Veri Yapıları ve Programlama

MARDİN ARTUKLU ÜNİVERSİTESİ

HAZIRLAYAN: Öğr.Gör. Hüseyin AHMETOĞLU

1.HAFTA

Veri Yapılarına Giriş

BÖLÜM - 1

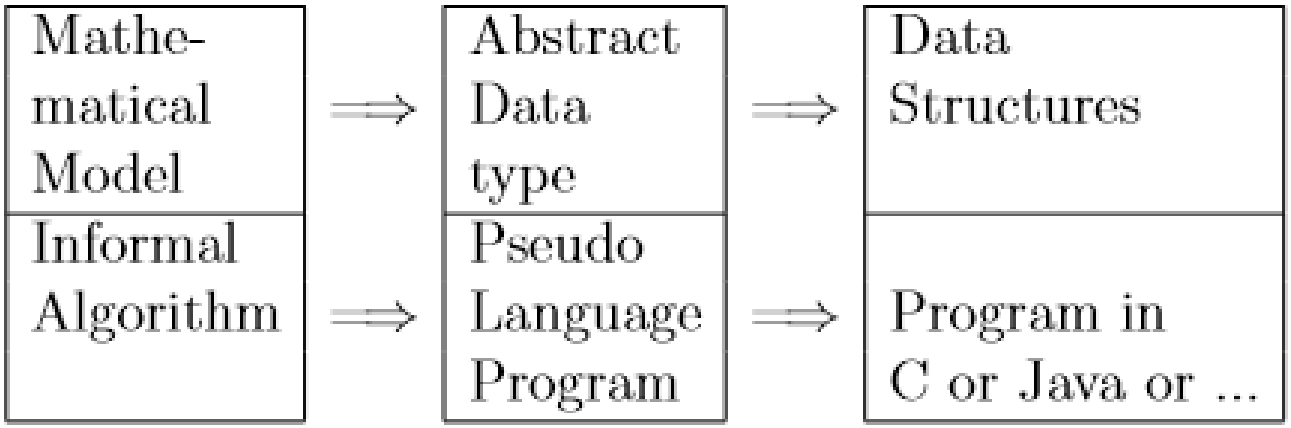
Bu bölümde,

- Algoritmalar ve Veri Yapıları,
- Lineer ve Non-Lineer Veri Yapıları,
- Statik ve Dinamik Veri Yapıları,
- Veri Yapılarının ADT ile Modellenmesi

konularına değinilecektir.

Veri Yapıları ve Algoritmalar

- **Bilgisayar bilimleri**ndeki klasik problem çözme süreci aşağıdaki gibidir:

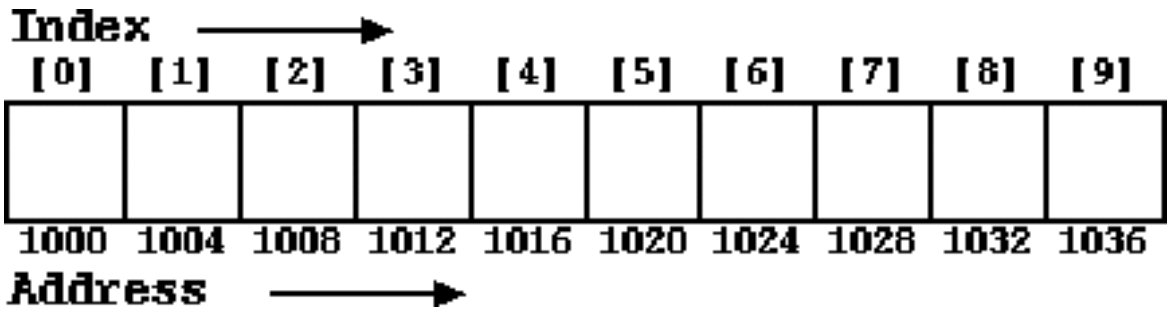


Veri Yapıları ve Algoritmalar (devam…)

- Veri yapıları ve Algoritmalar, ***"Bilgisayar Bilimleri"*** alanındaki en köklü ve klasik konulardan birisidir.
- Tüm bilgisayar programları Algoritma ve **Veri Yapılarından** oluşur.
- Algoritmalar **veri** ve değişkenleri kullanarak bir problemin çözümüne **ulaşır**.
- Bir **değişken** bir *integer* veya *char* türünde **bellekte** bir *hafızalık* bir yer tutarken, bir Veri Yapısı *algoritmanın kullanacağı* bilgiyi bir **grup hafızada** tutar.

Veri Yapıları ve Algoritmalar (devam…)

- Örneğin, Array (**dizi**) bir veri yapısı bir **isme sahip *aynı türde*** verilerin sabit uzunluklu bir *kümesidir*.
- Bu bağlamda, bir Array için *programın çalışması süresince* sabit uzunluklu bir hafıza ayrılır. Bu yönüyle Array ***statik*** bir **Veri Yapısıdır**.

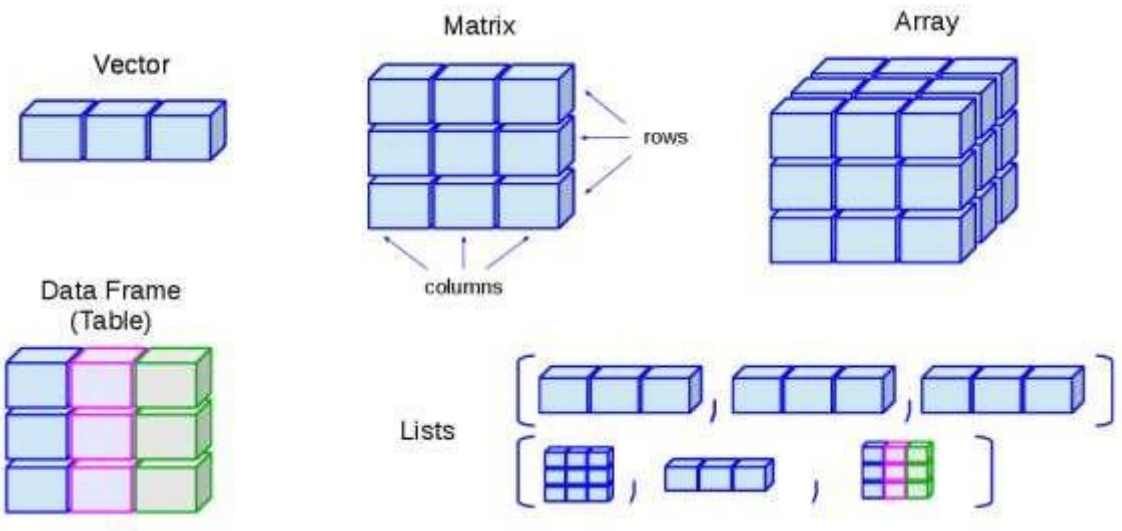


Veri Yapısı Nedir?

- Bilgisayar bilimlerinde **işlenecek** veri miktarı büyüdükçe bu verilerin
  - saklanması,
  - sıralanması,
  - veriler içinde arama yapılması
- gibi temel işlemlerin ***verimli şekilde*** (hafıza, performans) gerçekleştirilebilmesi için farklı *Veri Yapılarının* geliştirilmesi **ihtiyacı doğmuştur**.

Veri Yapısı Nedir? (devam…)

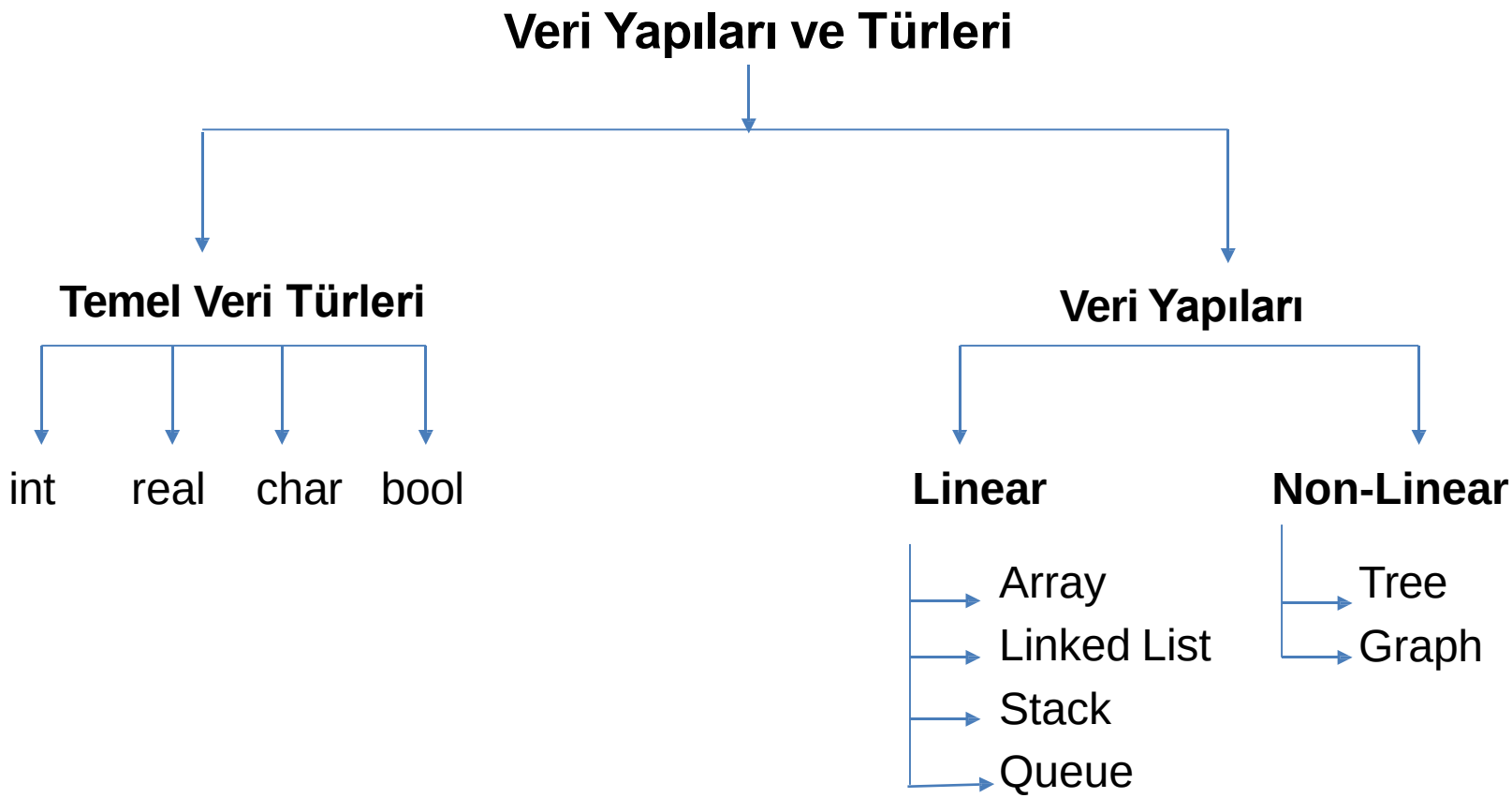
- Bir problemin çözümü için öncelikle *işlenecek verinin* **çözüme uygun** şekilde **organize** edilmesi gerekir. Bu bağlamda VY tanımı:
- "Veri Yapısı, verinin **verimli şekilde** kullanılabilmesi için *belirli bir formatta* **saklanıp organize edilmesini** sağlayan **matematiksel modeldir**."



Linear ve Non-Linear Veri Yapıları

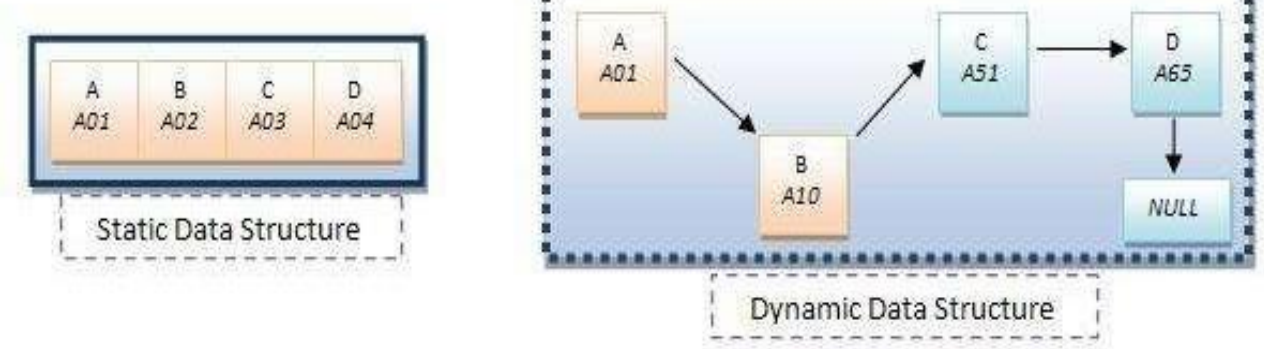
- Stack, Queue, Tree ve Graph'lar **Veri Yapılarına** birer örnektir.
- Verinin *organize edilmesine* bağlı olarak **Veri Yapıları** iki grupta sınıflandırılırlar:
  1. **Linear VY:** Elemanların (verinin) **sırayla erişildiği** Array, Linked List, Stack ve Queue gibi yapılardır.
  2. **Non-Linear VY:** Verinin lineer olmayan (**hiyerarşik, network**) bir şekilde **saklandığı-erişildiği** Tree ve Graph gibi yapılardır.

Linear ve Non-Linear Veri Yapıları (devam…)

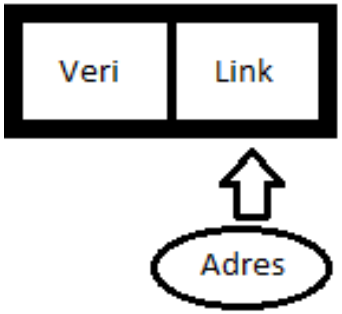


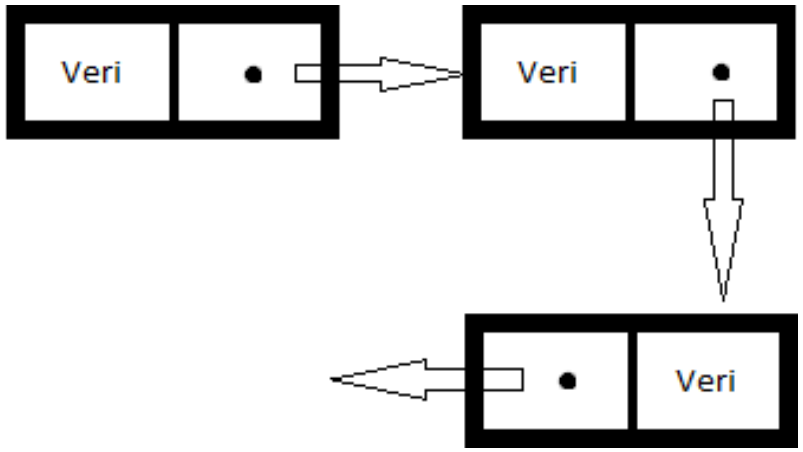
Veri Yapıları Implementasyonu

- VY'leri *implemente* etmek için iki yöntem bulunmaktadır:
  - ***Statik veri tipleri*** kullanmak (diziler). Statik veri türleri, *kendilerine ayrılan* **bölümün** tamamını kullanmasalar dahi **sabit** bir bellek alanı tahsis ederler.
  - ***Dinamik veri tipleri*** kullanmak (işaretçi, referans türleri). Program **çalışma zamanında** boyutları ***büyüyüp küçülebilir***.
- **Statik VY'ler**, **dinamik VY'**leri oluştururlar.

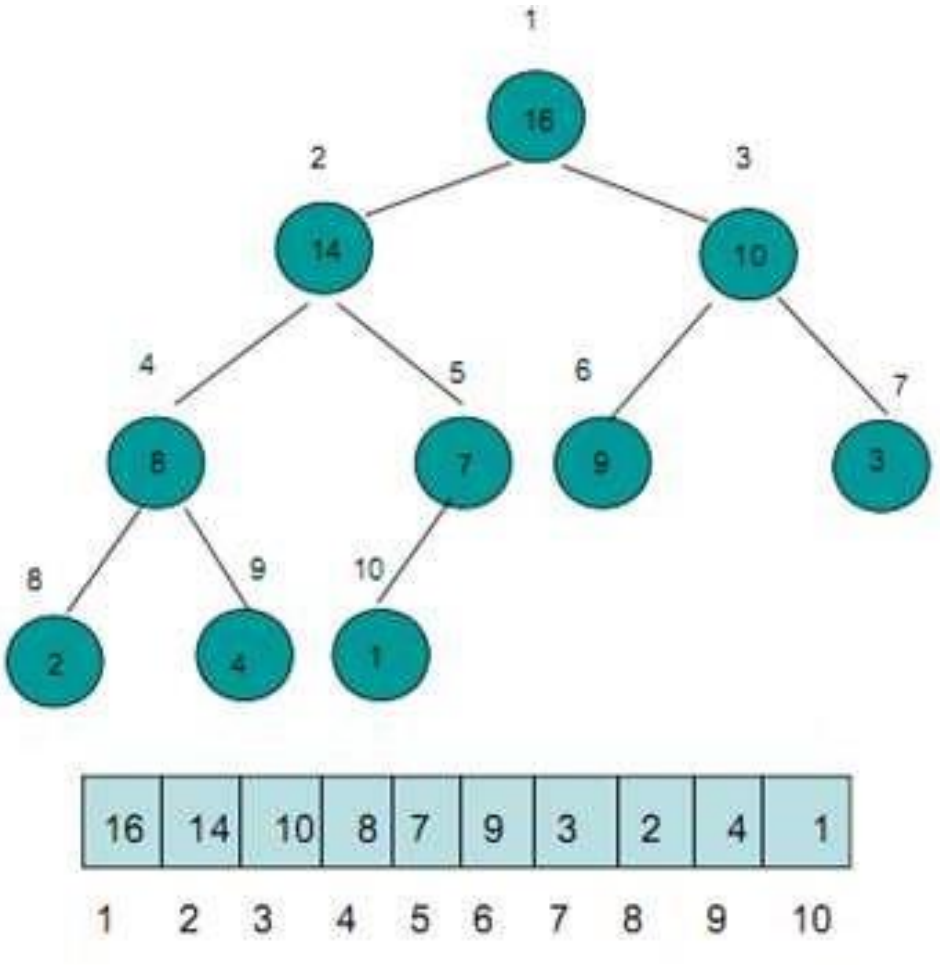


Örnek VY: Linked List (Bağlı Liste)





Örnek VY: Tree (Ağaç)



Hangi VY Kullanacağım?

Bir algoritma için **benzer VY modelleri** arasından seçim yapılırken **hangi kriterleri** göz önünde bulunduracağız?

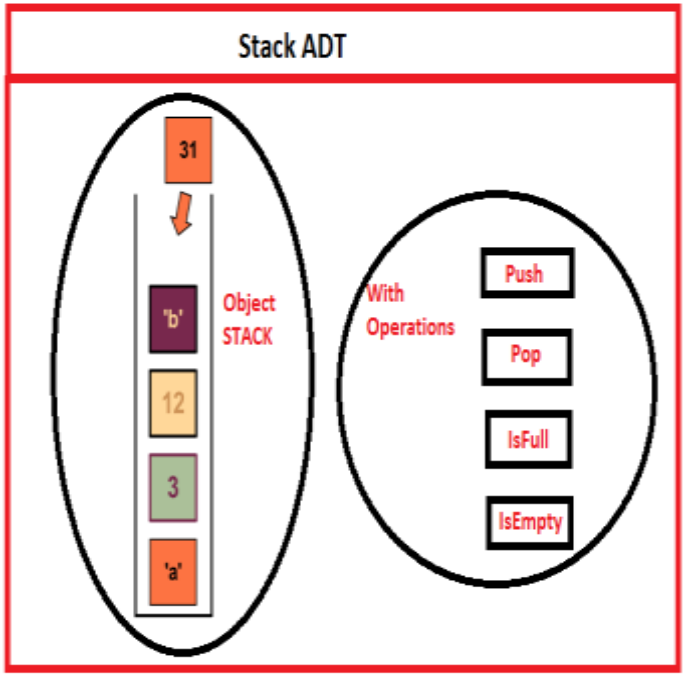
1. **Hafıza kullanımı**
2. **Verimliliği (kabaca hızı)**

Abstract Data Type (ADT)

- ADT (Abstract Data Type), VY bağlamında **bir problemin** *bilgisayar çözümü için* kullanılan, genelde matematiksel bir modeldir.
- Örneğin, **bir grup sayıyı sıralamak** için ADT,
  - bir dizi ve
  - dizi üzerinde çalışan sıralama algoritması olarak düşünülebilir.
- Bir başka ifadeyle, bir ADT
  - verinin organizasyonu ve
  - onların üzerinde **yapılacak işlemlerin** matematiksel (**kısmen**) modellenmesidir.
- Örneğin, bir **tamsayı dizisi** ve bu *dizi üzerinde* read, sort, search ve print işlemleri **basit bir ADT modelidir**.

Örnek: Stack ADT

Yaygın kullanılan ADT'ler'den biri olan **Stack ADT'si**, verinin saklanma biçimi ve bu ***veri üzerinde tanımlanan işlemler*** şeklinde modellenir.



Yığın İşlemleri	Açıklama
push()	Yığına eleman ekler.
pop()	En son eklenen elemanı ayırır.
top()	Son eklenen eleman değerini getirir.
size()	Eleman sayısını döndürür.
isEmpty()	Yığında eleman olup olmadığını kontrol eder.

Örnek: Stack ADT (devam…)

- ADT, kullanıcı açısından **arka plandaki karmaşıklığı kullanıcıdan soyutlama (abstraction) anlamında**, OOP mantığına **uygun** bir *yaklaşımdır*.
- ADT'ler NYP yaklaşımında ***Interface ve Abstract Class*** ile tanımlanırlar.
- Interface'de metot adları, parametre türleri ve döndürecekleri değer türleri gibi soyut tanımlamalar yapılırken metotlar ilgili Interface'i implemente eden sınıflarda tanımlanır.

Örnek: Stack ADT (devam…)

```
interface StackADT
{
        Boolean isEmpty();
        void Push(Object element);
        Object Pop();
        Object Peek();
        void Display();
}
```

Telif ve Kaynaklar

Doç.Dr. Deniz KILIÇ'ın ders notlarından yararlanılmıştır.

Kaynak gösterilmek şartıyla her türlü kullanıma açıktır.