Bilgisayar Mühendisliğine Giriş

Yrd.Doç.Dr.Hacer KARACAN

VERİ YAPILARI VE VERİ MODELLERİ

- Veri, Yapı ve Algoritma Tanımları
- Veri Yapısı ve Bilgi
- Temel Veri Yapıları

Tanımlar

- **Algoritma :** Bir problemin çözümünde kullanılan komutlar dizisi.
- Bir problemi çözmek için geliştirilmiş kesin bir yöntemdir.
- Bir algoritma, bir programlama dilinde (C++, C, Java, Pascal gibi) ifade edildiğinde **program** adını alır.

Tanımlar

- **Veri :** Algoritmalar tarafından işlenen en temel elemanlar (sayısal bilgiler, metinsel bilgiler, resimler, sesler ve girdi, çıktı olarak veya ara hesaplamalarda kullanılan diğer bilgiler ...).

- Bir algoritmanın etkin, anlaşılır ve doğru olabilmesi için, algoritmanın işleyeceği verilerin düzenlenmesi gerekir.

Tanımlar

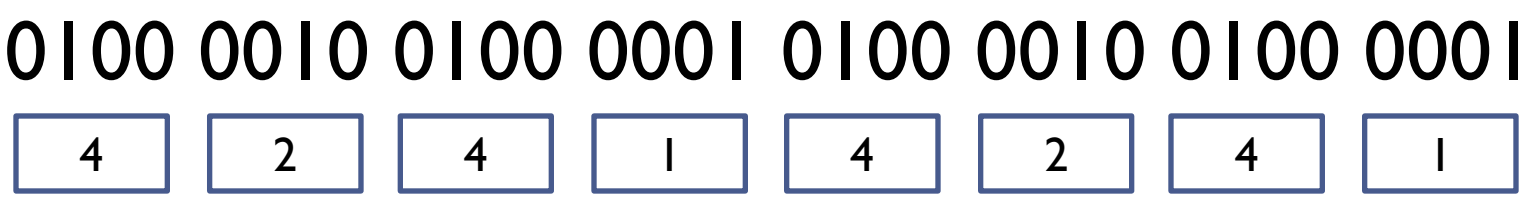
- **Veri Yapıları :** Verilerin düzenlenme biçimini belirleyen yapıtaşlarıdır.
- Karakter, Tamsayı ve Gerçel Sayı gibi değişkenler temel veri yapısı olarak kabul edilir.
- Karakterler bir araya gelerek sözceleri (string), sayılar bir araya gelerek dizileri (array) oluşturur.
- Değişik algoritmalarda verilerin diziler, yığıtlar(stack), kuyruklar(queue), ağaçlar(tree) ve çizgeler(graph) gibi veri yapıları şeklinde düzenlenmesi gerekebilir.

Veri, Yapı ve Algoritma

- Bir programda veri, yapı ve algoritma birbirinden ayrılmaz bileşenlerdir ve her biri önemlidir.

- Verilerin düzenlenme biçimleri önemlidir.
  - Çünkü, yapı iyi tasarlandığında, etkin, doğru, anlaşılır ve hızlı çalışıp az kaynak kullanan algoritma geliştirmek kolaylaşır.

Veri Yapısı ve Bilgi



- Yukarıdaki bit dizisi;
  - Karakter dizisi (string) ise (ASCII): B A B A
  - BCD (Binary Coded Decimal) ise:4 2 4 1 4 2 4 1
  - 16-bit tam sayı ise: 16961 16961
  - 32-bit tam sayı ise: 1111573057
  - 32-bit gerçel sayı ise: 0.4276801×10^{66}

Temel Veri Yapıları

- **Karakterler**
  - ASCII Her karakter 8 bit (2^8 = 256 farklı karakter)
  - Unicode Her karakter 16 bit (2^{16} = 65536 farklı karakter)

- **Tamsayılar**
  - 8 bit (-128 ' den 127 ' ye)
  - 16 bit (-32768 ' den 32767 ' ye)
  - 32 bit (-2147483648 ' den 2147483647 ' ye)
  - 64 bit (-9,223,372,036,854,775,808'den 9,223,372,036,854,775,807' ye)

- **Ondalıklı (Gerçel) Sayılar**
  - 16 bit half (IEEE 754-2008)
  - 32 bit single, float (C)
  - 64 bit double, real (Pascal)
  - 128 bit quad

Tamsayı Formatları

Sayı:	**39**
Doğal ikili sayı	100111
1'e Tümleyen	011000
2'ye Tümleyen	011001
BCD Kodlamalı	00111001

Diziler

- **String:** Karakter dizileri (Sözce)
  - Karakter sayısının tutulması (PASCAL)
    - [10, b, i, l, g, i, s, a, y, a, r]
  - Sonlandırma karakterinin (\0) kullanılması (C)
    - [b, i, l, g, i, s, a, y, a, r, \0]

- **Array:** aynı kümeye ait verilerin bellekte art arda tutulması
  - Tek boyutlu(dizi), İki boyutlu (matris)

$$\begin{bmatrix} a & b & c \end{bmatrix} \qquad \begin{bmatrix} 2 & -3 & -1 \\ 5 & 2 & 4 \\ 0 & 3 & -2 \end{bmatrix}$$

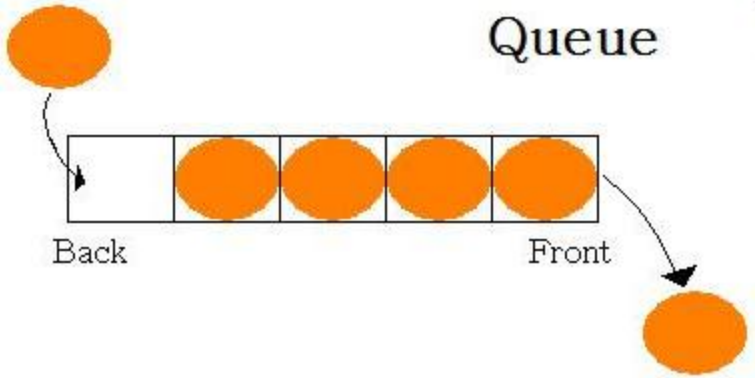
Soyut Veri Yapıları

- Bir grup veriyi ve bu veriler üzerinde yapılabilecek tüm işlemleri bir araya getiren yapıya **soyut veri yapısı** (abstract data type: ADT) denir.

- Kullanıcı için yapının içinin tamamen soyut olması (bilinmesinin gerekmemesi) nedeniyle soyut veri yapısı adını almıştır.

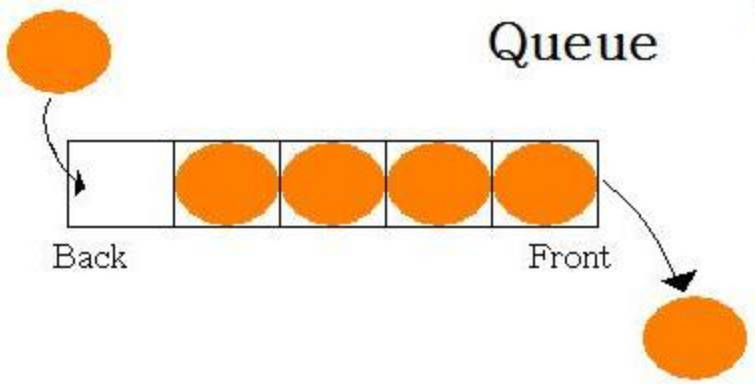
- En çok kullanılan soyut veri tipleri:
  - kuyruk
  - yığın
  - bağlı liste
  - ağaç

Kuyruk (Queue)



- Kuyruklar, eleman eklemelerin sondan (back) ve eleman çıkarmaların baştan (front) yapıldığı veri yapılarıdır.
- Bir eleman ekleneceği zaman kuyruğun sonuna eklenir.
- Bir eleman çıkarılacağı zaman kuyrukta bulunan ilk eleman çıkarılır.
- Bu eleman da kuyruktaki elemanlar içinde ilk eklenen elemandır.
- Bu nedenle kuyruklara FIFO (First-In First-Out = ilk giren ilk çıkar) listeleri de denilmektedir.

Kuyruk (Queue)



- Gerçek yaşamda da bankalarda, duraklarda, gişelerde, süpermarketlerde, otoyollarda kuyruklar oluşmaktadır.
- Kuyruğa ilk olarak girenler işlemlerini ilk olarak tamamlayıp kuyruktan çıkarlar.
- Veri yapılarındaki kuyruklar bu tür veri yapılarının simülasyonunda kullanılmaktadır.
- Ayrıca işlemci, yazıcı, disk gibi kaynaklar üzerindeki işlemlerin yürütülmesinde ve bilgisayar ağlarında paketlerin yönlendirilmesinde de kuyruklardan yararlanılmaktadır.

Yığın (Stack)

- Eleman ekleme çıkarmaların en üstten (top) yapıldığı veri yapısına yığın (stack) adı verilir.
- Bir eleman ekleneceğinde yığının en üstüne konulur.
- Bir eleman çıkarılacağı zaman yığının en üstündeki eleman çıkarılır.
- Bu eleman da yığındaki elemanlar içindeki en son eklenen elemandır.
- Bu nedenle yığınlara LIFO (Last-In First-Out : son giren ilk çıkar) listeleri de denilir.

Yığın (Stack)

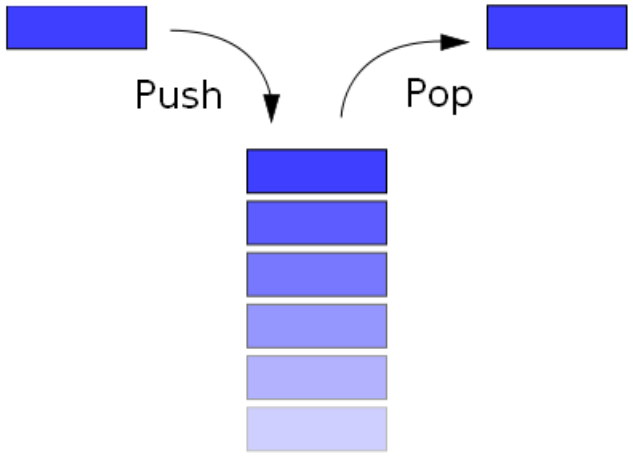
- Yığın yapısına gerçek yaşamdan örnek verirsek: üst üste konulan eşyaları taşımak için en üste konulan eşyayı (en son konulmuş olanı) ilk olarak almamız gerekir.

- Bir feribotun hem önünde hem arkasında araç indirme/bindirme kapısı varsa, o feribot FIFO düzeninde, sadece 1 kapısı varsa LIFO düzeninde araç indirip/bindirir.

- Bir web tarayıcısında önceki sayfalara dönmek ve bir uygulamada en son yapılan işlemleri geri almak gibi işlerde yığın yapısı kullanılabilir.

Yığın İşlemleri ve Tanımları

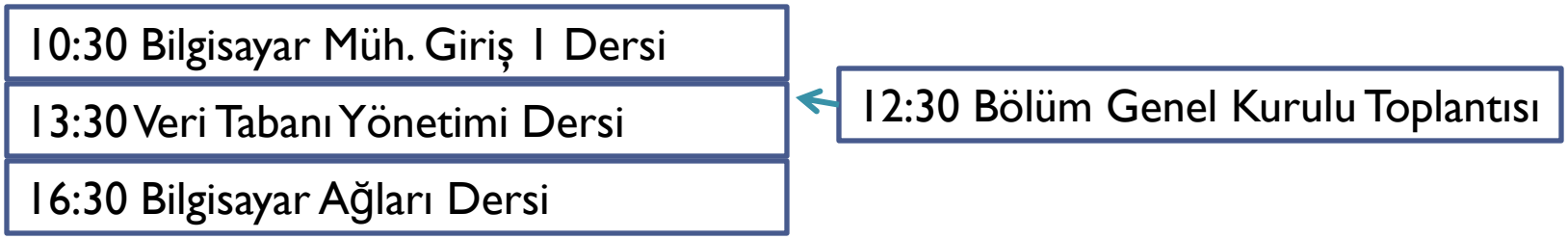
- **push(s,i) :** s yığınının en üstüne i değerini eleman olarak ekler.
- **i = pop(s) :** s yığınının en üstündeki elemanı çıkartır ve değerini i değişkenine atar.
- **stacktop :** (yığıntan çıkarılmaksızın en üstteki elemanın değerini döndüren işlem, diğer adı **peek**)



Liste (List)

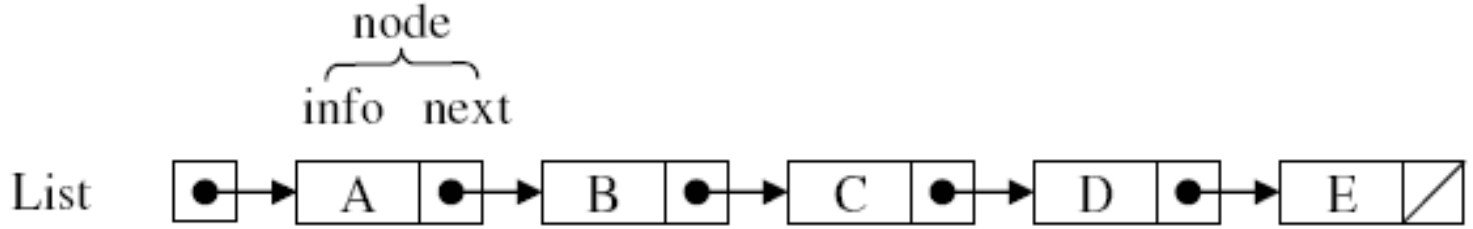
- Eleman ekleme ve çıkarma işlemlerinin herhangi bir sınırlama olmaksızın istenilen yerden yapılabildiği veri yapısıdır.

- Örneğin daha önce oluşturduğumuz bir **bugün yapılacak işler** listesine bir eleman eklerken, her zaman en sona veya en başa değil araya eleman eklememiz de gerekebilir.



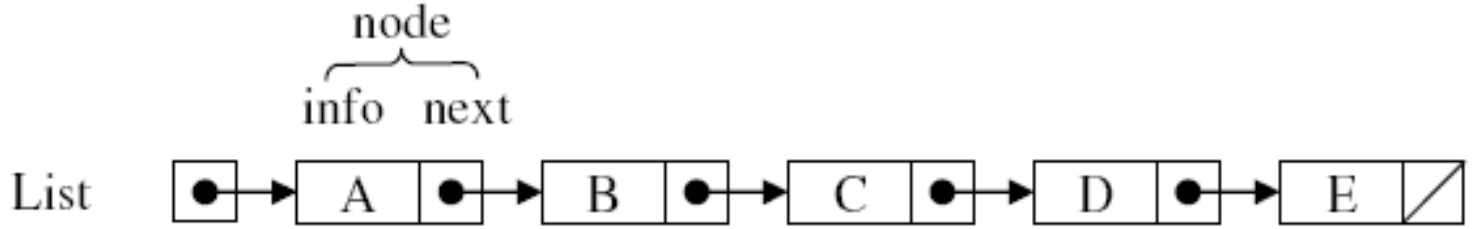
Bağlı Liste (Linked List)

- Kuyruk ve Yığın veri yapılarını diziler ile gerçekleştirmek mümkün olsa da, liste yapısını gerçekleştirmek için dizi çok uygun değildir.
  - Araya eleman eklenmesi/çıkartılması gerektiğinde, o elemandan sonra gelen tüm elemanların birer kademe ileri/geri kaydırılması gereklidir.
- Bağlı liste yapısı, listedeki sıralamayı bir bağ ile göstererek bu gerekliliği ortadan kaldırmıştır.



Bağlı Liste (Linked List)

- Listenin her bir elemanına düğüm (node) adı verilir.
- Düğümler, bilgi ve bağ (adres) alanlarından oluşmaktadırlar.
- Bağ alanında bir sonraki düğümün adresi genellikle bir işaretçi (pointer) ile saklanır.
- Eğer bilgi alanında kimlik no, ad, soyad gibi birden çok veri bulunuyorsa ve bu alanlardan birkaç tanesine göre sıralama bilgisi tutulması gerekliyse, birden çok bağ alanı kullanılabilir.



Bağlı Liste Kullanmanın Avantajları

- Yığın ve kuyrukların gerçekleştirilmesinde dizi kullanmanın dezavantajları:
  - hiç kullanılmasa veya az kullanılsa bile sabit miktardaki belleğin bu yapılara ayrılması gerekir
  - sabit bellek dolduğunda eleman eklenemez

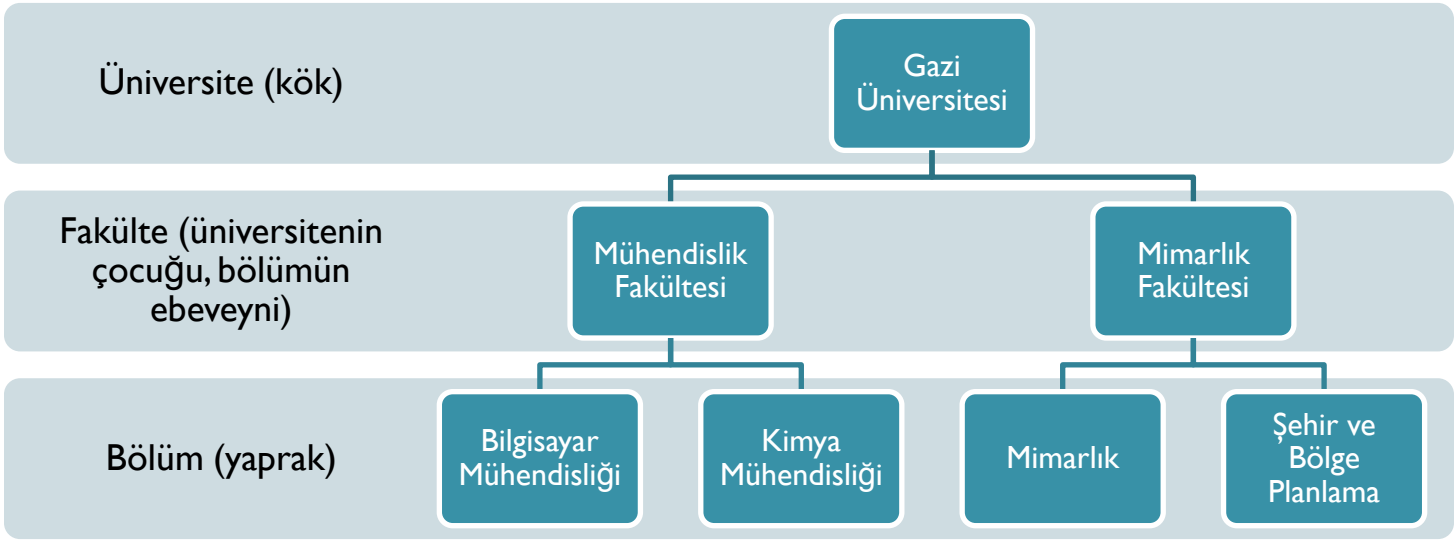
- Bağlı listeler kullanılırsa bu problemler ortadan kalkmaktadır:
  - Bellekten sabit miktarda bir yer ayrılmaz, ana bellek dolana kadar bu yapılara ekleme işlemi yapılabilir

Ağaç (Tree)

- Ağaç yapıları sıradüzensel (hiyerarşik) bir yapıyı gerçekleştirmek için kullanılır.
- Ağacın her bir elemanına da listede olduğu gibi düğüm (node) denir.
- En üstteki elemana kök düğüm (root node), en uçtaki elemanlara ise yaprak düğüm (leaf node) denir.
- Bir düğümü işaret eden (üst seviyedeki) düğüme ebeveyn düğüm (parent node), bir düğümün işaret ettiği (alt seviyedeki) düğümlere çocuk düğüm (child node) denir.

İkili Ağaç (Binary Tree)

- Eğer bir ağaç yapısında her düğümün sadece iki çocuk düğümü olabiliyorsa ikili ağaç, ikiden çok çocuk düğümü olabiliyorsa çoklu ağaç denir.



* Gerçekte üniversite, fakülte ve bölüm bilgilerini çoklu ağaçlarda saklamak gerekir.

Ağaç (Tree)

- Arama ve sıralama işlemleri için kullanılan İkili Arama Ağacı (BST: Binary Search Tree) gibi özel ağaç türleri de vardır.

- Ağaç yapıları ikili veya çoklu bağlı listeler ile gerçekleştirilebilir.