**VERİ YAPILARI DERS NOTLARI**

**BAĞLI LİSTELER:**

**Tek Yönlü Bağlı Listeler:**

-Bağlı Listelerde Düğüm Yapısı:

Bağlı Listelerde Düğüm sınıflarımızda o düğüm de tutacağımız Düğüm verisi ve bir sonraki düğüm adresini gösteren bir pointerımız olmalıdır.

-Bağlı Listede Arama Silme Ekleme işlemleri:

Bağlı listede Arama silme ve ekleme işlemlerini yapan fonksiyonlar içinde düğüm cinsinden geçici bir nesne oluştururuz ve istediğimiz düğüme ulaşmak için bu nesneyi kullanırız.

Ekleme fonksiyonu: Genel olarak bu fonksiyona parametre olarak eklemek istediğimiz düğüm verisi göndermemiz yeterlidir.

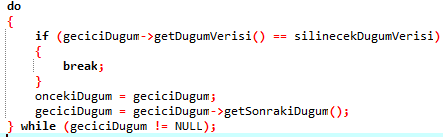
Oluşturacağımız geçici değişken ile son düğüme ulaşma işlemi



Şeklinde düğümlerde ilerleme sağlıyabiliriz. Oluşturduğumuz yeni düğümüde



Silme fonksiyonunda geçici nesneden dışında bir nesne daha oluştuturuz ve bu nesne geçici nesnenin bir önceki adresini bulundurur.



Şeklindeki işlemle aradığımız düğümü bulma işlemi tamamlanır. Daha sonra 

Şeklindeki işlemle bulunan düğümü silme işlemi yapılır.

Arama fonksiyonumuzuda yukarıdaki döngü ile halledebiliriz.

**Çift Yönlü Bağlı Liste:**

Çift yönlü bağlı listelerde düğüm yapısı: Tutacağımız düğüm verisi sonraki düğüm ve önceki düğüm adresleri olabilir.

Genel olarak tek yönlü bağlı liste ile arasında pek fark yoktur.

Arama fonksiyonda aynı döngü ile son düğüme gidip



İşlemleriyle düğüm bağlantılarını kurabiliriz.

Silme fonksiyonunda herhangi bir değişiklik yapmamıza gerek yoktur.

**STACK**

**Static Dizi İle Stack Yapısı:**

Stack yapımızda elemansayımızı ve verileri depolayacağımız bir dizi olmalıdır.

Push fonksiyonumuz 

Pop fonksiyomuz

Şeklinde olabilir.

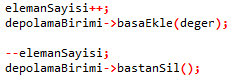
**Dinamik Dizi ile Stack Yapısı:**

Static diziden tek farkı diziyi heap bölgesinde oluşturmamızdır. İşlemler aynıdır.

**Düğüm ile Stack Yapısı:**

Bunun için oluşturduğumuz düğüm sınıfını kullanabiliriz. Tek ihtiyacımız olan basaEkle ve bastanSil fonksiyonlarını eklememiz olacaktır.

Stack sınıfımızda depolamaBirimi mizin türü Bağıl liste türünde olmalıdır ve ekleme silme işlemlerinde eleman sayısında oynama yaparak basaEkle ve bastanSil fonskiyonlarıyla işimizi hallederiz.



İlki ekleme ikinci silme işlemi içindir.

**İKİLİ ARAMA AĞACI**

İkili Arama Ağaçlarının genel yapısı şu şekildedir;

Ağacımızı bir düzene göre oluştururuz. İkili arama ağaçlarında eklenecek düğüm verisi karşılaştırma yaptığımız düğüm verisinden küçükse düğümün soluna büyükse düğümün sağına yeni düğüm olarak ekleriz.

İkili arama ağaçlarında düğüm yapımız o düğümde tutacağımız veri ve sağ ve sol düğümleri gösterecek birer pointerımız olması yeterlidir.



İkili arama ağacımızda ise değişken olarak başdüğüm adresini tutmamız yeterlidir.



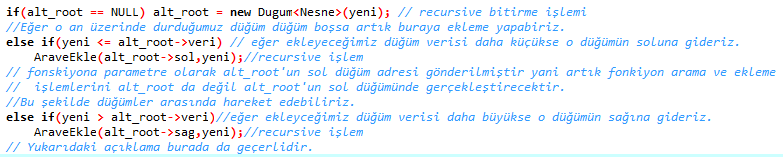
Diğer işlemlerimizi yapabileceğimiz fonksiyonlarda bu sınıfta yer almalıdır.

Fonksiyonlarmızın genel yapısı recursive dir. Bu fonksiyonlarda recursive olmasının mantığı ise bağlı listedeki temel while döngüsü mantığıyla aynıdır. Yani istediğimiz düğüme ulaşana kadar düğümlerde gezmemiz bu işlemiz tek işlemle halletmemiz zordur bunun için bir döngü ya da recursive bir işlem kullanabiliriz.

**Ekleme Fonksiyonu:**

Bu fonskiyonun mantığı genel olarak şöyledir;

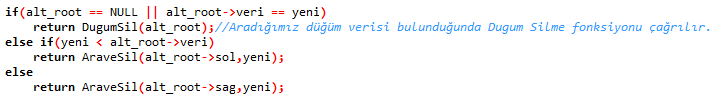
O an üzerinde durduğumuz düğümün verisi ile ekleyeceğimiz düğüm verisi karşılaştırılır. Eğer ekleyeceğimiz düğüm verisi daha küçükse üzerinde durduğumuz düğümü sol düğümü olarak ağaca eklenir eğer büyükse sağ düğüm olarak ağaca eklenir.



**Arama ve Silme Fonksiyonu:**

Bu fonskiyonun genel mantığı şu şekildedir;

Yine bi üsteki gibi o an üzerinde durduğumuz düğüm verisiyle sileceğimiz düğüm verisin karşılaştırırız ve o düğümün sağına mı soluna mı gideceğimizi belirleriz. Düğümlerde sağ veya sola gitme işlemlerini yine aynı recursive yapı ile halledebiliriz ve sileceğimiz düğüm verisini bulduğumuzda silme fonskiyonunu çağırırız.

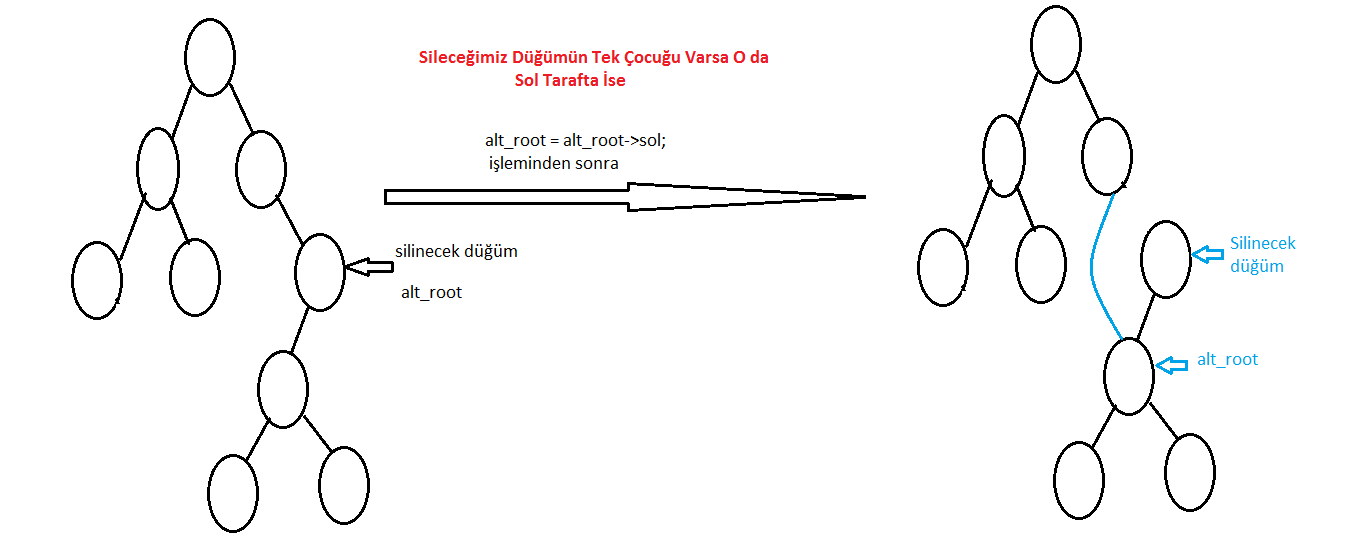


Düğüm silme Fonksiyonumuzun mantığı

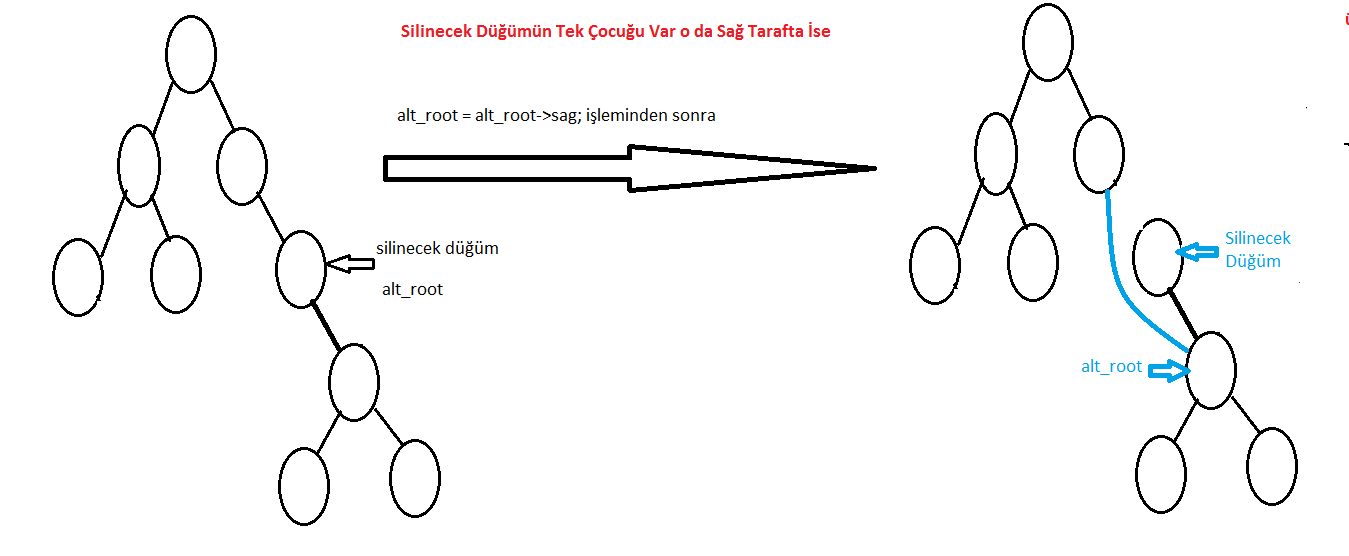
Düğüm silerken üzerinde durmamız gereken temel üç konu vardır.

1. Silinecek düğümün sadece sol çocuğu mu var? Yapmamız gereken işlem.

**Şekil 1**

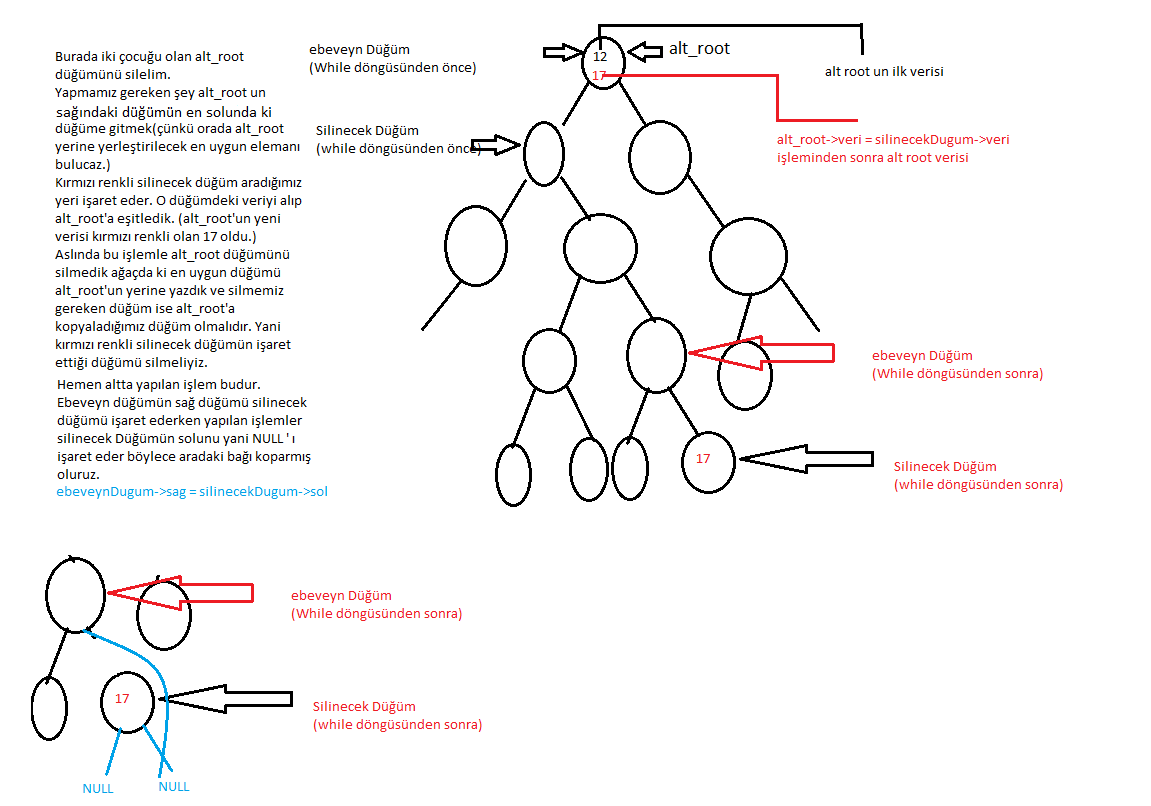


1. Silinecek düğümün sadece sağ çocuğu mu var? Yapmamız gereken işlem.

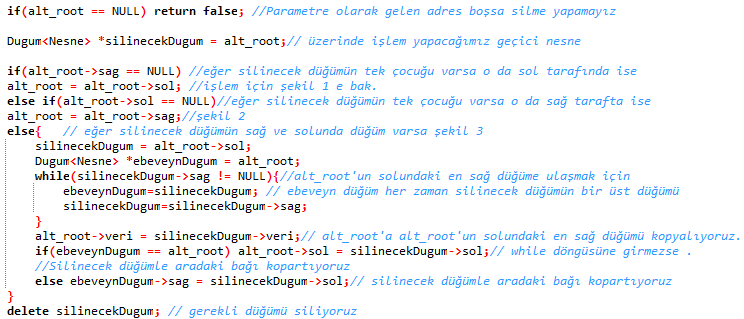
**Şekil 2**

1. Silinecek Düğümün 2 çocuğu da var mı? Yapmamız gereken işlem.

**Şekil 3**

****

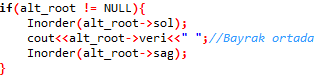
İşlemlerin kodu



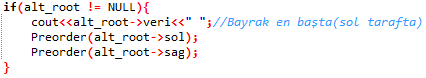
Yazdırma Metodları:

3 tane yazdırma metodumuz vardır:

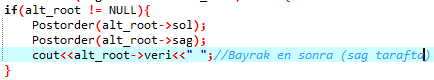
İnorder;



Preorder;



Postorder;



**HEAP AĞACI**

Heap ağaçları iki türlüdür.

Minimum heap: elemanları yukardan aşağı doğru gittikçe daima büyür. Yani en küçük eleman en baştadır ve elemanlar değerleri alt düğümlere indikçe artar.

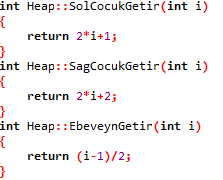
Maximum heap: bu heap ağacı ise minumum heap ‘ ın tam tersidir.

Heap Ağaçlarında düğümlerden ziyade dizilerle uğraşırız. Veriler arasındaki bağlantıları dizilerin indeksleri üzerinden sağlarız. Örn; bir verinin dizideki yerine i indeksi dersek sağ çocuk indeksi 2\*i+2 , sol çocuk indeksi 2\*i+1 , ebeveyn indeksi (i-1)/2 olur.

Heap Ağacı sınıfımızda;

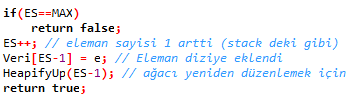
Değişken olarak dizideki elemanların yerlerini kontrol edip ekleme çıkarma yapmak için eleman sayisi ve verileri depolayacağımız bir dizi değişkeni oluşturmamız yeterlidir.

Bunun haricinde yardımcı fonksiyon olarak;

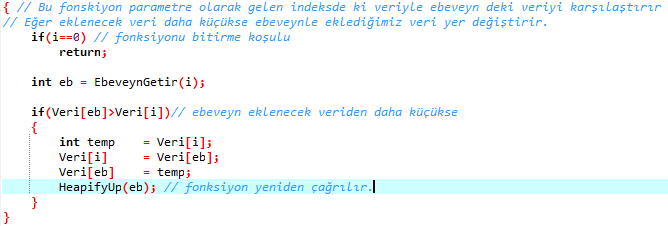
 fonksiyonlarını kullanıcağız.

Ekleme Fonksiyonu:

Ekleme fonksiyonunda yapmamız gereken veriyi dizinin sonuna direk olarak eklemek ve yukarı heap fonksiyonun çağırıp ağacı düzenlemek olacaktır. Çünkü Minimum heap de aşağı doğru indikçe veriler daima artıyordu ekliyeceğimiz veri küçük değere sahip olabilir bu yüzden ağacımızın yapısı bozulabilir. Ağacımızın yapısını yeniden düzenlemek için de yukarıya heap fonksiyonunu çağrırız.

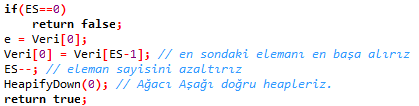


HeapifyUp fonksiyonu recursive bir fonksiyondur. Yaptığı işlem gönderilen parametreki veri ile ebeyninin verisini karşılaştırır eğer ebeveyn daha büyükse yer değiştitirler(Minimum heap a göre). Bu işlemler bitirme koşulları sağlanana kadar devam eder.



Çıkarma Fonksiyonu:

Bu fonskiyonda yapmamız gereken şey çıkarıcağımız eleman her zaman dizinin başındaki elemandır. Burada bir sorun yok fakat agacımızın kökünü çıkardık böyle bir durum olmamalı. Bunun için geçici olarak dizinin son elemanını dizinin baş elemanı yapıyoruz. Fakat alt tarakdaki düğüm verileri üst taraftaki düğüm verilerinden daha büyük olduğu için bu sefer ağaç yapımız bozuldu. Ağacımızı yeniden düzenlememiz gerekli bunun için Aşağı doğru heap fonksiyonunu kullanırız.



HeapifyDown Fonksiyonu:

Bu fonskiyona yine parametre olarak bir dizi indeksi göndeririz. 