Merge Sort
Bireşen Sıralama

Merge sort (bireşen sıralama), diziyi ardışık olarak en küçük alt dizilerine kadar yarılayan sonra da onları sıraya koyarak bireştiren özyineli bir algoritmadır. Yarılama işlemi en büyük alt dizi en çok iki öğeli olana kadar sürer. Sonra merge (bireşim) işlemiyle altdiziler ikişer ikişer bölünüş sırasıyla sıralı olarak bir üst dizide bireşir. Süreç sonunda en üstte sıralı diziye ulaşılır. Bu algoritma *O(nlogn)* grubuna ait etkin bir sıralama yöntemidir.

Bu algoritmanın nasıl çalıştığını açıklamak için şöyle bir örnek verelim. Bir salonda yapılacak törene çağrılan 100 kişinin adları yazılı birer etiket, koltuklara harf sırasıyla yapıştırılacaktır. Etiketler sırasız biçimde veriliyor. Etiketleri doğru sıraya dizmesi için görevliye önerebileceğimiz bir çok yöntem arasında en hızlı olanlardan birisi şudur.

A. Diziyi en çok 2 öğeli alt dizilere ulaşana kadar ortadan ikiye böl.
   a. Karışık etiketleri ortadan iki altdiziye böl. 50 şer etiketlik sırasız iki altdizi oluşur.
   b. 50 şer etiketlik iki alt dizinin her birini kendi içinde 25 şer etiketlik altdizilere böl. Sırasız dört altdizi oluşur.
   c. Aynı yöntemi uygulayarak, 25 şer etiketlik altdizilerin her birini 12 ve 13 etiketlik altdizilere böl.
   d. 12 etiketlik alt dizileri ikişer tane 6 etiketlik altdizilere; 13 etiketlik alt dizileri ise 6 ve 7 etiketlik altdizilere böl.
   e. 6 etiketlik alt dizileri ikişer tane 3 er etiketlik altdizilere, 7 etiketlik olanları ise 3, ve 4 etiketlik ikişer altdizilere böl.
   f. 3 er etiketlik alt dizileri 1 ve 2 etiketlik altdizilere, 4 etiketlik altdizileri ise 2 şer etiketlik altdizilere böl.

B. Bölünen alt dizileri, bölünüşün ters yönünde ikişer ikişer merge (bireşim) algoritması ile sıralı olarak bireştir.
   a. 1 ve 2 etiketlik altdizilere merge algoritmasını uygulayarak 3 etiketlik sıralı üstdizileri oluştur. Aynı yöntemle 2 etiketlik altdizileri bireştirerek 4 etiketlik sıralı üstdizileri oluştur.
   b. 3 ve 4 etiketlik dizilere merge algoritmasını uygula; 6 ve 7 etiketlik sıralı üstdizileri oluştur.
   c. 6 ve 7 etiketlik dizilere merge algoritmasını uygula; 12 ve 13 etiketlik sıralı üstdizileri oluştur.
   d. 12 ve 13 etiketlik dizilere merge algoritmasını uygula; 25 etiketlik sıralı üstdizileri oluştur.
   e. 25 etiketlik dizilere merge algoritmasını uygula; 50 etiketlik sıralı üstdiziyi oluştur.

Yukarıda söylenenleri bir yalancı kod (pseudo code) biçiminde yazalım. Her hangi bir adımdaki altdiziyi *a[sol…sağ]* simgesiyle gösterelim. *sol* altdizinin en küçük indisi, *sağ* ise altdizinin en

büyük indisi olsun. *sol < sağ* ise *m = (sol+sağ)/2 diyelim.* Bu bir tamsayı bölme işlemidir; yani *m* sayısı bölümün tamsayı kısmıdır.

A. Altdizilere bölme ve sıralama:
a. *a[sol…m]* altdizisini sırala;
b. *a[m+1…sağ]* altdizisini sırala;
c. *a[sol…m]* ile *a[m+1…sağ]* altdizilerini sıralı bireştir;
d. Bireşen diziyi verilen dizi üstüne kopyala;
e. Dur.

Tabii, burada bölme ve bireştirme işlemleri özyinelidir. Süreç bitene kadar bölme ve bireştirme işlemleri devam eder.

Şimdi de bunu bir örnek üzerinde şekille gösterelim. Basitliği sağlamak için 10 terimli bir bir dizi alalım.

```
String a = {"Yüce", "Ulusoy", "Aygün","Volkan","Çelik", "Can", "Alp", "Baydar", "Kazan", "Demir"};
```

String dizisi verilsin. Şimdi bu dizinin *merge sort algoritması* ile sıralanışını şekil üzerinde gösterelim.

Yüce	Ulusoy	Aygün	Volkan	Çelik	Can	Alp	Baydar	Kazan	Demir
Yüce	Ulusoy	Aygün	Volkan	Çelik	Can	Alp	Baydar	Kazan	Demir
Yüce	Ulusoy	Aygün	Volkan	Çelik	Can	Alp	Baydar	Kazan	Demir
Yüce	Ulusoy	Aygün	Volkan	Çelik	Can	Alp	Baydar	Kazan	Demir
Ulusoy	Yüce	Aygün	Çelik	Volkan	Alp	Can	Baydar	Demir	Kazan
Ulusoy	Yüce	Aygün	Çelik	Volkan	Alp	Can	Baydar	Demir	Kazan
Aygün	Çelik	Ulusoy	Volkan	Yüce	Alp	Baydar	Can	Demir	Kazan
Alp	Aygün	Baydar	Can	Çelik	Demir	Kazan	Ulusoy	Volkan	Yüce

Şimdi bu algoritmayı yapan özyineli (recursive) metotları yazalım. İstersek bölme ve bireştirme işlemlerini yapan kodları bir metotta bir araya getirebiliriz. Ama, daha anlaşılır olması için, bölme ve bireştirme işlemlerini ayrı ayrı metotlarla yapabiliriz. Zaten, bireştirme algoritmasını daha önceden biliyoruz.

Aşağıdaki özyineli metot, diziyi en küçük alt dizilerine ayırır ve özyineli merge metodunu çağırarak sıralı bireşim işlemini yaptırır.

```
void mergesort(int alt, int üst)
{
    if (alt<üst)
    {
        int m=(alt+üst)/2;
        mergesort(alt, m);
        mergesort(m+1, üst);
        merge(alt, m, üst);
    }
}
```

Anımsayınız. Aşağıdaki merge metodu şu işleri yapar:

1. Verilen *a[]* dizisinin öğelerini yedek *b[]* dizisine kopyala
2. i <= m ve j <= üst olduğunda a[k] = min{a[i], b[j]}

3. Geri kalan altdizinin bütün öğelerini *a[]* ya ekle

```
void merge(int alt, int m, int üst)
{
    int i, j, k;

        // a dizisinin her iki yarısını yedek b dizinine kopyala
    for (i=alt; i<=üst; i++)
        b[i]=a[i];

    i=alt; j=m+1; k=alt;
       // her adımda bir sonraki en büyük terimi kopyala
    while (i<=m && j<=üst)
        if (b[i]<=b[j])
            a[k++]=b[i++];
        else
            a[k++]=b[j++];
       // varsa, ilk yarıdan arta kalan terimlerin hepsini kopyala
    while (i<=m)
        a[k++]=b[i++];
}
```

```
//package mergesort;

//import java.util.Arrays.*;

public class MergeSort01 {

      static int a[] = { 12, 9, 4, 99, 120, 1, 3, 10 };

      public static void main(String[] args) {
            mergeSort(0, 7);
            diziYaz(a);
      }

      static void mergeSort(int alt, int üst) {
            if (alt < üst) {
                  int m = (alt + üst) / 2;
                  mergeSort(alt, m);
                  mergeSort(m + 1, üst);
                  merge(alt, m, üst);
            }
      }

      static void merge(int alt, int m, int üst) {
            int[] b = new int[8];
            int i, j, k;
                  // a dizisinin her iki yarısını yedek b dizinine kopyala
            for (i = alt; i <= üst; i++) {
                  b[i] = a[i];
            }

            i = alt;
            j = m + 1;
            k = alt;
                  // her adımda bir sonraki en büyük terimi kopyala
            while (i <= m && j <= üst) {
                  if (b[i] <= b[j]) {
```

```
                        a[k++] = b[i++];
                  } else {
                        a[k++] = b[j++];
                  }
            }

          // varsa, ilk yarıdan arta kalan terimlerin hepsini kopyala
            while (i <= m) {
                  a[k++] = b[i++];
            }
      }

      static void diziYaz(int[] arr) {
            for (int i = 0; i < arr.length; i++) {
                  System.out.print(arr[i] + "  ");
            }
      }
}
```

İstenirse, yukarıdaki programda *static* nitelemeli *array* ile *metotlar* dinamik yapılabilir. Aşağıdaki program o işi yapmaktadır.

```
//package mergesort;

//import java.util.Arrays;

public class MergeSort01 {

      int a[] = { 12, 9, 4, 99, 120, 1, 3, 10 };

      public static void main(String[] args) {
            MergeSort01 ms = new MergeSort01();
            ms.mergeSort(0, 7);
            ms.diziYaz(ms.a);
      }

      void mergeSort(int alt, int üst) {
            if (alt < üst) {
                  int m = (alt + üst) / 2;
                  mergeSort(alt, m);
                  mergeSort(m + 1, üst);
                  merge(alt, m, üst);
            }
      }

      void merge(int alt, int m, int üst) {
            int[] b = new int[8];
            int i, j, k;
            // a dizisinin her iki yarısını yedek b dizinine kopyala
            for (i = alt; i <= üst; i++) {
                  b[i] = a[i];
            }

            i = alt;
            j = m + 1;
            k = alt;
            // her adımda bir sonraki en büyük terimi kopyala
            while (i <= m && j <= üst) {
                  if (b[i] <= b[j]) {
```

```
                        a[k++] = b[i++];
                  } else {
                        a[k++] = b[j++];
                  }
            }

          // varsa, ilk yarıdan arta kalan terimlerin hepsini kopyala
            while (i <= m) {
                  a[k++] = b[i++];
            }
      }

      void diziYaz(int[] arr) {
            for (int i = 0; i < arr.length; i++) {
                  System.out.print(arr[i] + "  ");
            }
      }
}
```

Bölümün başında açıkladığımız String arrayini sıralayan program.

```
//package mergesort;

//import java.util.Arrays;

public class MergeSort01 {

      String[] a = {"Yüce", "Ulusoy", "Aygün","Volkan","Ç elik", "Can",
"Alp", "Baydar", "Kazan", "Demir"};

      public static void main(String[] args) {
            MergeSort01 ms = new MergeSort01();
            ms.mergeSort(0, 9);
            ms.diziYaz(ms.a);
      }

      void mergeSort(int alt, int üst) {
            if (alt < üst) {
                  int m = (alt + üst) / 2;
                  mergeSort(alt, m);
                  mergeSort(m + 1, üst);
                  merge(alt, m, üst);
            }
      }

      void merge(int alt, int m, int üst) {
            String[] b = new String[10];
            int i, j, k;
            // a dizisinin her iki yarısını yedek b dizinine kopyala
            for (i = alt; i <= üst; i++) {
                  b[i] = a[i];
            }

            i = alt;
            j = m + 1;
            k = alt;
            // her adımda bir sonraki en büyük terimi kopyala
            while (i <= m && j <= üst) {
                  int r = b[i].compareTo(b[j]);
```

```
                if (r < 0)
                        {
                        a[k++] = b[i++];
                } else {
                        a[k++] = b[j++];
                }
        }

        // varsa, ilk yarıdan arta kalan terimlerin hepsini kopyala
        while (i <= m) {
                a[k++] = b[i++];
        }
    }

    void diziYaz(String[] arr) {
        for (int i = 0; i < arr.length; i++) {
                System.out.print(arr[i] + "  ");
        }
    }
}
```