İkili Arama Ağaçları: Ekleme, Silme, Arama, Güncelleme İşlemleri

İkili arama ağacını oluşturmak için gerekli düğüm yapısı ve işaretçi tanımlanmıştır. Her düğümün bir içeriği, sağ ve solundaki çocuklarını gösteren sag ve sol işaretçileri vardır. Ayrıca ağacın kök elemanını gösteren kok işaretçisi bulunmaktadır.

```
class Dugum {
    int icerik;
    Dugum sag;
    Dugum sol;
    Dugum(int icerik){
        this.icerik=icerik;
        sag=null;
        sol=null;
    }
}
    public class İkiliAramaAgaci {
    Dugum kok = null;
```

Ağaca eleman eklemek için ilk olarak eklenecek elemanın nereye yerleştirileceğinin bulunması gerekir. Bunun için ağacın kökünden başlayarak x düğümü ile aşağı doğru ilerlenir. Her defasında bulunulan düğümün değeri ile eklenecek elemanın değeri karşılaştırılır. Eğer eklenecek elemanın değeri düğümün değerinden küçük ise yeni eleman ağacın sol tarafına eklenecektir. Bu yüzden ağacın sol tarafına doğru ilerlenir. Eğer eklenecek elemanın değeri düğümün değerinden büyük ise yeni eleman ağacın sağ tarafına eklenecektir. Bu yüzden ağacın sağ tarafına doğru ilerlenir. Eklenecek elemanın yeri bulunurken bağlantıları ayarlamayı kolaylaştırmak için o anda bulunulan düğümün ebeveyni y düğümünde saklanır. Bu işlem while döngüsü ile tanımlanan kısımda yapılır. Eklenecek elemanın yeri bulunduğunda, bulunulan x düğümünün değeri null gösterir. Ancak x düğümünün ebeveyni y düğümü tutulduğu için bu düğüm üzerinden yeni eleman eklenir. Ebeveyn null ise ağacın kökü yeni eklenen düğüm yani z düğümü yapılır. Eğer null değilse yeni düğümün y düğümün sağına ya da soluna mı ekleneceği belirlenir. Eğer elemen y düğümünün içeriğinden büyük ise sağa küçük ise sola yeni düğüm eklenir.

```
    public void Ekleme(int eleman){
    Dugum y = null;
    Dugum x = kok;
    Dugum z = new Dugum(eleman);
    while(x!=null){
        y=x;
        if(eleman<x.icerik)
            x = x.sol;
        else
            x = x.sag;
    }
    if(y==null)
        kok = z;
    else
        if(eleman<y.icerik)
            y.sol = z;
        else
            y.sag = z;
}
```

Silme işleminde ilk olarak silinecek elemanın ağacın hangi düğümü olduğu bulunur. Bunun için kök düğümünden başlanarak silinecek elemanın içeriği ile karşılaştırılır. Silinecek eleman kökten büyük ise sağ alt ağaçta küçük ise sol alt ağaçta arama yapılır ve silinecek düğüm bu işlem sonucunda kok değişkeninde yer alır.

```
    public void Silme(int eleman){
        kok = sil(kok,eleman);
        }
    public Dugum sil(Dugum kok, int eleman){
        if(kok==null)
            return null;
        if(eleman<kok.icerik)
            kok.sol = sil(kok.sol,eleman);
        else if(eleman>kok.icerik)
            kok.sag = sil(kok.sag,eleman);
```

Silme işleminde üç durum söz konusudur. Birinci durum silinecek elemanın çocuğu yoksa düğüm null yapılır ve silme tamamlanır. İkinci durum silinecek elemanın sağında veya solunda bir çocuğa sahip ise bu çocuk silinen düğümün yerine geçirilir. Üçüncü durum silinecek elemanın sağında ve solunda iki çocuğu varsa iki farklı şekilde silme işlemi yapılır. Ya silinen düğümün yerine sol alt ağacının maksimum elemanı getirilir ya da sağ alt ağacının minimum elemanı getirilir. Daha sonra bu silinecek düğümün yerine getirilen maksimum ya da minimum düğüm ağaçtan silinir.

```
        else{
            if(kok.sol!=null && kok.sag!=null){
//                  Dugum MinSag = EnKucuk(kok.sag);
//                  kok.icerik = MinSag.icerik;
//                  kok.sag = sil(kok.sag,MinSag.icerik);
                Dugum MaxSol = EnBuyuk(kok.sol);
                kok.icerik = MaxSol.icerik;
                kok.sol = sil(kok.sol,MaxSol.icerik);
            }
            else if(kok.sol!=null){
                kok = kok.sol;
            }
            else if(kok.sag!=null){
                kok = kok.sag;
            }
            else{
                kok = null;
            }
        }
        return kok;
```

```
    public Dugum EnBuyuk(Dugum d){
    while(d.sag!=null){
        d = d.sag;
    }
    return d;
}

    public Dugum EnKucuk(Dugum d){
    while(d.sol!=null){
        d = d.sol;
    }
    return d;
}
```

Arama işleminde kökten başlanarak ağacın alt dallarına doğru ilerlenir. Eğer aranan eleman bulunulan düğümün içeriğinden küçük ise sol alt ağaçta aramaya devam edilir, büyük ise sağ alt ağaçta aramaya devam edilir. Eğer eleman ağaçta bulunuyor ise true, bulunmuyor ise false değeri geri döndürülür.

```
    public boolean Arama(int eleman){
    Dugum d;
    d = kok;
    while(d!=null){
        if(d.icerik==eleman)
            return true;
        else
            if(d.icerik>eleman)
                d = d.sol;
            else
                d = d.sag;
    }
    return false;
}
```

Güncelleme işleminde basit olarak eski değeri içeren düğüm Silme metodu ile ağaçtan silinir. Yeni eklenecek eleman ise Ekleme metodu ile ağaca eklenir.

```
    public void Güncelleme(int eski_deger, int yeni_deger){
    Silme(eski_deger);
    Ekleme(yeni_deger);
}
```