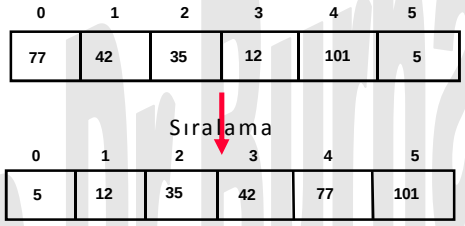
Sıralama Algoritmaları

6.1 Giriş

Bilgisayar uygulamalarında en sık rastlanan ve istenen verinin sıralı olmasıdır. En azından sıralama ikili arama için gereklidir. Sıralamadan kasıt, düzensiz olan bir veri yapısını küçükten büyüğe veya büyükten küçüğe doğru düzenli hale getirilmesidir. Şekil 6.1 sıralamanın anlamını şekil olarak sunmaktadır.



Şekil 6.1. Sıralama

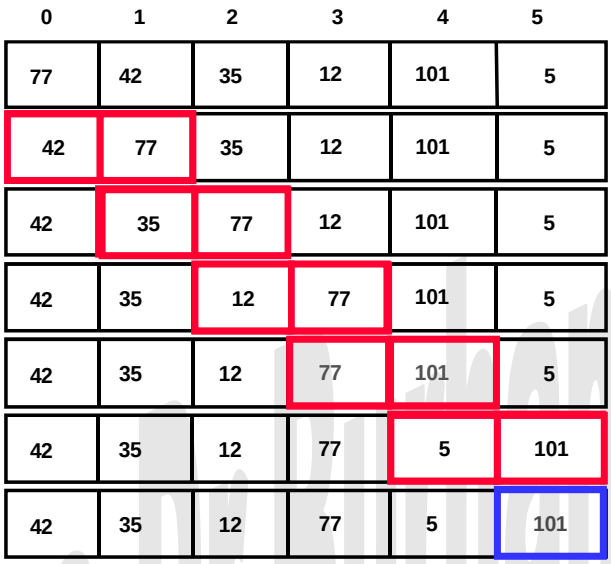
Sıralama için geliştirilmiş bir çok algoritma vardır. Bunlar temelde üç şekilde sınıflandırılabilir. Klasik yer değiştirme mantığı, ağaç oluşturma ile sıralama ve böl-yönet mantığına dayalı sıralama algoritmalarıdır. Klasik yer değiştirme mantığına baloncuk sıralaması (bubble sort), seçme sıralaması (selection sort) ve araya yerleştirme sıralaması (insertion sort) verilebilir. Ağaç oluşturma mantığına ise yığın sıralaması (heap sort) ve ikili ağaç sıralaması (binary tree sort) verilebilir. Böl ve yönet mantığına göre sıralamaya ise hızlı sıralama (quick sort) ve birleştirme sıralaması (merge sort) verilebilir.

6.2 Baloncuk Sıralaması

Sıralama dendiğinde genel kabul olarak verinin küçükten büyüğe doğru sıralanması olarak algılanır ve kabul edilir. Büyükten küçüğe sıralama da mümkün olmakla beraber akla ilk gelen küçükten büyüğe doğru sıralamadır. Bu neden bu algoritmanın ve diğer algoritmaların açıklanmasında sıralamada dendiğinde küçükten büyüğe doğru sıralama kasıt edilmektedir.

Her sıralamanın kendine özgü bir kabul ve çalışma mantığı vardır. Baloncuk sıralamasında temel düşünce veri sıralı kabul edilir. Eğer veri veya dizi sıralı ise bir önceki eleman bir sonrakinden küçük olacaktır. Eğer bir önceki eleman bir sonrakinden büyükse bir önceki ve bir sonraki eleman yer değiştirilerek düzenlenmeye çalışılır. Eğer bir önceki eleman bir sonrakinden küçükse durum istenen yapıdadır ve her hangi bir değişikliğe gerek yoktur. Daha sonra bir adım ilerlenerek ardışık iki veri arasında düzenlilik tekrar aranır. Düzenli değilse yer değişikliği yapılarak düzenlenir, düzenli ise bir sonraki ardışık iki veriye bakılır. İşlem kontrol edilmemiş tek bir eleman kalmayıncaya kadar devam eder.

Bu mantığın dizi boyunca bir defa yapılması sonucunda en büyük eleman aktarıla aktarıla en sona taşınır. Aktarılma şekli akvaryumdaki gibi bir sıvıdaki baloncukların yükselmesine benzetildiği için baloncuk sıralaması (bubble sort) denir. En büyük en sona taşındığından incelenmesi gereken aralık artık bir azalmıştır. Dolayısı ile bu sıralama algoritması (aslında diğer sıralama algoritmaları da) iki döngü ile gerçeklenir. Birinci (dış) döngü düzenlenen parçayı düzenlenmeyen parçadan ayırmak için kullanılırken, ikinci (iç) döngü düzensizliği bulmak için kullanılır. Şekil 6.2 iç döngünün bir defa çalışması ile en büyük elemanın en sona getirilmesi verilmiştir.



Şekil 6.2. İç döngü ile en büyün en sona gelmesi.

Genelde çift for döngülü olarak isimlendirilen bu algoritma aşağıdaki gibi verilebilir.

```
public void Bubblesort(int nElems,int intArray[]) {
int i,j;
for (i=nElems-1; i>0; i--)
        for (j=0;j<i; j++){
            if(intArray[i]>intArray[i+1])        //Düzensizlik var mı?
                swap(i,i+1);                     //Düzenle
}
```

Bu algoritmadaki swap metodu iki veri alanındaki bilgilerin yer değiştirilmesi için kullanılan basit bir metot dur. Şöyle yazılabilir;

```
private void swap(int one, int two){
        int temp = intArray[one];
        intArray[one] = intArray[two];
        intArray[two] = temp;
}
```

Algoritmanın bu hali ile çalışmasında eğer dizi düzenli ise farkına varılmaz ve düzensiz olduğu düşünülen parça tükeninceye kadar karşılaştırmalara devam edilir. Düzeltilmiş biçiminde veri yapısı veya dizi kontrol edilir düzenli ise işlem tamamlanır. Eğer bir adet dahi düzensizlik varsa o düzensizlik düzeltilir ve yeni düzensizlikler olabileceği için yeniden kontrol edilir. Düzensizliğin olduğu ise bir boolean bir değişkenle bayrak gibi belirtilir. Yani düzensizlik bulunduğu anda bayrak sallanarak problem olduğu belirlenir. Bu mantığa ise bayraklı baloncuk sıralaması denilir. Java kodlaması ise aşağıdaki gibi verilebilir.

```
public void Bubblesort(int nElems,int intArray[N]) {
        int i;
        boolen Continue=true;                    //Bayrak
        while(continue){
                continue=false;
```

```
            for (i=0;(i < nElems-1);i++) {
                  if(intArray[i]>intArray[i+1]){     // Düzensizlik var mı?
                        swap(i,i+1);                 //Düzenle
                        continue=true;               // Bayrağı kaldır
                  }
      }
}
```

6.3 Seçme Sıralaması

Seçme sıralamasının mantığı ise adından da anlaşılacağı gibi belirli bir değerin seçilmesini gerektirir. Algoritmada dizideki veride en küçük değere sahip yapı bulunur ve en baştaki ile yer değiştirilir. Bir adım ilerlenerek geriye kalan listede aynı işlem tekrarlanır. Sıralamada seçilenler sıralı parçayı geriye kalanlar ise sırasız parçayı oluşturur. İşlem sırasız eleman kalmayıncaya kadar devam eder. Java kodlaması aşağıdaki gibi verilebilir.

```
public void selectionSort(int nElems, int intArray[]) {
      int i, j, min;
      for(i=0; i<nElems-1; i++) {          // Dış döngü
      min = i;                             // En küçüğün indisi
      for(j=i+1; j<nElems; j++)            // iç döngü
        if(intArray[j] < intArray[min] )   // En küçüğü bul
          min = j;
            swap(i, min);                  // yer değiş
      }
}
```

Bu sıralama mantığının bağlı liste yapısı ise aşağıda verilmiştir. Metod sırasız listeyi alıp sıralı listeyi geri döndürür. Metodun çalışma mantığı şöyledir. Sırasız listede en bul ve çıkar. Çıkarılan en büyük düğüm sıralı listenin başına ekle ve böylece sıralı liste küçükten büyüğe sıralı olur.

```
public node selectionSort(node Head) {
      node SortedList=null,
            maxbefore,
            max,
            p;
      if (Head==null||Head.next==null)                  //Boş liste ve tek düğüm sıralıdır.
            return Head
      else
            while (Head!=null){
                  maxbefore=null,
                  max=Head,
                  p=Head;
                  while (p.next!=null){                 //En büyüğü bul
                        if (max.elemen<p.next.element){
                              maxbefore=p;
                              max=p.next;
                        }
                  maxbefore.next=max.next;              //En büyüğü çıkar
                  max.next=SortedList;                  //Sıralı listeye ekle
                  SortedList=max;
```

```
        }
}
```

6.4 Araya Yerleştirme Sıralaması

Araya sıkıştırma sıralaması mantığında ise başlangıçtan itibaren diziyi sıralı ve sırasız olarak iki parça olarak kabul eder. Sırasız listeden bir eleman alarak sıralı listede uygun yere yerleştirerek sıralı hale getirir. Dizi üzerinde başlangıçta ilk elaman sıralı liste olarak kabul edilir. Çünkü tek eleman ne olursa olsun sıralıdır. Daha sonra ikinci elemandan itibaren sırasız liste başlar. Sırasız listenin ilk elemanı geçici olarak çıkarılır ve geriye doğru uygun yer aranır ve oraya yerleştirilir.

Bu mantığa göre sıralama algoritması aşağıda verilmiştir.

```
public void insertionSort(int nElems, int intArray[]){
    int i,j;
    int temp;
    for(i=1; i<nElems; i++){                    //Dış döngü, tek eleman sıralıdır.
        temp = intArray[i];                     // İşaretli elamanı listeden al
        j = i;                                  // Kaydırmaya başla
        while(j>0 && intArray[j-1] >temp) {     // Bir küçük oluncaya kadar,
            intArray[j] = intArray[j-1];        // Elemanları bir aşağı al
            --j;                                // Bir pozisyon geri gel
        }
        intArray[j] = temp;                     // İşaretli elmanı araya koy
    }
}
```

Araya sıkıştırma algoritmasını bağlı liste versiyonu ise aşağıda verilmiştir. Verilen bu algoritma yine sırasız listeyi alıp sıralı listeyi geri döndürür.
Mantığı ise şöyledir;

I. Gelen listenin (Head) ilk düğümünü çıkar ve tek düğümlü sıralı liste elde et.
II. Sırasız listeden düğüm çıkar sıralı listede uygun yere yerleştir. Çıkarılan düğüm sıralı listenin ilk düğüm olabilir, değilse yer ara. Bulunan yere düğümü yerleştir.
III. Sırasız liste tükeninceye kadar devam et.

Bu metod yukarıdaki adımlar doğrultusunda alt metodlara ayrılabilir.

```
public node insertionSort(node Head){
    node SortedList=null,
    r,
    p;
    if (Head==null||Head.next==null)                    //Boş liste ve tek düğüm sıralıdır.
        return Head
    else{
        SortedList=Head;
        SortedList.next=null;
        Head=Head.next;
        while (Head!=null){
            deleted=Head;
            Head=Head.next;
            İf (deleted.element<SortedList.element){
```

```
                        deleted.next=SortedList;
                        SortedList=deleted;
                    }else{
                        r=SortedList;
                        p=SortedListe.next;
                        while (p.next!=null&&p.element<deleted.element){
                            r=p;
                            p=p.next;
                        }
                        r.next=deleted;
                        deleted.next=p;
                }
        }
return SortedList;
}
```