BMB2006
VERİ YAPILARI

Doç. Dr. Murtaza CİCİOĞLU

Bursa Uludağ Üniversitesi

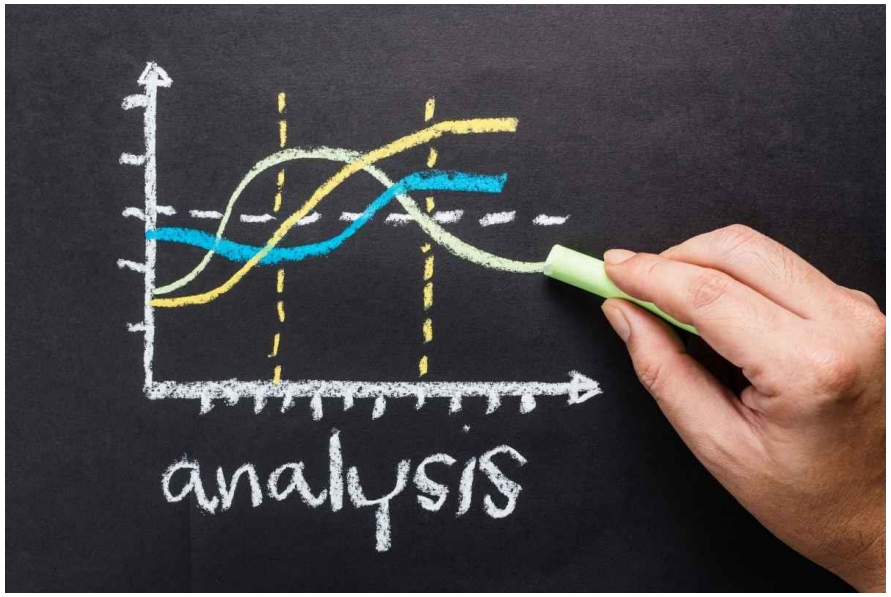
Bilgisayar Mühendisliği Bölümü

Bu ders notları Olcay Taner Yıldız ile Erkan TANYILDIZI hocaların notları kullanılarak hazırlanmıştır.

Hafta 3: Bağlı Liste

Amaç:

- Bağlı liste çalışma yapısı
- Bağlı liste çeşitleri



Yol haritası:

- Giriş
- Bağlı Liste Tanımı
- Temel Bağlı Liste İşlemleri
- İki Listeyi Birleştirme
- Çift Bağlı Liste
- Temel Çift Bağlı Liste
- Dairesel Bağlı Liste

Giriş

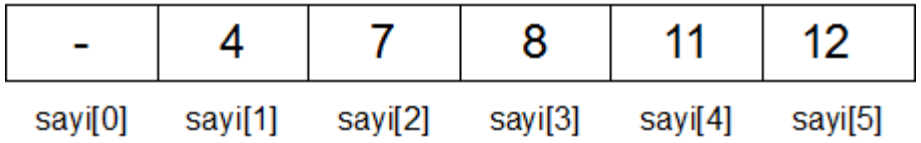
- Günlük hayatta listeler; alışveriş listeleri, davetiye, telefon listeleri vs. kullanılır.
- Programlama açısından liste; aralarında doğrusal ilişki olan veriler topluluğu olarak görülebilir.
- Veri yapılarında değişik biçimlerde listeler kullanılmakta ve üzerlerinde değişik işlemler yapılmaktadır.

Doğrusal Listeler (Diziler)

- Diziler(arrays), doğrusal listeleri oluşturan yapılardır. Bu yapıların özellikleri:
  - Süreklilik, elemanlar aynı türden olup bellekte art arda saklanır
  - Dizi elemanları arasında başka elemanlar bulunamaz. Diziye eleman eklemek gerektiğinde yer değiştirme gerekir.
  - Program başında tanımlanır ve ayrılacak bellek alanı belirtilir. Dinamik değildir.

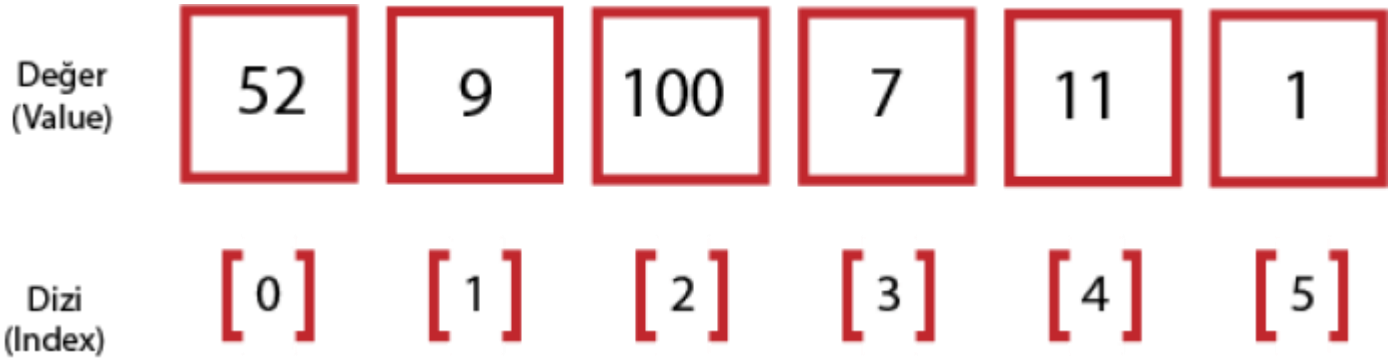
Veriler: 4, 7, 8, 11, 12

Sayı Dizisi



Doğrusal Listeler (Diziler)

- Dizinin boyutu baştan çok büyük tanımlanabilir!!!
- Diziye eleman ekleme veya çıkarma!!!
- Dizinin sıralanması!!!



Doğrusal Listeler (Diziler)

- Bir dizinin k'ncı elemanını silen algoritma

```
void dizidenSil(int[] dizi , int k){
    int i;
    for (i = k; i < dizi .length - 1; i++){
        dizi [i] = dizi [i + 1];
    }
}
```

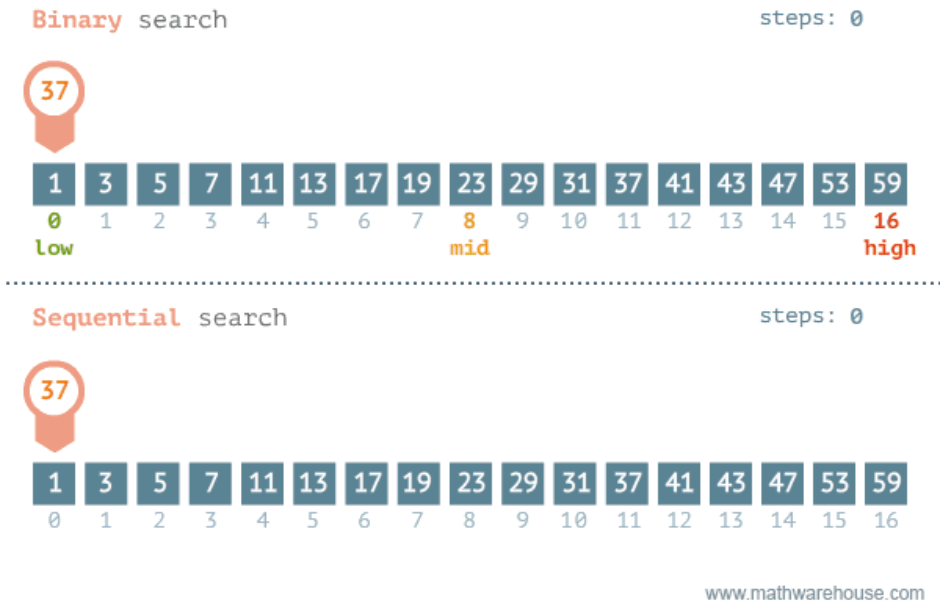
Doğrusal Listeler (Diziler)

- Bir dizinin k'nıncı yerine yeni eleman ekleyen algoritma

```
void diziyeEkle(int[] dizi, int k, int yeni){
    int i;
    for (i = dizi.length - 2; i >= k; i --){
        dizi[i + 1] = dizi[i];
    }
    dizi[k] = yeni;
}
```

Doğrusal Listeler (Diziler)

- Sıralı bir dizide ikili arama yöntemiyle k sayısını arayan algoritma




```
void dizideAra(int[] dizi, int k){
   int sol = 0, sag = dizi.length - 1, orta;
   orta = (sol + sag) / 2;
   while (sol <= sag){
      if (k < dizi[orta])
         sag = orta - 1;
      else
         if (k > dizi[orta])
            sol = orta + 1;
         else
            return orta;
      orta = (sol + sag) / 2;
   }
   return -1;
}
```

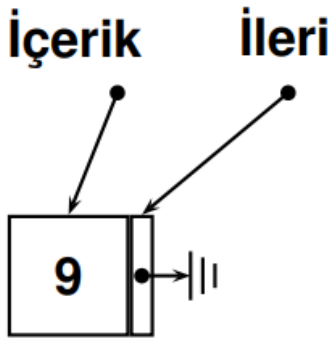
BAĞLI LİSTELER

- Dizilerde;
- Boyut değiştirmek
- Yeni bir eleman eklemek
- Bir elemanı silmek
- Dizinin tüm elemanları için hafızada yer ayırmak

- Bağlı listede;
- Her liste elamanı için ayrı hafıza alanı ayrılır.
- Bilgi kavramsal olarak sıralıdır ancak hafızada bulunduğu yer sıralı değildir.
- Her bir eleman (node) bir sonrakini gösterir.

BAĞLI LİSTELER

- Bellekte ardışık olarak bulunmayan yapılar
- Her eleman kendinden sonraki elemanın nerede olduğu bilir.
- Bağlı liste dizisinin her elemanı bir struct nesnesidir.

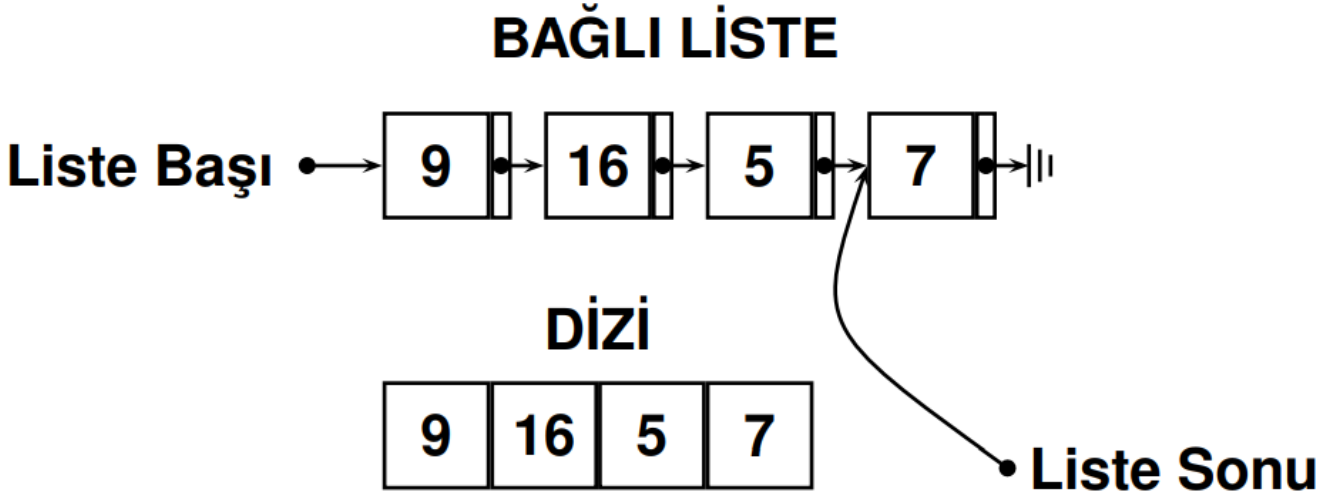



```
struct node{
        int icerik;
        struct node* ileri;
        };
typedef struct node Dugum;
```

```
public class Eleman{
    int icerik;
    Eleman ileri;
    public Eleman(int icerik){
        this.icerik = icerik;
        ileri  = null;
    }
}
```

BAĞLI LİSTELER

- İki boyutlu dizi yapısı
- Silinen veri alanları hala listede yer alır, veri silindiği halde listenin boyu kısalmaz.
- Dinamik bir veri yapısına ihtiyaç vardır.



	Sütun 0	Sütun 1	Sütun 2	Sütun 3
Satır 0	a[0,0]	a[0,1]	a[0,2]	a[0,3]
Satır 1	a[1,0]	a[1,1]	a[1,2]	a[1,3]
Satır 2	a[2,0]	a[2,1]	a[2,2]	a[2,3]

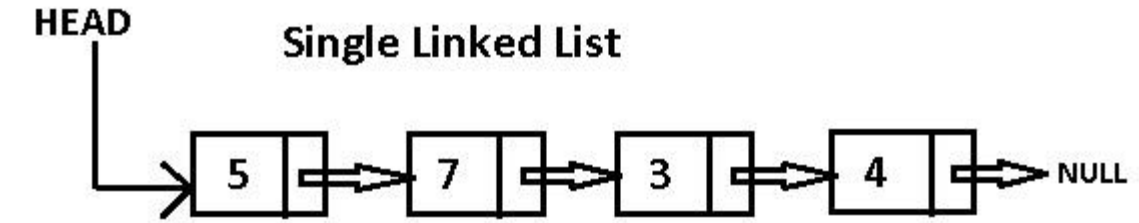
Dizi adı

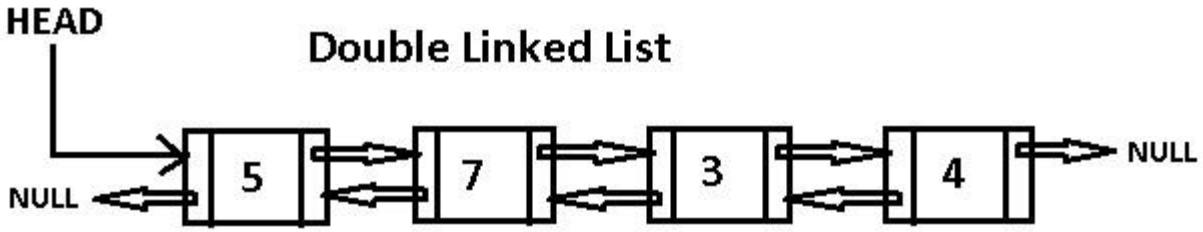
Satır indeksi

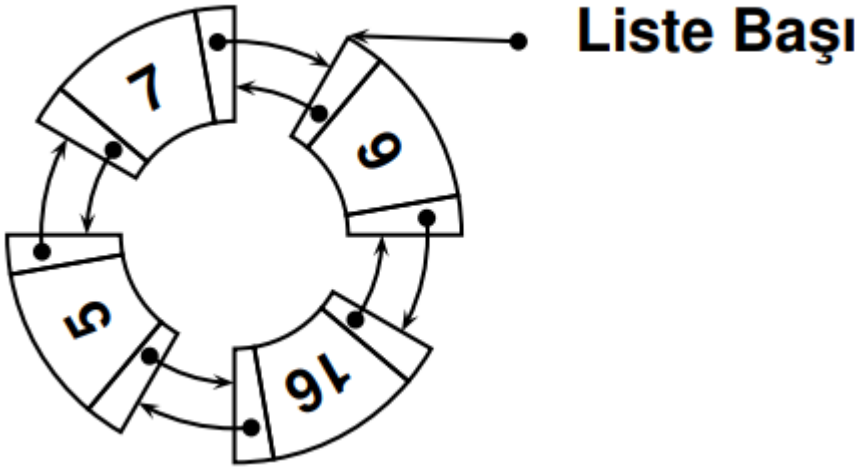
sütun indeksi

BAĞLI LİSTELER

- Tek yönlü doğrusal bağlı liste
- İki yönlü doğrusal bağlı liste
- Tek yönlü dairesel bağlı liste
- İki yönlü dairesel bağlı liste





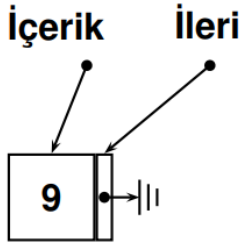


Bağlı Liste İşlemleri

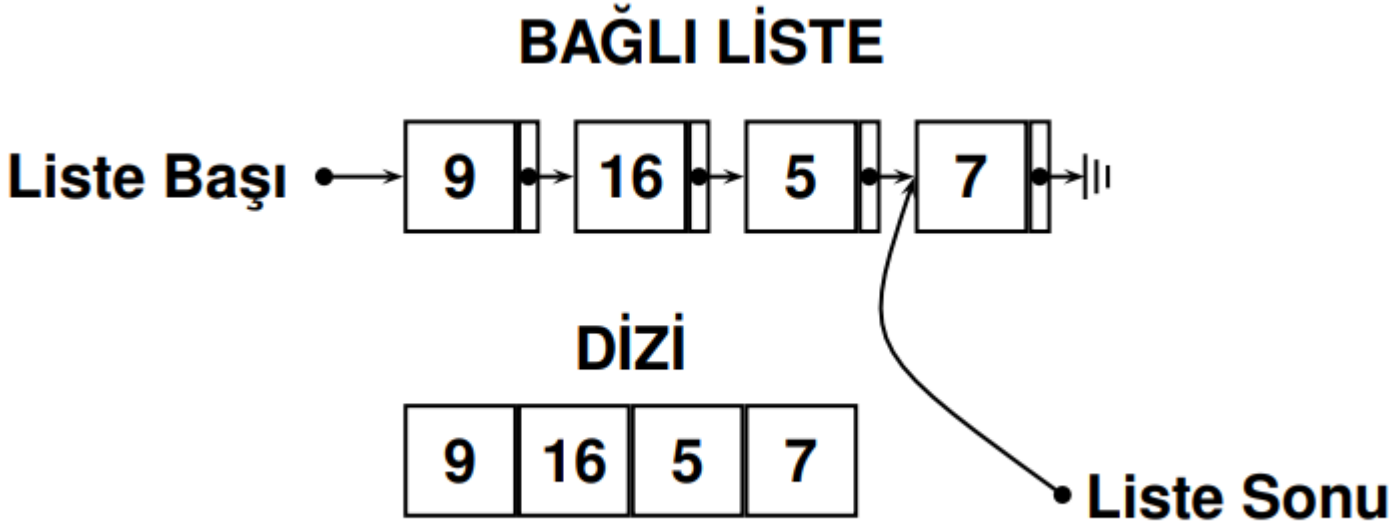
- Listeye eleman ekleme
  - Başına, Sonuna, Ortasına
- Listeden eleman silme
  - İlk, Son, Ortasından
- Arama
- Eleman sayısı
- Listeleme
- Kontrol
  - Boş liste
  - Liste boyutu

Bir bağlı liste ve dizi yapısı

- Tam sayılar içeren baglı liste tanımı



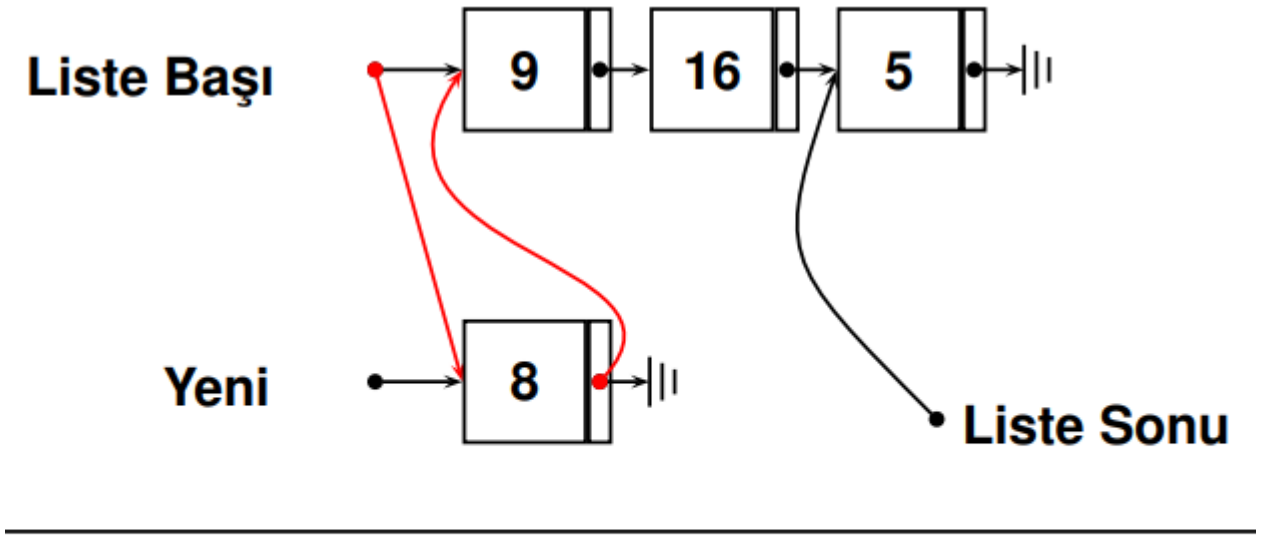

```
public class Eleman{
    int icerik;
    Eleman ileri;
    public Eleman(int icerik){
        this.icerik = icerik;
        ileri = null;
    }
}
```

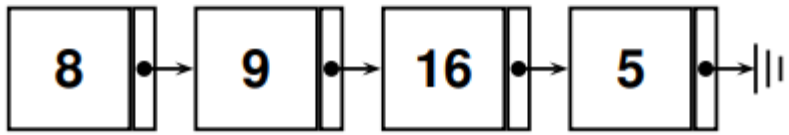



```
public class Liste{
    Eleman bas;
    Eleman son;
    public Liste(){
        bas = null;
        son = null;
    }
}
```

Temel Bağlı Liste İşlemleri

- Bağlı listenin başına eleman ekleme

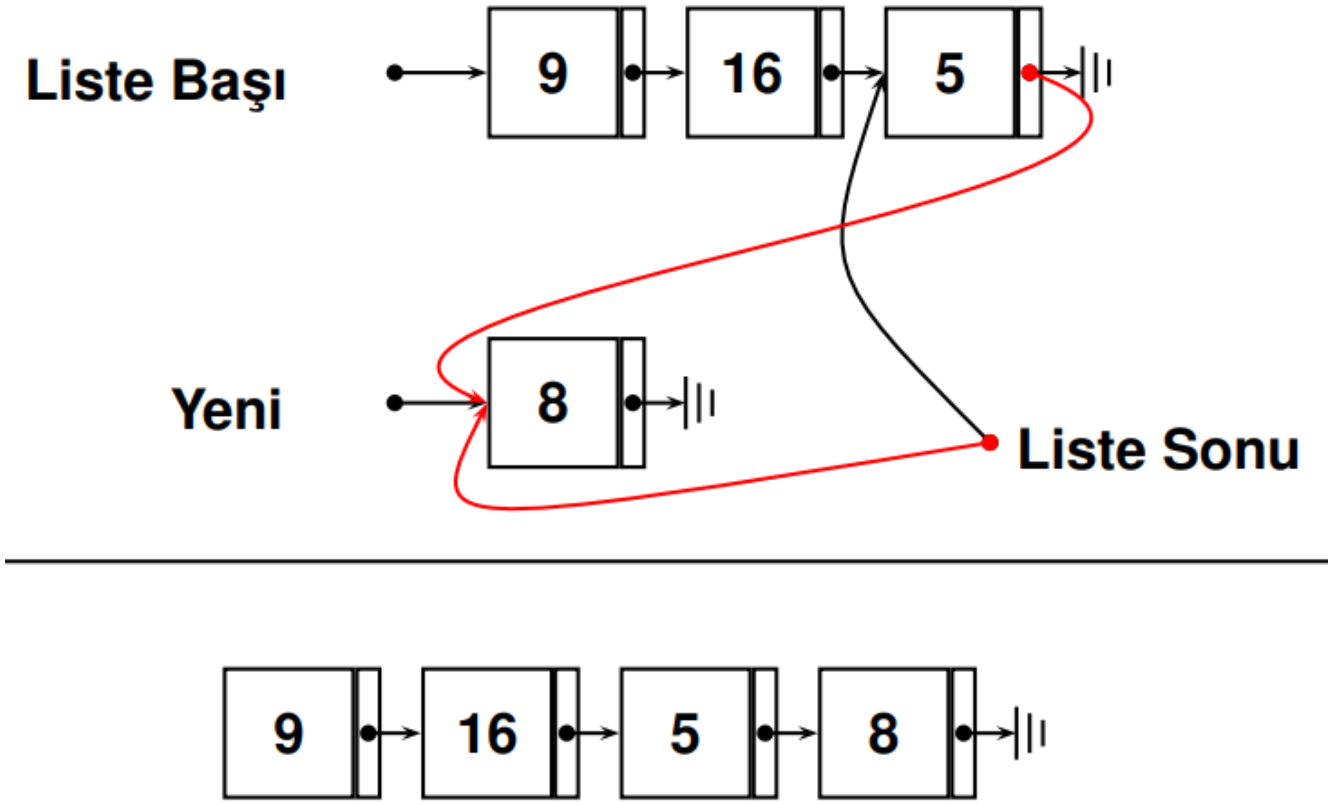





```
void listeBasinaEkle(Eleman yeni){
    if (son == null)
        son = yeni;
    yeni. ileri  = bas;
    bas = yeni;
}
```

Temel Bağlı Liste İşlemleri

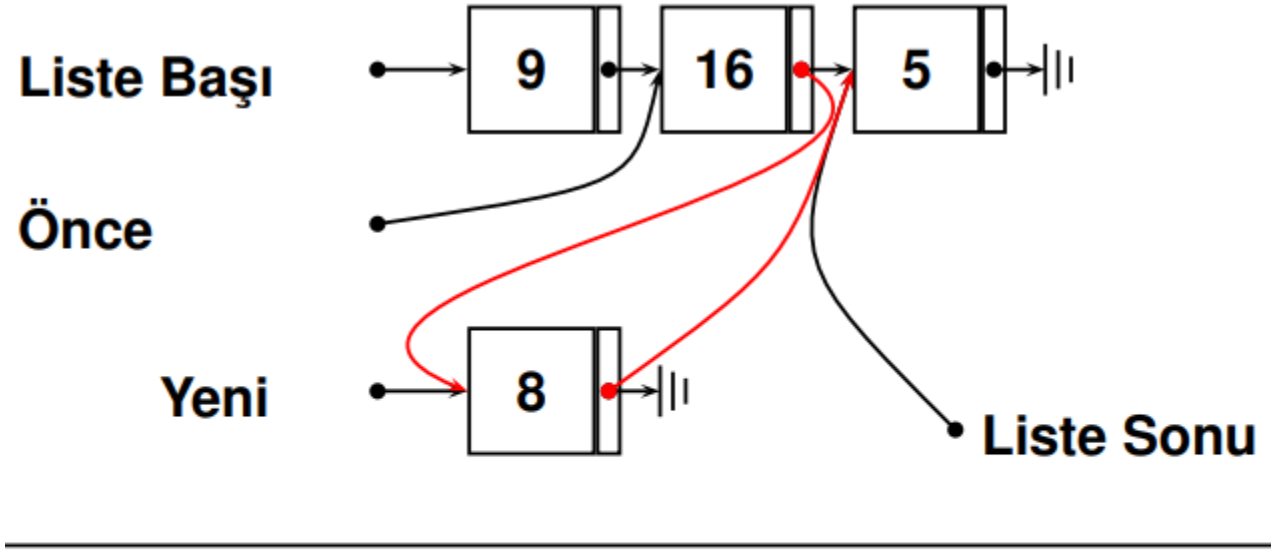
- Bağlı listenin sonuna eleman ekleme

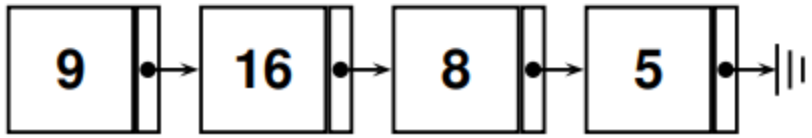



```
void listeyeEkle(Eleman yeni){
    if (bas == null)
        bas = yeni;
    else
        son. ileri  = yeni;
    son = yeni;
}
```

Temel Bağlı Liste İşlemleri

- Bağlı listenin ortasına eleman ekleme





Liste başına ekleme: $\mathcal{O}(1)$
Liste sonuna ekleme: $\mathcal{O}(1)$
Liste ortasına ekleme: $\mathcal{O}(1)$

```
void listeOrtaEkle(Eleman yeni, Eleman once){
    yeni. ileri  = once. ileri ;
    once. ileri  = yeni;
}
```

Temel Bağlı Liste İşlemleri

- Baglı listede verilen bir değeri arama
- Baglı listenin i'ninci elemanını döndürme

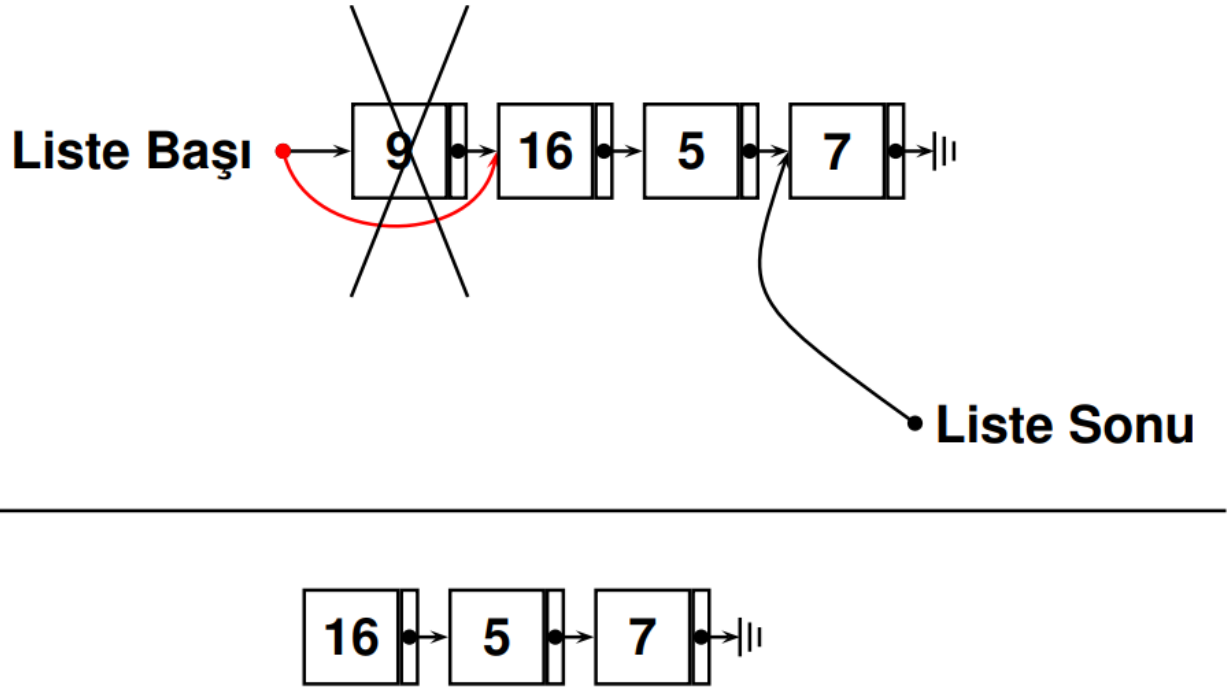
```
Eleman listeAra(int deger){
    Eleman tmp;
    tmp = bas;
    while (tmp != null){
        if (tmp.icerik  == deger)
            return tmp;
        tmp = tmp. ileri ;
    }
    return null;
}
```

```
Eleman elemanI(int i){
    Eleman tmp = bas;
    int j = 0;
    while (tmp != null && j < i){
        j++;
        tmp = tmp. ileri ;
    }
    return tmp;
}
```

Listede arama: $\mathcal{O}(N)$
i'ninci elemanı getirme: $\mathcal{O}(N)$

Temel Bağlı Liste İşlemleri

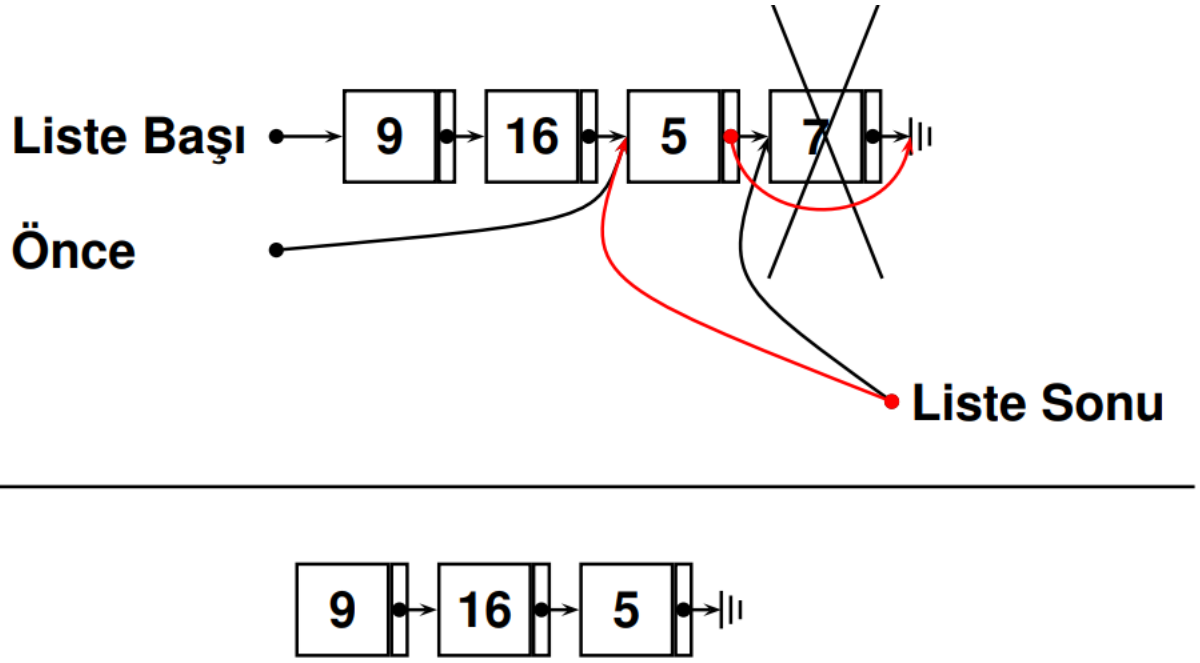
- Baglı listenin ilk elemanını silme




```
void listeBasiSil (){
    bas = bas. ileri ;
    if (bas == null)
        son = null;
}
```

Temel Bağlı Liste İşlemleri

- Baglı listenin son elemanını silme

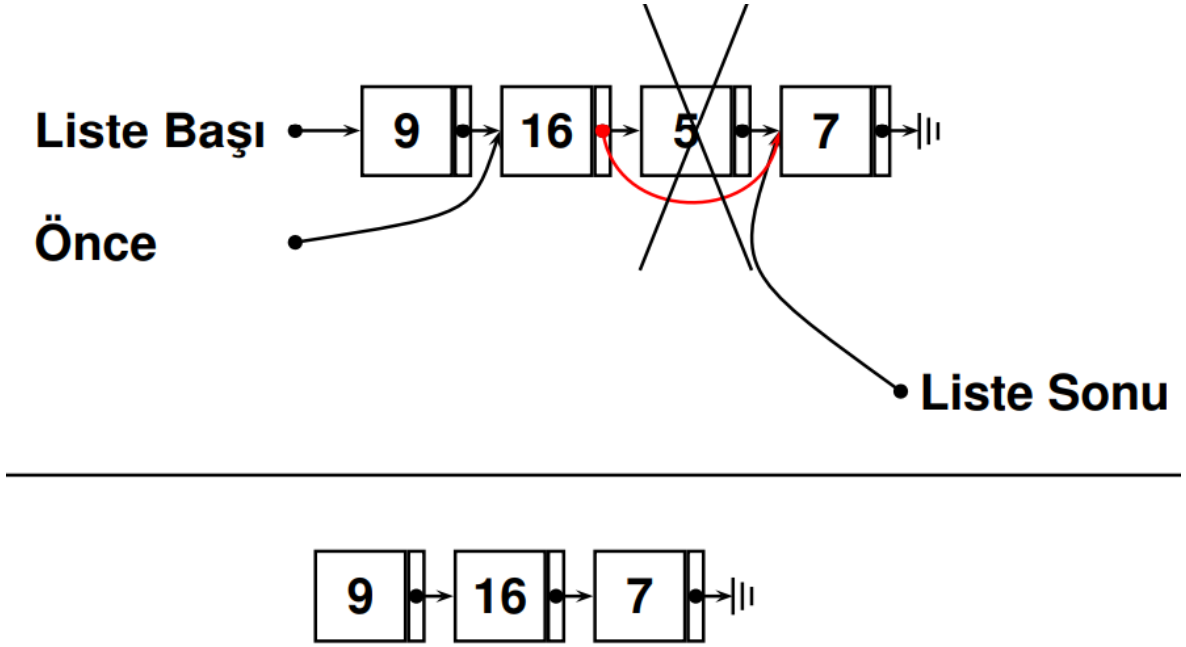



```
void listeSonuSil(){
    Eleman tmp, once;
    tmp = bas;
    once = null;
    while (tmp != son){
        once = tmp;
        tmp = tmp. ileri ;
    }
    if (once == null)
        bas = null;
    else
        once. ileri  = null;
    son = once;
}
```

Temel Bağlı Liste İşlemleri

- Baglı listenin ortasından elemanı silme

Listenin ilk elemanını silme: $\mathcal{O}(1)$
Listenin son elemanını silme: $\mathcal{O}(N)$
Listenin ortasından silme: $\mathcal{O}(N)$




```
void listedenSil(Eleman s){
    Eleman tmp, elemanonce;
    tmp = bas;
    elemanonce = null;
    while (tmp != s){
        elemanonce = tmp;
        tmp = tmp. ileri ;
    }
    elemanonce.ileri = s. ileri ;
}
```

Temel Bağlı Liste İşlemleri

- Baglı listedeki eleman sayısını bulma

```
int elemanSayisi(){
    int sayac = 0;
    Eleman tmp;
    tmp = bas;
    while (tmp != null){
        tmp = tmp. ileri ;
        sayac++;
    }
    return sayac;
}
```