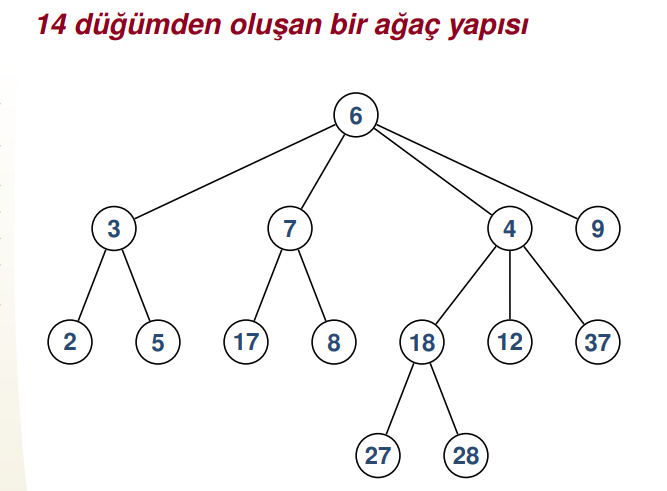
**Ağaç**



**Tanımlar**

● Agaçtaki x dügümünden y dügümüne bir ok varsa x dügümüne y dügümünün ebeveyni y dügümüne ise x’in çocugu denir.

● Bir dügümün çocuk sayısı o dügümün derecesini belirtir.

● Agaçtaki bir x dügümünden baslayıp çocukları üzerinden giderek bir y dügümüne ulasılabiliyorsa, y dügümüne x dügümünün soyu, x dügümüne de y dügümünün atası denir. ● Agaçtaki bir dü gümün hiç çocugu yoksa o dügüme yaprak dügüm denir.

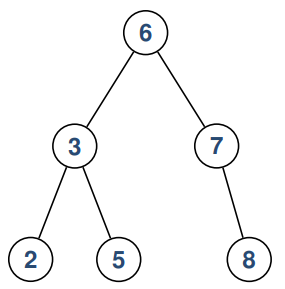
● Bir agacın derinligi o agacın kökünden herhangi bir yapraga ulasmak için geçtigimiz elemanların sayısının en fazlasıdır.

**Ikili Arama Agacı Tanımı**

Eğer bir ağaçtaki her düğümün en fazla iki çovugu varsa bu ağaca ikili arama ağacı denir.\*\* Bir ikili ağacın ikili artama ağacı olması için gerek ve yeter şart her düğümdeki değernin ,sol alt ağacında bulunan tüm düğümlerin değerlerinden büyük,sağ alt ağacında bulunan tüm düğümlerin değerlerinden küçük olmasıdır.

**Altı elemandan olusan bir ikili arama agacı**

Bu ağacın derinliği 2 dir.Dengeli bir ağaç soz konusu ie h derinliğindeki bir ağaçta 2h tane eleman olur.



**Tam sayılar içeren düğüm tanımı**

Public class Dugum{

Int icerik;

Dugum sol;

Dugum sağ;

Public Dugum(int icerik){

This.icerik=icerik;

Sol=null;

Sağ=null;

}

}

Tam sayılar içeren ikili arama ağacı tanımı

Public class Ağaç{

Dugum kok;

Public Agac(9{

Kok=null;

}

}

**Temel ikili Arama Ağacı İşlemleri**

**Ağaçta Eleman Arama**

Verilen bir degeri ikili arama agacında arayan özyinelemeli algoritma

Dugum agacAra(int eleman){

If(icerik==eleman)

Return this;

Else

If(icerik>eleman)

If(sol!=null)

Return sol.agacAra(eleman);

Else

Return null;

Else

If(sag!=null)

Return sag.agacAra(eleman);

Else

Return null;

}

Verilen bir degeri ikili arama agacında arayan yinelemesiz algoritma

1 Dugum agacAra(int eleman){

2 Dugum d;

3 d = kok;

4 while (d != null){

5 if (d. icerik == eleman)

6 return d;

7 else

8 if (d. icerik > eleman)

9 d = d.sol;

10 else

11 d = d.sag;

12 }

13 return null;

14 }

Arama agacındaki en küçük elemanı bulan yinelemesiz algoritma

1 Dugum asgariAra(){

2 Dugum sonuc = this;

3 while (sonuc.sol != null)

4 sonuc = sonuc.sol;

5 return sonuc;

6 }

Arama agacındaki en küçük elemanı bulan özyinelemeli algoritma

1 Dugum asgariAra(){

2 if (sol == null)

3 return this;

4 else

5 return sol.asgariAra();

6 }

Arama agacındaki en büyük elemanı bulan yinelemesiz algoritma

1 Dugum azamiAra(){

2 Dugum sonuc = this;

3 while (sonuc.sag != null)

4 sonuc = sonuc.sag;

5 return sonuc;

6 }

Arama agacındaki en büyük elemanı bulan özyinelemeli algoritma

1 Dugum azamiAra(){

2 if (sag == null)

3 return this;

4 else

5 return sag.azamiAra();

6 }

Arama agacına yeni bir eleman ekleyen algoritma

1 void agacaEkle(Dugum yeni){

2 Dugum y = null;

3 Dugum x = kok;

4 while (x != null){

5 y = x;

6 if (yeni. icerik < x. icerik )

7 x = x.sol;

8 else

9 x = x.sag;

10 }

11 if (y == null)

12 kok = yeni;

13 else

14 if (yeni. icerik < y. icerik )

15 y.sol = yeni;

16 else

17 y.sag = yeni;

18 }

Arama agacından bir eleman silen algoritma

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);

Else {

If(kok.sol!=null && kok.sag!=null){

Dugum maxsol=Enbuyuk(kok.sol);

Kok.icerik=maxSol.icerik;

Kok.sol=sil(kok.sol, maxSol.icerik)

}

Else if(kok.sag!=null){

Kok=kok.sag;

}else{

Kok=null;

}

Return kok;

}

Public void guncelleme(int eski.deger,int yeni){

agacSi(eski.deger);

Agacaekle(yeni.deger);

}

Gezintiler

Önce gezinti algoritması (ebeveyn,Sol çocuk,sağ çocuk : PreOrder)

Void onceGezinti(){

System.out.print(icerik);

If(sol!=null)

Sol.oncegezintı();

If(sag!=null)

Sag.onceGezinti();

\*\*inorder: sol çocuk,ebeveyn,sağ çocuk

\*\*postorder:sol çcuk,sağ çocuk,ebeveyn “