İki Yönlü Bağlı Listeler

Doç. Dr. Fatih ÖZYURT

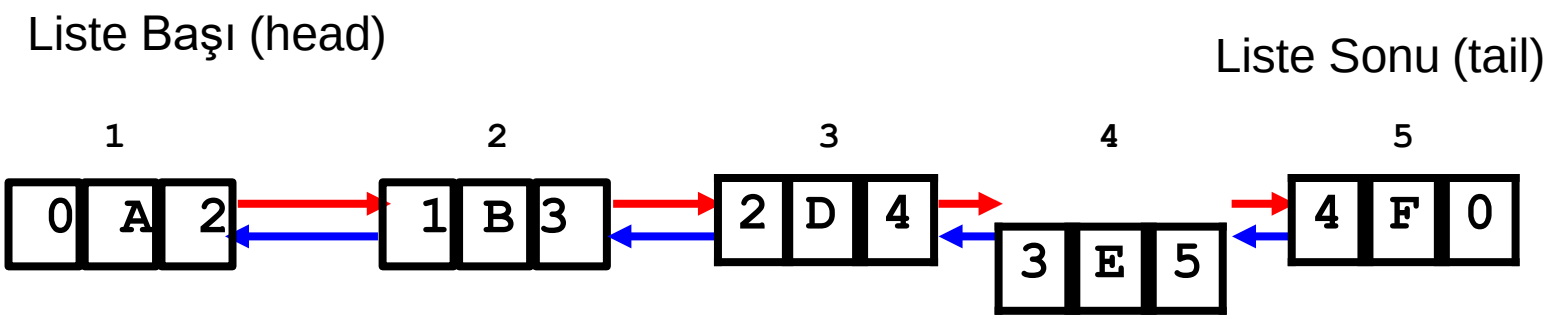
İki Yönlü Bağlı Listeler:

Bağlı listelerde bir düğüm kendisinden sonra gelen düğümün adres bilgisini ve kendisinden önceki düğümün adres bilgisini içerebilir. Bu şekilde bir yapıya sahip bağlı listelere çift yönlü bağlı liste (**Doubly Linked List**) denir.

Çift yönlü bağlı listelerde bir düğümde bulunan iki işaretçiden önceki düğümün adresini gösteren işaretçi **önceki**, sonraki düğümün adresini gösteren işaretçi **sonraki** olarak isimlendirilebilir. Şekil 1 ‘de çift yönlü bağlı listede bir düğümün yapısı gösterilmiştir.

Şekil 1

- Şekil 2'de çift yönlü bağlı listenin doğrusal gösterimi verilmiştir. Şekilden de anlaşılacağı gibi Çift yönlü bağlı listeler (DLL), tek bağlantılı listede bulunan sonraki işaretçi ve verilerle birlikte, önceki işaretçi olarak adlandırılan ekstra bir işaretçiye de sahiptir.
- Çift Yönlü Bağlı Listeler doğrusal olarak tasarlanabileceği gibi dairesel olarak da tasarlanabilir.



Şekil 2

- Çift Yönlü Bağlı Listelerde son düğümden sonra liste başına geri gitmek mümkündür. Bunun için son düğümden itibaren listenin elemanları geriye doğru taranır. Bu işlemin n-1 elemana erişmek anlamına geldiğini söyleyebiliriz.
- C programlama dili kullanılarak, işaretçi yaklaşımı ile bir yapı aşağıdaki şekilde tanımlanır;

```
struct cbagliListe{
    int veri1;
    int veri2;
    struct cbagliListe *onceki;
    struct cbagliListe *sonraki;
};
struct cbagliListe* ilkDugum=NULL;
struct cbagliListe* sonDugum=NULL;
```

Çift Yönlü Bağlı Listelerin Avantajları ve Dezavantajları

Avantajları

- Liste üzerinde çift yönlü hareket edilebilir,
- Ekleme, Silme gibi bazı işlemler daha kolaydır.

Dezavantajları

- Önceki işaretçi için bellekte fazladan yer kaplar,
- Her düğümün önceki (prev) ve sonraki (next) adında iki işaretçisi olduğu için liste işlemleri daha yavaştır,
- Hata yapılma ihtimali yüksektir. Örneğin listeye eleman ekleme sırasında, önceki işaretçileri sonraki işaretçilerle birlikte değiştirmemiz gerekir. Herhangi bir işlemin atlanması hataya neden olur.

Çift Yönlü Bağlı Liste Oluşturmak

```
struct cbagliListe *dOlustur(int veri1, int veri2){
    struct cbagliListe* yeniDugum=
    (struct cbagliListe*)malloc(sizeof(struct cbagliListe));
    yeniDugum->veri1=veri1;
    yeniDugum->veri2=veri2;
    yeniDugum->onceki=NULL;
    yeniDugum->sonraki=NULL;
    return yeniDugum;
}
```

Liste Yazdır

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
struct cbagliListe{
    int veri1;
    int veri2;
    struct cbagliListe *onceki;
    struct cbagliListe *sonraki;
};
struct cbagliListe* ilkDugum=NULL;
struct cbagliListe* sonDugum=NULL;

struct cbagliListe *dOlustur(int veri1, int veri2){
    struct cbagliListe* yeniDugum=
    (struct cbagliListe*)malloc(sizeof(struct cbagliListe));
    yeniDugum->veri1=veri1;
    yeniDugum->veri2=veri2;
    yeniDugum->onceki=NULL;
    yeniDugum->sonraki=NULL;
    return yeniDugum;
}
void listeYazdir(struct cbagliListe* ilkDugum, int yon){
    struct cbagliListe* temp=ilkDugum;
    while(temp!=NULL)
    {
        printf("\n(%d  %d  )",temp->veri1,temp->veri2);
        if(yon==1){
          printf(" %d =>\n",temp->sonraki);
          temp=temp->sonraki;
            }
          else{
          printf(" %d =>\n",temp->onceki);
          temp=temp->onceki;
                }
    }
}

int main() {
    struct cbagliListe *birinci=dOlustur(13,23);
    struct cbagliListe *ikinci=dOlustur(33,43);
    struct cbagliListe *ucuncu=dOlustur(53,63);
    struct cbagliListe *dorduncu=dOlustur(73,83);
    struct cbagliListe *besinci=dOlustur(93,103);
    ilkDugum=birinci;
    sonDugum=besinci;
    birinci->sonraki=ikinci;
    ikinci->sonraki=ucuncu;
    ucuncu->sonraki=dorduncu;
    dorduncu->sonraki=besinci;
    besinci->onceki=dorduncu;
    dorduncu->onceki=ucuncu;
    ucuncu->onceki=ikinci;
    ikinci->onceki=birinci;
    listeYazdir(birinci,1);
    return 0;
}
```

Çift Yönlü Bağlı Listede Düğüm Sayısını Bulmak

```
void Say(){
    int kontrol = 0,sayac=1;
    if(ilkDugum == NULL){
        printf("\n ==>Listede Herhangi Bir eleman yok \n");
        return;
    }
     struct cbagliListe* temp = ilkDugum;
     while(temp->sonraki != NULL){
        sayac+=1;
        kontrol=1;
        temp = temp->sonraki;
    }
    if(kontrol == 1){
        printf("\n ==>Listede %d Dügüm var \n",sayac);
        return;
    }
```

Çift Yönlü Bağlı Listenin Sonuna Eleman Eklemek

```
void sonaEkle(int veri1,int veri2){
    if(ilkDugum == NULL) {
    ilkDugum = (struct cbagliListe *)malloc(sizeof(struct cbagliListe));
    ilkDugum -> veri1 = veri1;
    ilkDugum -> veri2 = veri2;
    ilkDugum -> sonraki = NULL;
    ilkDugum -> onceki = NULL;
    }
    else {
        struct cbagliListe *temp1 = ilkDugum;
        struct cbagliListe *temp2 =
        (struct cbagliListe *)malloc(sizeof(struct cbagliListe));
        while(temp1 -> sonraki != NULL)
        temp1 = temp1 -> sonraki;
        temp2 -> veri1 = veri1;
        temp2 -> veri2 = veri2;
        temp2 -> sonraki = NULL;
        temp2 -> onceki = temp1;
        temp1 -> sonraki = temp2;
}
```

Çift Yönlü Bağlı Listenin Başına Eleman Eklemek

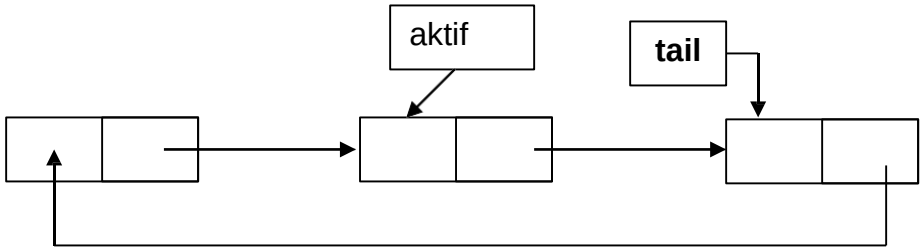
```
void basaEkle(int veri1,int veri2){
    if(ilkDugum==NULL)
    {
        ilkDugum = (struct cbagliListe *)malloc(sizeof(struct cbagliListe));
        ilkDugum -> veri1 = veri1;
        ilkDugum -> veri2 = veri2;
        ilkDugum -> sonraki = NULL;
        ilkDugum -> onceki = NULL;
        }
        else {
                struct cbagliListe *temp =
                (struct cbagliListe *)malloc(sizeof(struct cbagliListe));
                 temp -> veri1 = veri1;
                 temp -> veri2 = veri2;
                 temp -> sonraki = ilkDugum;
                 temp -> onceki = NULL;
                 ilkDugum -> onceki = temp;
                 ilkDugum = temp;
```

Çift Yönlü Bağlı Listenin Arasına Eleman Eklemek

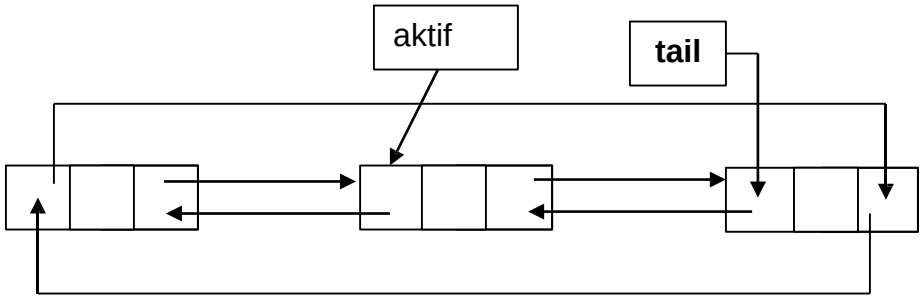
```
void arayaEkle(int veri1,int veri2,int sira){
    struct cbagliListe* ArayaEklenecek = dOlustur(veri1,veri2);

    if(sira == 0){
        basaEkle(veri1,veri2);
        return;
    }
    if(ilkDugum == NULL && sira > 0){
        printf("\n Ekleme işlemi yapılamaz ");
        return;
    }
    int sayac = 0,kontrol=0;
    struct cbagliListe* temp = ilkDugum;
    while(temp != NULL){
        if(sayac == sira){
            kontrol = 1;
            break;
        }
        temp = temp->sonraki;
        sayac++;
    }
    if(kontrol == 0){
        printf("\n Eklenecek Pozisyon Yok\n");
        return;
    }
    if(temp->sonraki == NULL){
        sonaEkle(veri1,veri2);
        return;
    }
    struct cbagliListe* OncekiDugum = temp->onceki;
    OncekiDugum->sonraki = ArayaEklenecek;
    ArayaEklenecek->onceki = OncekiDugum;
    ArayaEklenecek->sonraki = temp;
    temp->onceki = ArayaEklenecek;
}
```

Dairesel Bağlı Listeler:

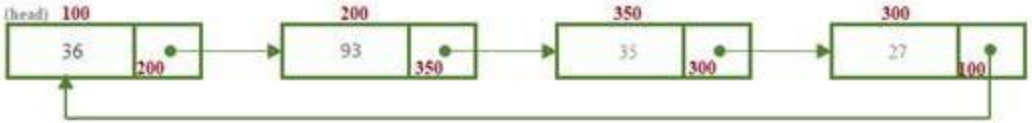
- **Tek Yönlü Dairesel Bağlı Listeler** : Listedeki elemanlar arasında tek yönlü bağ vardır. Tek yönlü bağlı listelerden tek farkı ise son elemanın göstericisi ilk listenin ilk elamanının adresini göstermesidir. Bu sayede eğer listedeki elemanlardan birinin adresini biliyorsak listedeki bütün elemanlara erişebiliriz.



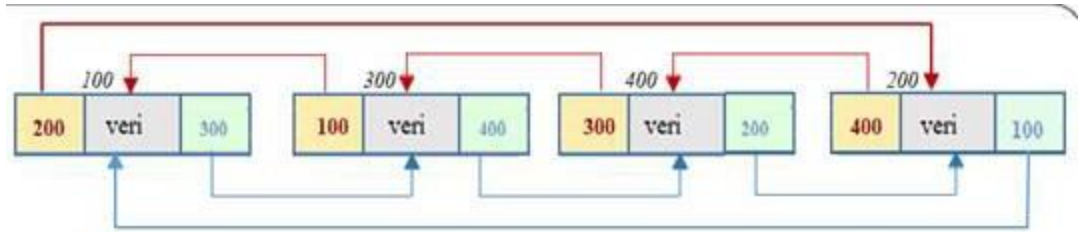
- **İki Yönlü Dairesel Bağlı Listeler** : Hem dairesellik hem de çift bağlılık özelliklerine sahip listelerdir. İlk düğümden önceki düğüm son, son düğümden sonraki düğüm de ilk düğümdür.



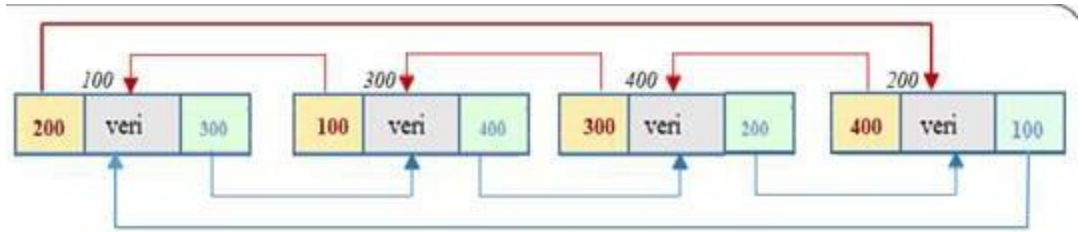
Dairesel Tek Yönlü Bağlı Listeler



Dairesel İki Yönlü Bağlı Listeler



Dairesel İki Yönlü Bağlı Listeler



Kaynaklar

Veri Yapıları ve Algoritmalar – Dr. Rifat ÇÖLKESEN, Papatya yayıncılık
Veri Yapıları ve Algoritmalar-Dr. Öğ. Üyesi Ömer ÇETİN
Veri Yapıları – Prof. Dr. Erkan TANYILDIZI