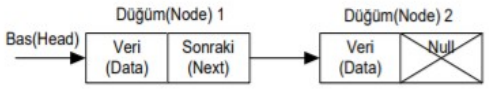
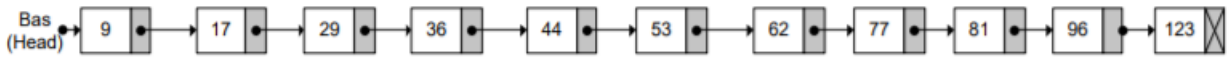
1. Skip-list veri yapısını açıklayın, bu veri yapısı üzerinde tanımlı operatörlerin çalışma şeklini basit birer örnek ile gösterin ve açıklayın. Skip-list'in düz listelere göre avantajını açıklayın.

Liste, aralarında ilişkili olan veriler topluluğu olarak düşünülebilir. Veri yapılarında değişik biçimlerde listeler kullanılmakta ve üzerlerinde değişik işlemler yapılmaktadır. Listeleme veri yapıları ile bilgisayarda belli verileri tutabilir, bu verilere istediğimiz zaman erişebilir, değiştirebilir ya da silebiliriz. Mevcut listeye yeni veri ekleyebiliriz. Skip-list veri yapısı bunlardan biridir. Bu veri yapısında bağlı listeler kullanılır.

Bağlı listeler bir grup düğümün (verinin) sıralı bir şekilde bağlanmasıyla oluşur. Listenin baş kısmına head uç kısmına tail adı verilir. Her bir düğüm veri ve bir sonraki verinin adresini içeren pointer ları tutar.

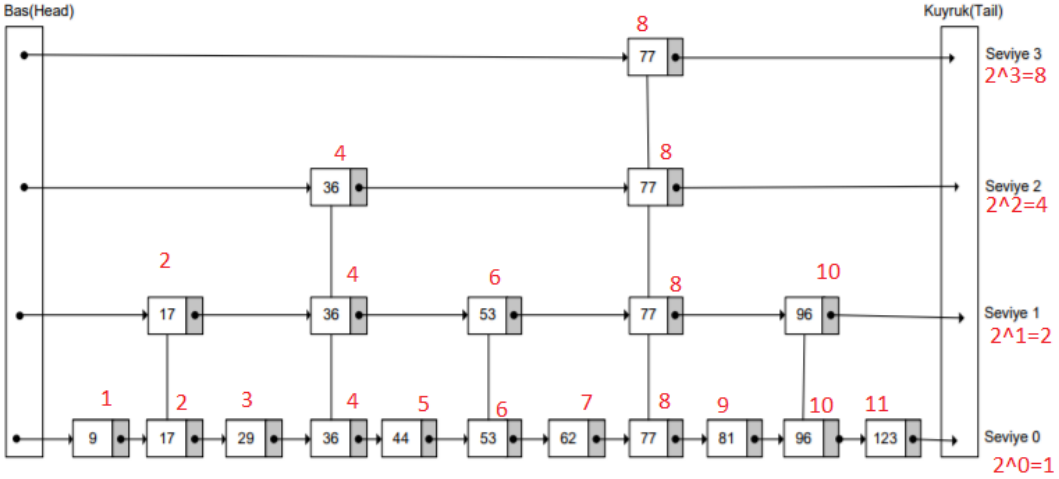


Şekil 1. Bağlı Liste düğüm Yapısı



Şekil 2. Bağlı Liste

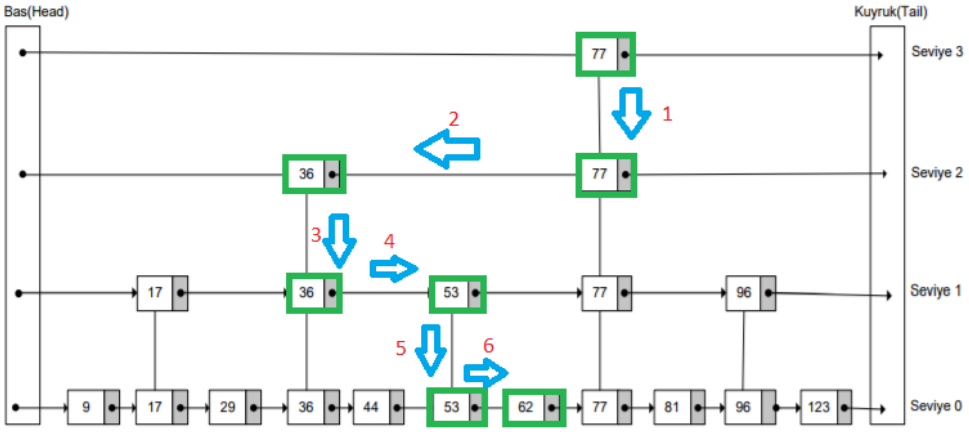
Skip list veri yapısında ise bağlı liste elemanları sıralı olarak piramit şeklinde farklı katmanlara yerleştirilerek arama işlemlerinde kolaylık sağlanması amaçlanır. Bu piramitteki her bir liste üst katmandaki listeyi kapsar.



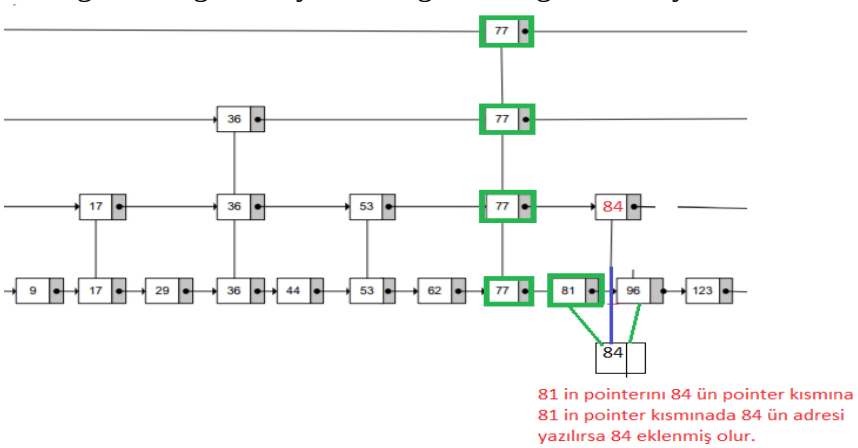
Şekil 4 - Skip List

İlk olarak tüm düğümler Seviye 0'da yer alır ve sol taraftan başlanarak her 2^i düğüm üste doğru her seviyeyi temsil eden işaretçiler oluşturulur.

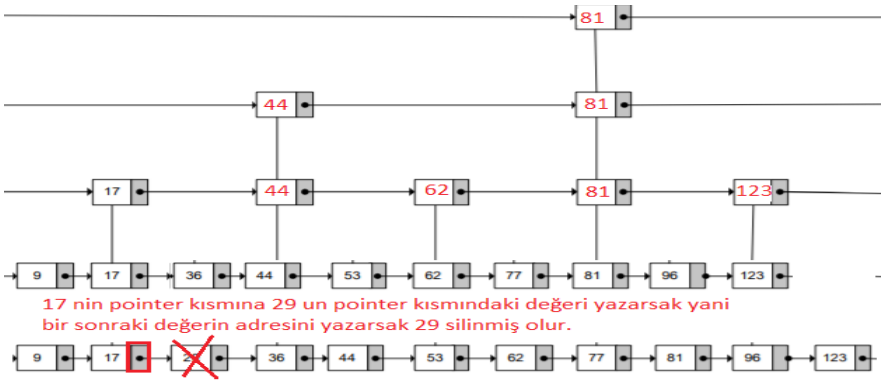
Arama operatörü kullanılacaksa aranacak sayı üst seviyeden başlanarak aşağı seviyelere doğru aranır. Eğer kendinden küçükse sola büyükse sağ a kaydırılarak aranır.
Örneğin 62 sayısını arayalım.



Ekleme, silme veya veriyi değiştirme operatörlerinde hemen hemen aynı işlemler yapılır.
Ekleme operatöründe istediğiniz değeri önce arıyor bulması gereken yere giderek kendinden önceki verinin pointerını değiştiriyor ve kendi pointerını bir sonraki veri için yazıyor daha sonra sırayla üst seviyelerin değerlerini güncelliyor. Örneğin 84 değerini ekleyelim.



Silme operatöründe istediğiniz değeri arayıp kendinden önceki değerin pointerına kendi pointerını giriyor yani bir sonraki veriye atlamış oluyor böylelikle silinmiş oluyor sonra üst seviyedeki değerleri güncelliyor. **Örneğin 29 değerini silelim.**



Değiştirme operatöründe ise değişecek değere silme operatörü yeni değere ekleme operatörü uygulanır.

Düz listeye göre avantajı: Skip-List aslında veri aramayı kolaylaştıran bir yapıdır. N elemanlı normal listede zaman N kadar iken skip-list yapısında $\log_2 N$ dir. Fakat ekleme ve silme işlemlerinde pratiklik sağlamayacaktır.

2. Elimizde myList adında bir soyut veri yapısı olduğunu varsayalım. Bu veri yapısı sıralı bir tamsayı listesi ve bu liste üzerinde yapılabilecek işlemlerden oluşmaktadır.
Liste üzerinde eleman ekleme, eleman çıkarma gibi işlemlerden başka, yapılacak işlemlerden bazıları şu şekildedir:

1. Verilen bir sayının listede kaç defa geçtiğini döndürme:

int countInList(myList list,int key)
sayac=0 olsun
listedeki her bir veriyi key le eşit olup olmadığını kontrol et.
Eğer eşitlik var sayac ı 1 arttır ve listedeki diğer değere geç
Eğer eşitlik yoksa diğer değere geç.
Sayacı döndür.

2. Listede birden fazla geçen ilk elemanı döndürme:

int firstRepeating(myList list)
n=0 olsun;
listedeki elamanlar for döngüsüne girsin.
listedeki n. elemanı al ve bir ifadeye ata ve tut x olsun.
listedeki n+1. elemanı al ve farklı bir ifadeye ata ve tut y olsun.
2 değeri kıyasla
Eğer eşitlik varsa x i döndür sonra for döngüsünden çık.
Eşitlik yoksa n sayısına 1 ekle for döngüsüne devam et.

3. Listedeki elemanları ters sırada yeni bir liste üzerinde döndürme:

int reverse(myList list, myList newList)
Listenin elaman sayısını bul ve n olsun.
Döngü: n!=0 ise döndür
N. elamanı döndür
n i 1 azalt