştır.
- Bilinmesi gereken konular:
 - Temel veri türleri (int, float)
 - Kontrol yapısı (if else yapısı)
 - Döngüler
 - Fonksiyonlar(Methods)
 - Giriş çıkış işlemleri
 - Basit düzeyde diziler ve sınıflar

Dersin amacı

- Veri yapıları ve veri modellerini anlamak
- Algoritmaları incelemek, mevcut çözümleri anlamak
- Bilgisayar yazılım dünyasına ait bazı kavramlar, terimler ve sözcükleri anlamak
- Bağlı liste, ağaç, graf gibi çeşitli veri modellerini anlamak
- Programların bellek gereksinimleri ve çalışma hızlarını analiz edebilmek

İçerik

- Giriş
- Algoritma Analizi
- Bağlı Liste
- Stack (Yığıt, Çıkın, vb.)
- Queue (Kuyruk)
- Trees (Ağaçlar)
- Hash (Karma, Çırpı, vb.) Tabloları
- Heap (Yığın, Öbek)
- Set (Ayrık Küme)
- Graph (Graf, Çizge, vb.)

SORULAR???

Hafta 1: Giriş

Amaç:

- Veri yapılarına giriş



Yol haritası:

- Donanım ve Yazılım
- Bilgisayar Mimarisi
- İşletim Sistemleri
- Algoritma
- Veri Yapısı
- Veri Modeli
- Programlama Dilleri
- Kıyaslama

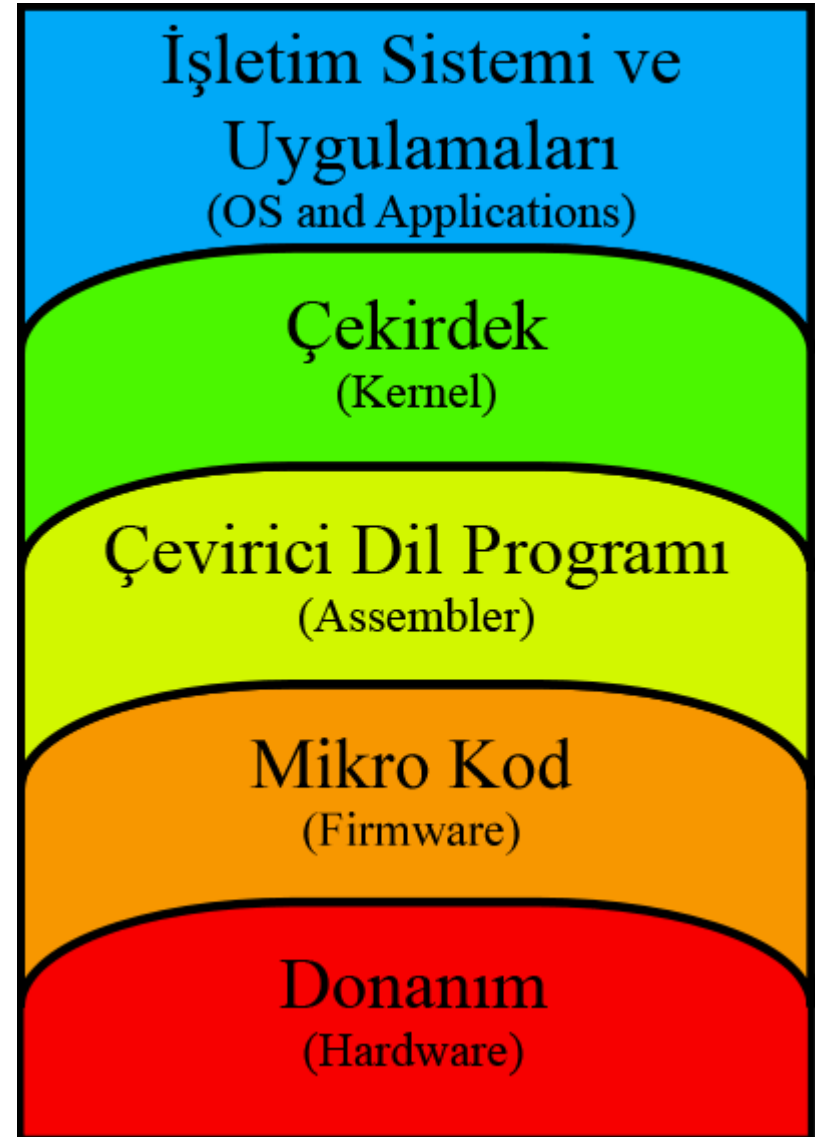
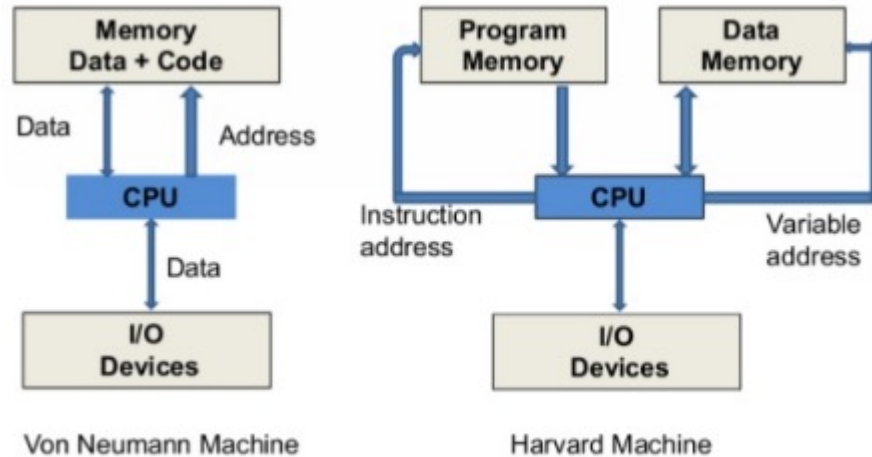
Donanım ve Yazılım

- Bilgisayarın oluşmasında, çalışmasında kullanılan somut parçaların tamamı
- Bilgisayar donanımının çalışmasını olanaklı hale getiren programların tamamı
- Program, bilgisayara verilen komutlar topluluğu olmakla birlikte, bu komutların, bir işi gerçekleştirme işlemidir.
- Yazılım programlar topluluğundan oluşmaktadır. Her bir programın yazılıma ait bir elemandır.



Bilgisayar Mimarisi

- Von Neumann
- Harvard



İşletim Sistemleri

- bilgisayarda çalışan, bilgisayar donanım kaynaklarını yöneten ve çeşitli uygulama yazılımları için yaygın servisleri sağlayan bir yazılımlar bütünüdür.
- sadece bilgisayar, video oyun konsolları, cep telefonları ve web sunucularında değil; arabalarda, beyaz eşyalarda hatta kol saatlerinin içinde bile yüklü olabilir.
- İşletim sistemleri işlevsellerinin genişliği ile değil, donanımı belli bir amaç doğrultusunda programlayabilme nitelikleriyle değerlendirilmelidir.



Algoritma

- **Algoritma**, belirli bir işi veya görevi var olan veya sonradan tanımlanan veri modeline dayandırılarak adım adım ortaya koymaktır.
- **Algoritma**, matematikte ve bilgisayar biliminde bir işi yapmak için tanımlanan, bir başlangıç durumundan başladığında, açıkça belirlenmiş bir son durumunda sonlanan, sonlu işlemler kümesidir.
- Başka bir deyişle, belli bir problemi çözmek veya belirli bir amaca ulaşmak için çizilen yola algoritma denir.
- **Verilen bir problemi çok farklı algoritmalar doğru bir şekilde çözebilir.**
- Fakat özel bir algoritmanın seçilmesi zaman ve hafıza kullanımında önemli ölçüde değişikliklere neden olabilir.



Algoritma Türleri

- Yinelemeli Algoritmalar
- Özyinelemeli (Böl & Yönet) Algoritmalar
- Rastgele Algoritmalar
- Açgözlü(Greedy) Algoritmalar
- Dinamik Programlama
- Yaklaşma Algoritmaları
- Genetik Algoritmalar

Veri Yapıları (Data Structures)

- **Veri** algoritmalar tarafından işlenen en temel elemanlardır (sayısal bilgiler, metinsel bilgiler, sesler ve girdi, çıktı olarak veya ara hesaplamalarda kullanılan diğer bilgiler...)
- Bir algoritmanın etkin, anlaşılır ve doğru olabilmesi için algoritmanın işleyeceği verilerin düzenlenmesi gerekir.
- **Veri yapısı** verinin veya bilginin bellekte tutulma şeklini veya düzenini gösterir.
- Tüm programlama dillerinin, Tamsayı (int), kesirli sayı (float), karakter (char) ve sözcük (string) saklanması için **temel (basit) veri yapıları vardır.**
- Programcı bu veri yapılarını, bunların bellekte nasıl saklandığı konusıyla ilgilenmeksizin bolca kullanır.
- Ayrıca, programcıda temel veri yapıları dışında yeni veri yapıları **(tanımlamalı, bileşik)** tanımlanması için programlama dillerinde bir çok özellik vardır. Örneğin: Struct, Class

Veri Yapıları (Data Structures)

- Temel ve tanımlamalı veri yapıları
- Temel veri yapıları, daha çok programlama dilleri tarafından doğrudan değişken veya sabit bildirimi yapılarak kullanılır.
- Tanımlamalı veri yapıları, kendisinden önceki tanımlamalı veya temel veri yapıları üzerine kurulurlar; yani, önceden geçerli olan veri yapıları kullanılarak sonradan tanımlanırlar.

```
int sayi=5;
char karakter='m';
float x = 4.156;
double y = 3.15;
bool kontrol=true;
char dizi[]="String karakter dizisi";
```

```
struct Dugum{
    int veri;
    struct Dugum *ileri;
    struct Dugum *geri;
};
typedef struct Dugum dugum;
typedef dugum* dugump;
```


Temel Veri Yapılar

- C dilinde temel veri tiplerine ait bilgiler

Tipi	Bit Boyutu	Tanım Aralığı
char	8	-127 – 127
unsigned char	8	0 – 255
signed char	8	-127 – 127
int	16 veya 32*	-32,767 – 32,767
unsigned int	16 veya 32*	0 – 65,535
signed int	16 veya 32*	-32,767 – 32,767
short int	16	-32,767 – 32,767
unsigned short int	16	0 – 65,535
signed short int	16	-32,767 – 32,767
long int	32	-2,147,483,647 – 2,147,483,647
signed long int	32	-2,147,483,647 – 2,147,483,647
unsigned long int	32	0 – 4,294,967,295
float	32	3.4×10^{-38} – 3.4×10^{38}
double	64	1.7×10^{-308} – 1.7×10^{308}

Tanımlamalı Veri Yapıları

- Tanımlamalı veri yapısı, temel veya daha önceden tanımlanmış veri yapılarının kullanılıp yeni veri yapıları oluşturulmasıdır. Üç değişik şekilde yapılabilir:
 - **Topluluk (Struct) Oluşturma**: Birden çok veri yapısının bir araya getirilip yeni bir veri yapısı ortaya çıkarmaktır. (C#, Java dillerinde sınıflar)
 - **Ortaklık (Union) Oluşturma**: Birden çok değişkenin aynı bellek alanını kullanmasını sağlayan veri yapısı tanımlamasıdır. Ortaklıkta en fazla yer işgal eden veri yapısı hangisi ise, ortaklık içerisindeki tüm değişkenler orayı paylaşır.
 - **Bit Düzeyinde Erişim**: Verinin her bir bit'i üzerinde diğerlerinden bağımsız olarak işlem yapılması olanağı sunar.
- Her birinin kullanım amacı farklı farklı olup uygulamaya göre bir tanesi veya hepsi bir arada kullanılabilir. Genel olarak, en çok kullanılanı topluluk oluşturmadır; böylece birden fazla veri yapısı bir araya getirilip/paketlenip yeni bir veri yapısı/türü ortaya çıkarılır.

Tanımlamalı Veri Yapıları

- C dilinde tanımlamalı veri yapılarına örnek aşağıda verilmiştir

```
struct kimlik {  
    char ad[15];  
    char soyad[20];  
    int yas;  
    char adres[50];  
};
```

```
union paylas {  
    int i;  
    flat f;  
    char kr;  
};_
```

her bit'e tek tek erişilebilir

dörtlü gruplar halinde erişilebilir

1 0 0 0 1 0 1 0

ikili gruplar halinde erişilebilir



Veri Yapılarına Neden İhtiyaç Duyulur?

- Bilgisayar yazılımları gün geçtikçe daha karmaşık bir hal almaktadır. *Google'un 55 milyardan fazla sayfanın indekslenmesi*
- Yazılımların programlanması ve yönetimi zorlaşmaktadır.
- Temiz kavramsal yapılar ve bu yapıları sunan çerçeve programları daha etkin ve daha doğru program yazmayı sağlar.
- İyi bir yazılım için;
 - Temiz bir tasarım
 - Kolay bakım ve yönetim
 - Güvenilir
 - Kolay kullanımlı
 - Hızlı algoritmalar
- Verimli veri yapıları
- Verimli algoritmalar

Veri Yapılarına Neden İhtiyaç Duyulur?

- 37 değerini dizide arayalım kaç adımda sonuç bulunur?

Binary search

steps: 0



Sequential search

steps: 0



Veri Yapılarına Neden İhtiyaç Duyulur?

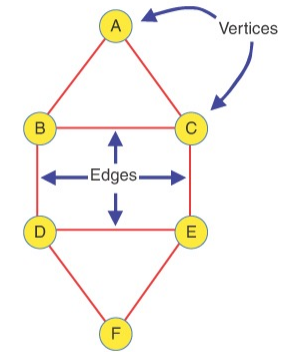
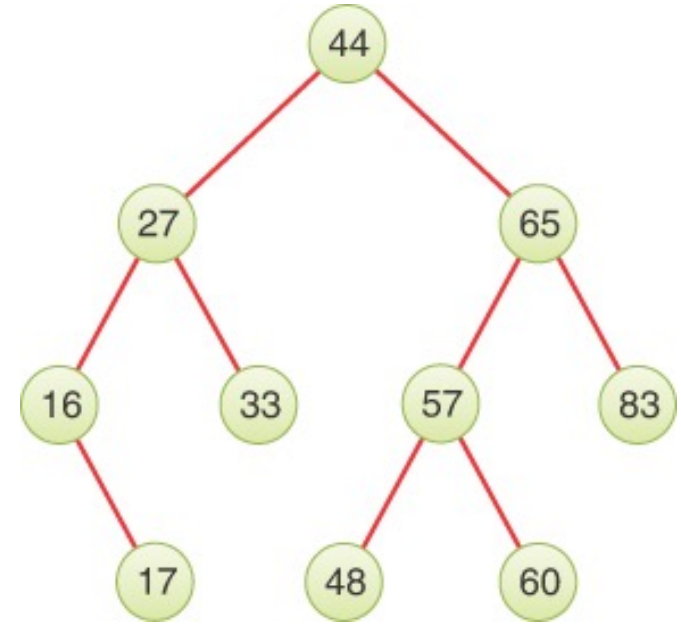
- Her bir satır başına ortalama 10 kelime ve yine ortalama 20 satırdan oluşan 3000 metin koleksiyonu olduğunu düşünelim.
 - 600,000 kelime
- Bu metinler içinde 'dünya' kelimesi ile eşleşecek tüm kelimeleri bulmak isteyelim.
- Doğru eşleştirme için yapılacak karşılaştırmanın 1sn. sürdüğünü varsayalım.
- Ne yapılmalıdır?
 - Sıralı eşleştirme: $1\text{sn} * 600,000 = 166 \text{ saat} ???$
 - İkili arama $\log_2 N = \log_2 600000$ yaklaşık 20 adım (çevrim) 20sn ???

Veri Modeli (Data Model)

- Veri modeli, verilerin birbirleriyle ilişkisel veya sırasal durumunu gösterir. Problemin çözümü için kavramsal bir yaklaşım yöntemi
- Bilgisayar ortamında uygulanacak tüm matematik ve mühendislik problemleri bir veri modeline yaklaştırılarak veya yeni veri modelleri tanımlaması yapılarak çözülebilmektedir.
- Tasarımı yapılacak programın **en uygun ve en etkin** şekilde olmasını sağlar ve daha baştan programın **çalışma hızı ve bellek gereksinimi** hakkında bilgi verir. Çoğu zaman programın çalışma hızıyla bellek gereksinimi miktarı doğru orantılıdır.

Veri Modeli (Data Model)

- Veri modelleri genel olarak;
 - Bağlantılı liste (Link List)
 - Ağaç (Tree)
 - Graf (Graph)
 - Durum Makinesi (State Machine)
 - Veritabanı-İlişkisel (Database Relational)
 - Ağ Bağlantı (Network Connection)
- Örneğin, veriniz üzerinde hızlı arama yapmak istiyorsanız bir ağaç modeli kullanabilirsiniz.
- Şehirler arasındaki mesafeleri hesaplayan bir yazılım geliştireceğiniz durumda ise graf modelini kullanabilirsiniz.



Liste ve Bağlantılı Liste Veri Modeli

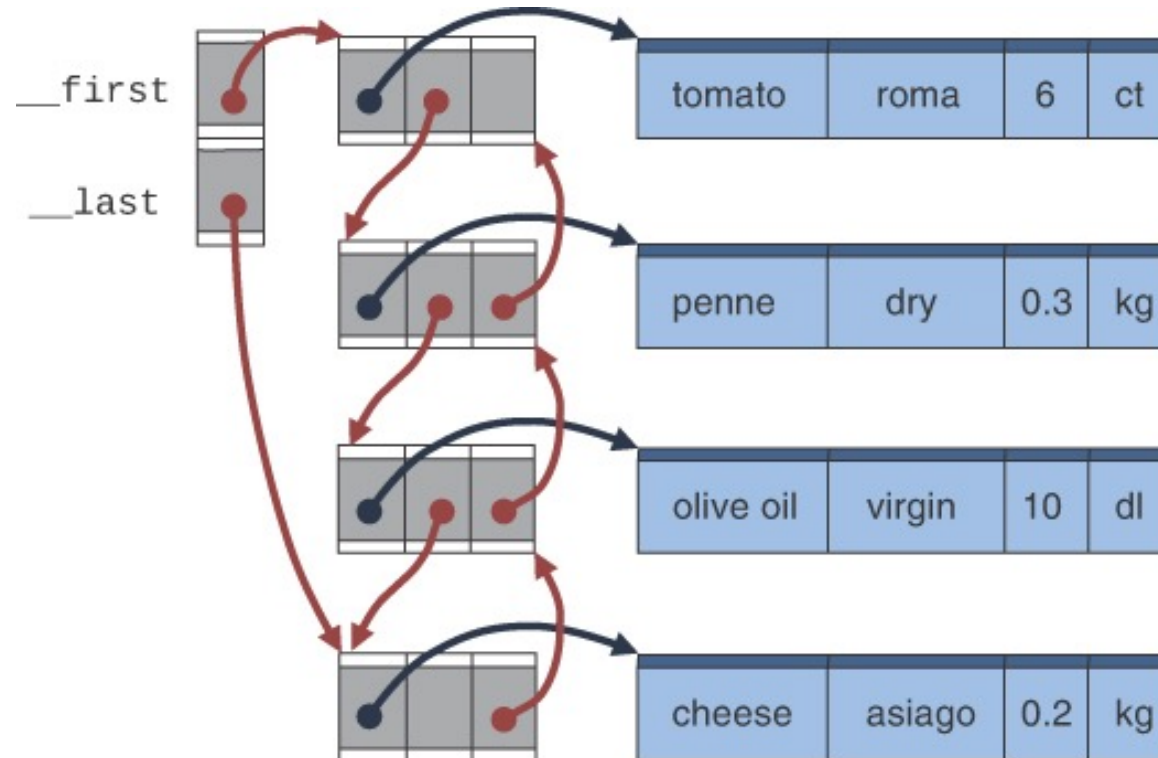
- **Liste veri modeli**, aynı kümeye ait olan verilerin bellekte art arda tutulması ilkesine dayanır. Veriler belirli bir düzen içerisinde (sıralı vs.) olabilir veya olmayabilir; önemli olan tüm verilerin art arda gelen sırada tutulmasıdır.
- En yalın liste veri modeli **bir boyutlu dizi** üzerinde tutulanıdır. Böylesi bir listeye eleman ekleme işlemi oldukça kolaydır; genel olarak, yeni gelen elemanlar listenin sonuna eklenir. Yalın listede bir sonraki eleman hemen o elemanın işgal ettiği bellek alanından sonradır.

5	→	dizi[0]
13	→	dizi[1]
45	→	dizi[2]
22	→	dizi[3]
-3	→	dizi[4]
8	→	dizi[5]
85	→	dizi[6]

Ardışıl Erişim
Dinamik Yaklaşım
Arama Maliyeti $O(n)$
Ekleme Maliyeti $O(1)$
Silme Maliyeti $O(1)$ veya $O(n)$
Esnek Bellek Kullanımı

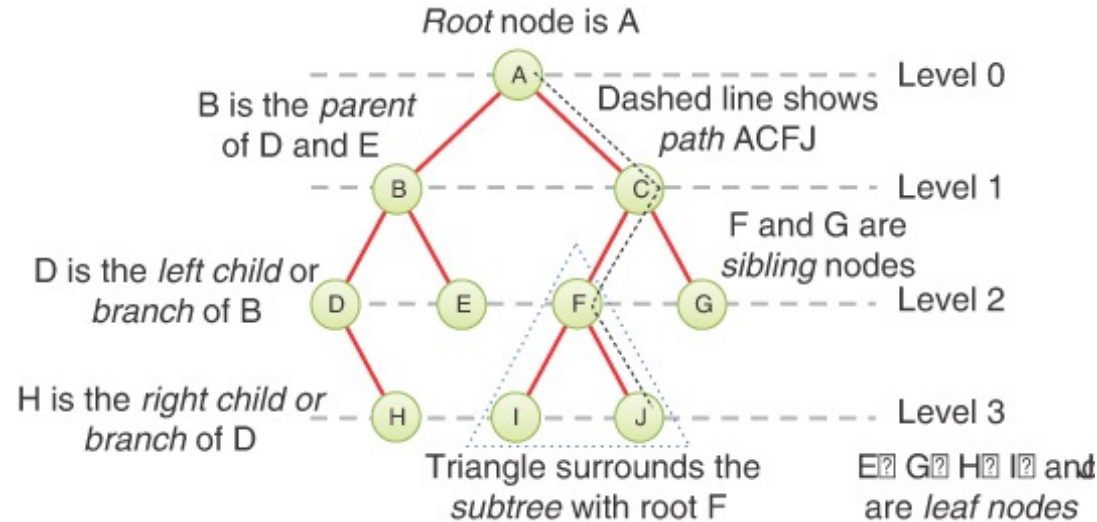
Liste ve Bağlantılı Liste Veri Modeli

- **Bağlantılı liste** (link list) ise, elemanların kendi değerlerine ek olarak bir de bağlantı bilgisinin kullanılmasıyla sağlanır; bağlantı bilgisi bir sonraki elemanın adresi niteliğindedir.



Ağaç Veri Modeli

- Ağaç veri modeli, düğümlerden ve dallardan oluşur; düğümlerde verilerin kendileri veya bir kısmı tutulurken, dallar diğer düğümlere olan bağlantı ilişkilerini gösterir. Ağaç veri modeli, özellikle kümenin büyük olduğu ve arama işleminin çok kullanıldığı uygulamalarda etkin bir çözüm sunar.
- En üstteki düğüm kök (root), kendisine alttan hiçbir bağlantının olmadığı düğüm yaprak (leaf), diğerleri de ara düğüm (internal node) olarak adlandırılır. Bir düğüme alttan bağlı düğümlere çocuk (child), üsten bağlı düğüme de o düğümün ailesi (parent) denilir.



Hızlı Erişim

Esnek Bellek Kullanımı

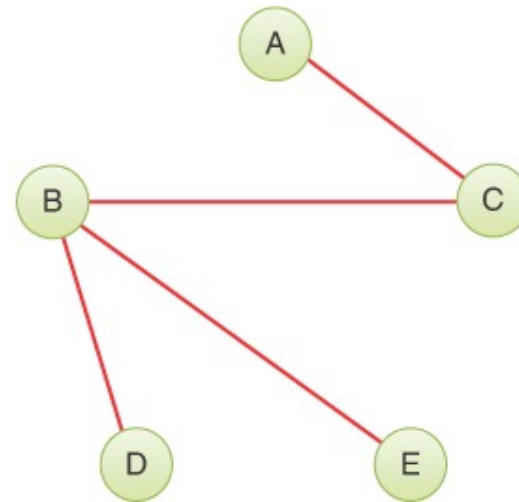
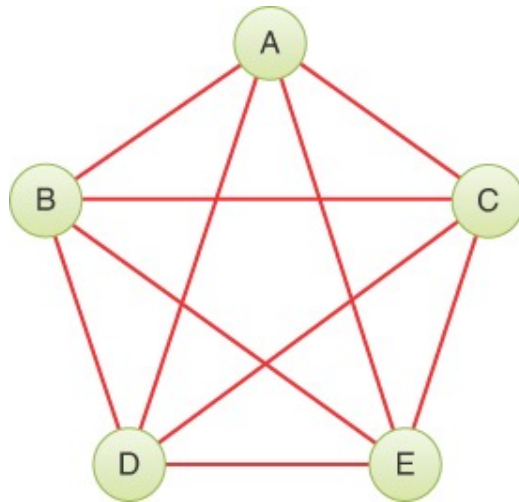
Arama-Ekleme Maliyetleri

Tasarım Esnekliği

Çok Farklı problemlere Model

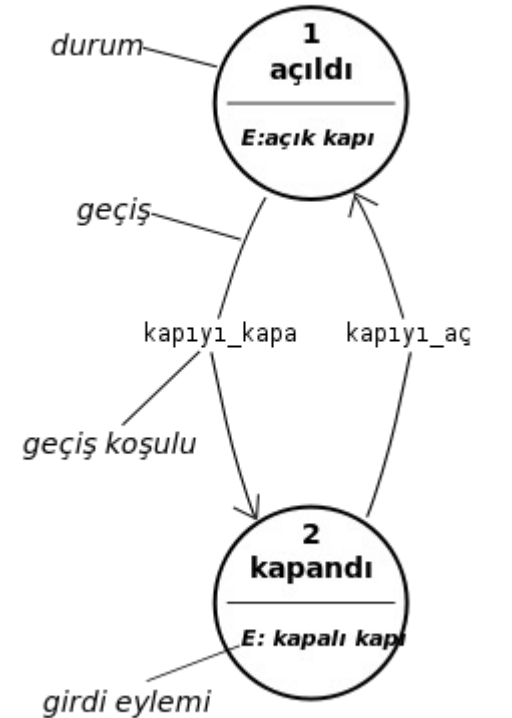
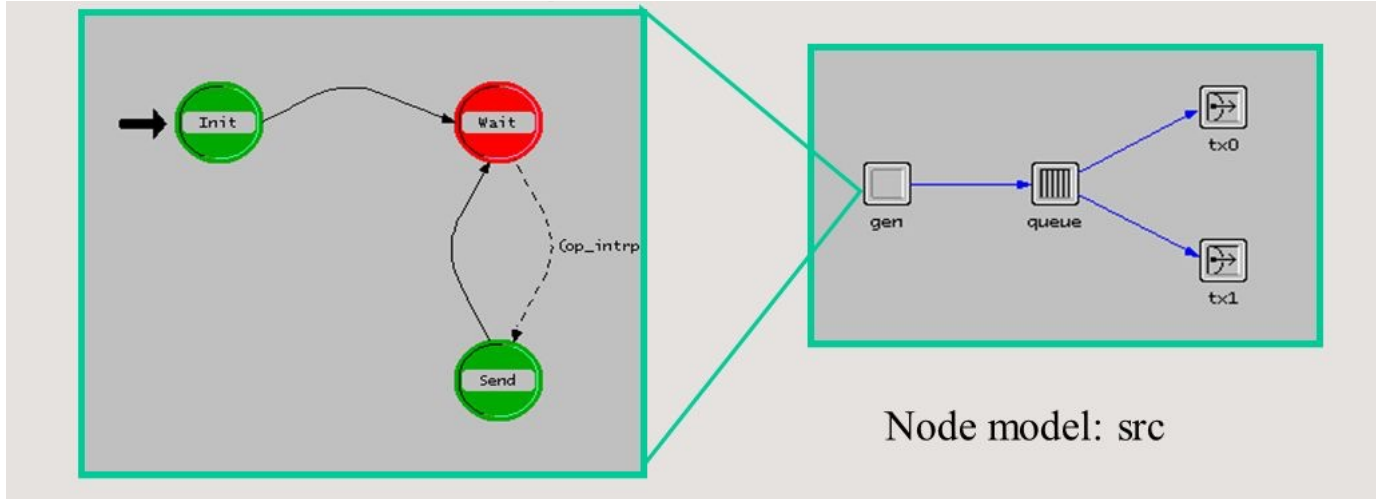
Graf Veri Modeli

- Graf veri modeli, aynı kümeye ait olan verilerin şekilde görüldüğü gibi düğümler, ayrıtlar (kenarlar) ve bunların birleştirilmesinden oluşur. Düğümler birleşme noktasını ayrıtlar da düğümlerin bağlantı ilişkisini gösterir. Verilerin kendileri veya bir kısmı hem düğümlerde hem de ayrıtların bilgi kısmında tutulabilir.
- Graflar, yazılım dünyasından önemli bir yere sahiptir. Örneğin, bir şehrin trafik altyapısından en yüksek akışın sağlanması, taşıma şirketinin en verimli taşıma şekli veya network bağlantılarında yüksek başarımla elde edilmesi gibi problemler



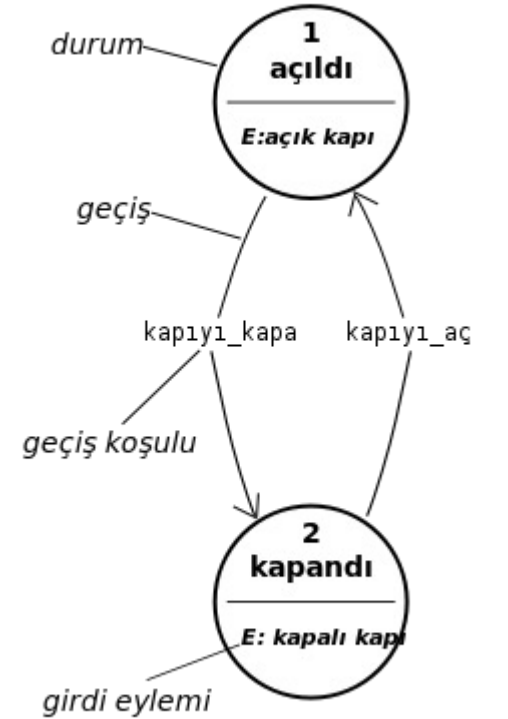
Durum Makinası Veri Modeli

- Durum makinası veri modeli, bir sistemin davranışını tanımlamak ve ortaya çıkarmak için kullanılan bir yaklaşım şeklidir; işletim sistemlerinde, derleyici ve yorumlayıcılarda, kontrol amaçlı yazılımlarda sistemin davranışını durumlara indirger ve durumlar arası geçiş koşullarıyla sistemi ortaya koyar.



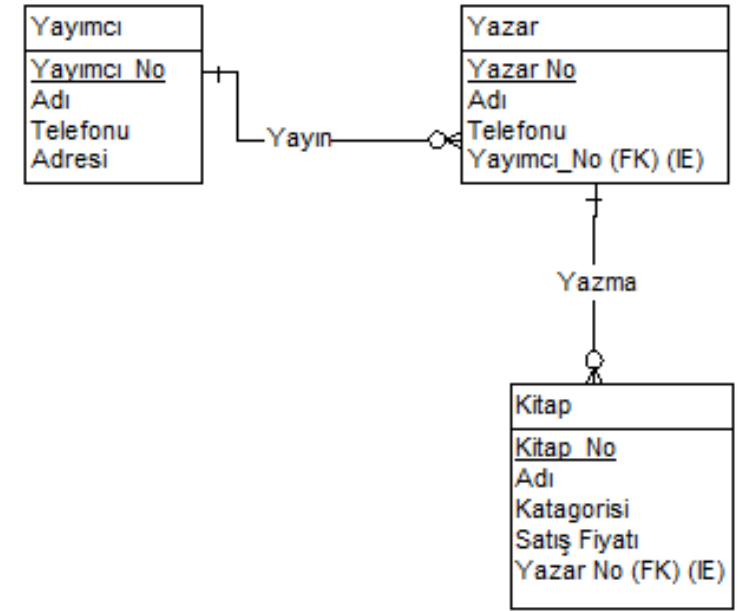
Durum Makinası Veri Modeli

- Durum makinası, yazılım uygulamasında birçok alanda kullanılabilir. Örneğin bir **robot kolunun hareketi, şifre çözme, gerçek zamanlı işletim sistemlerinde proses kontrolü** ve genel olarak kontrol alt sistemlerinin yazılımla uygulamayı başarılı bir şekilde sonuçlandırma durumlarında çözüm olur.
- Durum makinası veri modeli şeklen yönlü graflara benzer, ancak, birleşme noktaları graflarda olduğu gibi düğüm olarak değil de durum, ayrıtlar da geçiş eğrileri olarak adlandırılır. Durumlar arasındaki geçişler, sistemin o ana kadar ki durumlarına ve giriş parametrelerine bağlıdır.



Veritabanında İlişkisel Veri Modeli

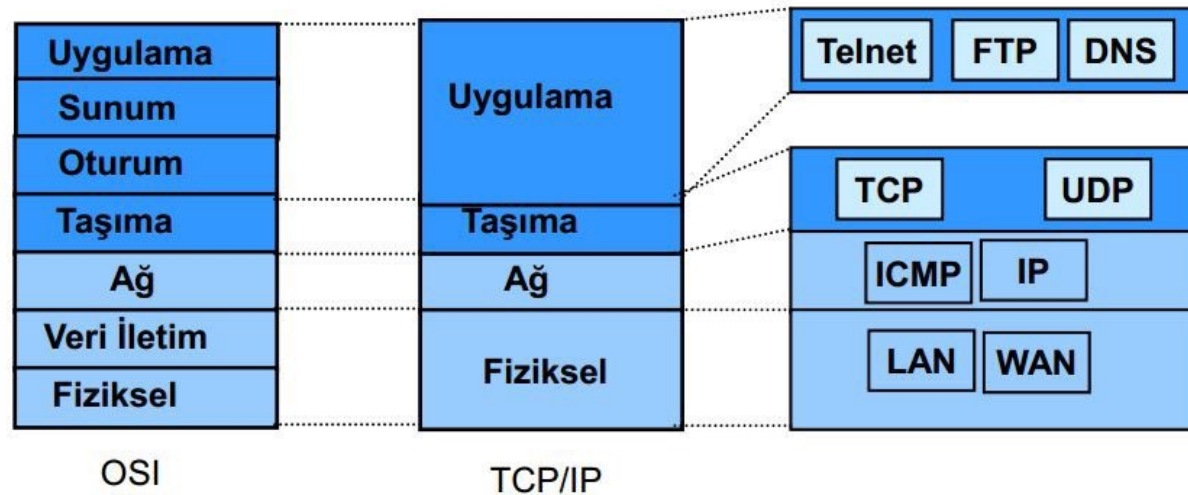
- Veritabanı ilişkisel veri modeli veritabanı uygulamalarında var olan dört beş sınıftan birisidir; veriler şekilde gösterildiği gibi tablolar üzerinden kurulan ilişkilere dayanmaktadır.
- SQL (Structured Query Language), sorgulama dili kullanılarak veritabanı üzerinde sorgulama yapılabilir. Access, Microsoft SQL, ORACLE, SYBASE, Ingres gibi birçok veritabanı yönetim sistemleri ilişkisel veri modelini desteklemektedir.
- Veritabanı yönetim sistemleri, veritabanı oluşturma, tablo yaratma, alanları tanımlama gibi işlerin başarılı bir şekilde sonuçlandırmasını ve genel olarak veritabanı yönetimini sağlarlar.



Relational Data Model

Ağ Veri Modeli

- Ağ veri modeli, katmalı ağ mimarilerinde, bilgisayarlar arasında eş katmanlar (peer layers) düzeyinde veri alış-verişini sağlayan dilim (segment), paket (packet) ve çerçeve yapılarını ortaya koyar ve iletişim için gerekli davranışı tanımlar. Veri haberleşmesinde hemen hemen tüm mimariler katmanlı yapıdadır. Tüm mimariler örnek temsil eden OSI başvuru modeli 7, TCP/IP (Transmission Control Protocol / Internet Protocol) protokol kümesi 4 katmanlıdır.























Veri Modelleri

- **Liste** : Sonlu sayıda elemandan oluşan ve elemanları doğrusal sırada yerleştirilmiş veri modeli. Herhangi bir elemanına erişimde sınırlama yoktur.
- **Yığıt (Stack)**: Elemanlarına erişim sınırlaması olan, liste uyarlı veri modeli (Last In First Out-LIFO listesi).
- **Kuyruk (Queue)**: Elemanlarına erişim sınırlaması olan, liste uyarlı veri modeli. (First In First Out-FIFO listesi).
- **Ağaç (Tree)** : Doğrusal olmayan belirli niteliklere sahip veri modeli
- **Çizge (Graph)** : Köşe adı verilen düğümleri ve kenar adı verilen köşeleri birbirine bağlayan bağlantılardan oluşan doğrusal olmayan veri modeli

Veri Yapısı	Artıları	Eksileri
Dizi	Hızlı ekleme ve çok hızlı erişim (indis biliniyorsa)	Yavaş arama, yavaş silme ve sabit boyut.
Sıralı Dizi	Sıralanmamış diziye göre daha hızlı arama.	Yavaş arama, yavaş silme ve sabit boyut.
Yığın (Stack)	Son giren, ilk çıkar (last-in, first-out) erişimi sağlar.	Diğer öğelere yavaş erişim.
Kuyruk	İlk giren, ilk çıkar (first-in, first-out) erişimi sağlar.	Diğer öğelere yavaş erişim.
Bağlı Liste	Hızlı ekleme ve silme.	Yavaş arama.
Hash Tablosu	Hızlı ekleme ve anahtar bilindiğinde çok hızlı erişim.	Yavaş silme, anahtar bilinmediğinde yavaş erişim ve verimsiz bellek kullanımı.
Yığın (Heap)	Hızlı ekleme ve silme	Diğer öğelere yavaş erişim. Başta en büyük öğeye erişim.
İkili Ağaç	Hızlı arama, ekleme ve silme(ağaç dengeli kalmışsa).	Silme algoritması karmaşık.
Graf	Gerçek-dünya problemlerini modelleyebilmesi.	Bazı algoritmaları yavaş çalışmakta ve karmaşıklığı yüksek.

Programlama Dilleri

- Programlama dili, yazılımcının bir algoritmayı ifade etmek amacıyla, bir bilgisayara ne yapmasını istediğini anlatmasının bir yoludur.
- Programlama dilleri, yazılımcının bilgisayara hangi veri üzerinde işlem yapacağını, verinin nasıl depolanıp iletileceğini, hangi koşullarda hangi işlemlerin yapılacağını tam olarak anlatmasını sağlar.
- Makine kodu, Assembly, C, C#, Java, Python, vb.

1	Java		11	MATLAB	
2	C		12	R	
3	Python		13	Perl	
4	C++		14	Assembly Language	
5	Visual Basic .NET		15	Swift	
6	Javascript		16	Go	
7	C#		17	Delphi/Object Pascal	
8	PHP		18	Ruby	
9	SQL		19	PL/SQL	
10	Objective-C		20	Visual Basic	

Kıyaslama (Benchmarking)

- Kıyaslama, aynı işi yapan iki programın veya yazılımın evrensel anlamda örnek veriler üzerinde çalıştırılarak başarımlarını karşılaştırmaktır; kıyaslama donanım için de yapılabilir.
- Bir yazılımda kıyaslama kriterleri:
 - Kullandığı CPU miktarı
 - Kullandığı RAM miktarı
 - Kullandığı Disk miktarı
 - Yanıt verme süresi
 - Aynı anda yapabileceği maksimum işlem sayısı
 - Ağın band genişliğine etkisi



Bölüm Özeti

- Veri modelleri ve onlara ait veri yapıları yazılım geliştirmenin temel noktalarındır; problemlerin en etkin şekilde çözülebilmesi için ona algoritmik ifadenin doğasına yakın bulunmasıdır. Kısaca, veri yapıları, verinin saklanma kalıbını, veri modelleri de veriler arasındaki ilişkiyi gösterir.
- Bilinen ve çözümlerde sıkça başvurulan veri modelleri, genel olarak, bağlantılı liste (link list), ağaç (tree), graf (graph), durum makinası (state machine), ağ (network) ve veritabanı-ilişkisel (database-relation) şeklinde verilebilir.
- Her veri modelinin, altında duran veri yapısına bağlı olarak, işlem zaman maliyetleri ve bellek gereksinimleri farklıdır. Program geliştirilirken, zaman ve bellek alanı maliyetlerini dengeleyecek çözüm üretilmeye çalışılır. Genel olarak, RAM türü ardışıl erişimlerin yapılabildiği bellek üzerinde, maliyeti ile bellek gereksinim ters orantılı olduğu söylenebilir.