SAKARYA
UYGULAMALI BİLİMLER
ÜNİVERSİTESİ

İŞLETİM SİSTEMLERİ

Proje

Teslim Tarihi:

18.05.2023

B200109041 – Emircan BAĞDU

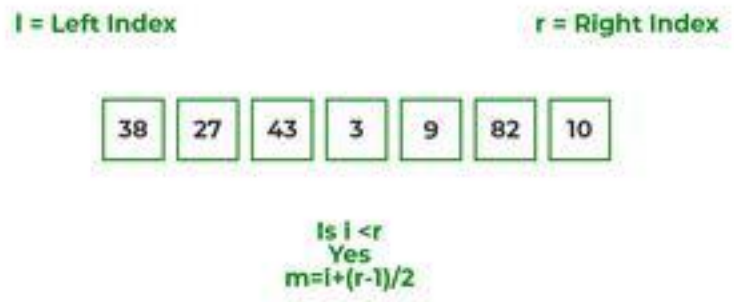
-- Multi-threading İle Merge Short—

Merge Sort Nedir ?

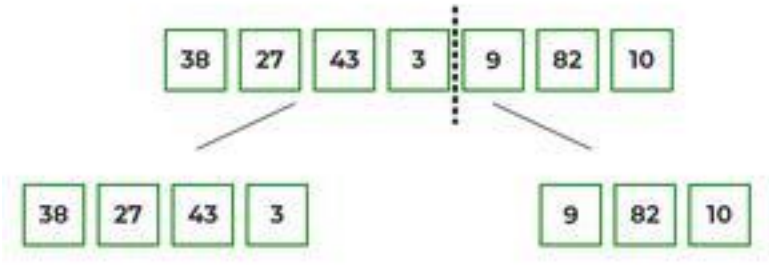
Bir diziyi küçük alt dizilere bölerek , her bir alt diziyi sıralı bir şekilde birleştirilmesi işlemidir.

Örnek olarak {38, 27, 43, 3, 9, 82, 10} dizisi için yapımı böyledir.

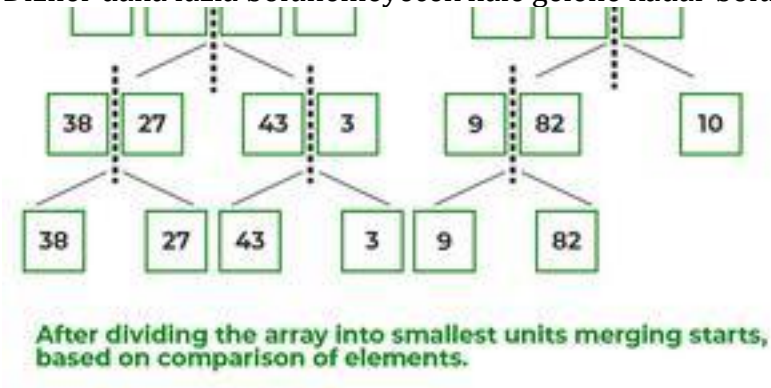
- İlk olarak sol dizinin sağ diziden küçük olup olmadığı kontrol edilir. Sonra orta nokta belirlenir.



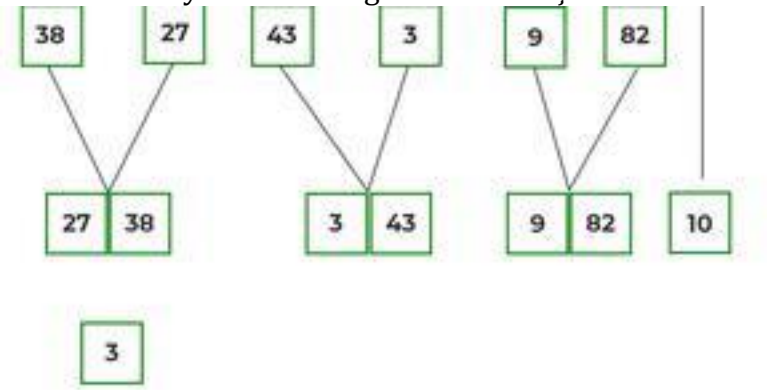
- Diziyi 4 sol ve 3 sağ olacak şekilde bölünür.



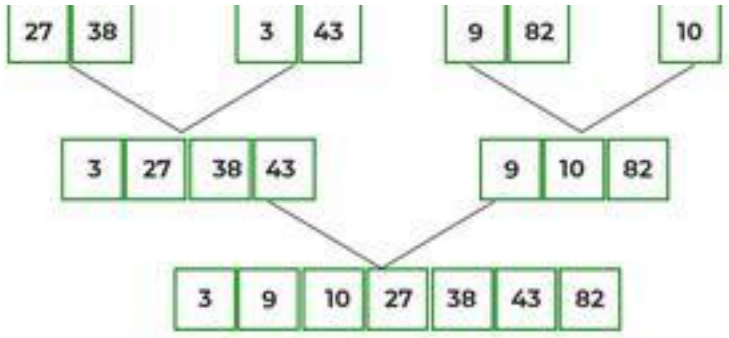
- Diziler daha fazla bölünemeyecek hale gelene kadar bölünme devam eder.



- Elemanlar büyüklüklerine göre sıralı bir şekilde listede birleşirler



- Listenin son hali bu şekilde olur



Multi Thread Nedir ?

Türkçesi çok iş parçacıklı demektir. Bir yazılım dilinde aynı anda birden fazla işlemin birlikte yapılması demektir.

Multi-threading İle Merge Short Amacı

Merge Sort algoritmasını çoklu iş parçacıklar (multi-threading) kullanarak uygulamaktır. Multi Thread kullanılarak sıralama süresi kısalır ve performans artar.
Bu yazdığım kod, Merge Sort algoritmasını uygularken paralel hesaplama için Multi Thread kullanır. Multi Thread ile diziyi daha küçük parçalara bölerek her bir parçayı ayrı bir iş parçacığına atar. Bu parçalar, bağımsız olarak sıralanır. Ardından, iş parçacıkları tamamlanınca sıralanmış alt diziler birleştirilir ve tam sıralanmış bir dizi elde edilir.

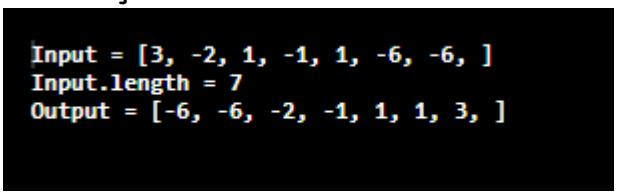
```
//SINIFLAR
import java.lang.System;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Random;

class MergeSort{
```

```
// THREADS değerini belirleme (iş parçacığı)
          private static final int MaxThread = 4;

          private static class SortThreads extends Thread{
                    SortThreads(Integer[] array, int begin, int end){//SortThreads SINIFI array begin end
parametrelerini alıyor
                              super(()->{
                                        MergeSort.mergeSort(array, begin, end);
                              });
                              this.start();
                    }
          }

          public static void threadedSort(Integer[] array){

                    // Dizinin uzunluğu
                    final int Uzunluk = array.length;
        // Eşit parçalara bölme kontrolü için bool değeri hesaplanır
                    boolean Esit =Uzunluk%MaxThread == 0;
                    // Her iş parçacığına düşecek eleman sayısı hesaplanır
                    int maxlim = Esit? Uzunluk/MaxThread: Uzunluk/(MaxThread-1);

                    maxlim = maxlim < MaxThread? MaxThread : maxlim;

                    final ArrayList<SortThreads> threads = new ArrayList<>();
                    // Dizi parçaları oluşturulur ve her biri için iş parçacığı oluşturulur
                    for(int i=0; i < Uzunluk; i+=maxlim){
                              int beg = i;
                              int remain = (Uzunluk)-i;
                              int end = remain < maxlim? i+(remain-1): i+(maxlim-1);
                              final SortThreads t = new SortThreads(array, beg, end);

                              threads.add(t);
                    }
                    // Tüm iş parçacıklarının tamamlanmasını bekler
                    for(Thread t: threads){
                              try{
                                        t.join();
                              } catch(InterruptedException ignored){}
                    }
                    // İş parçacıkları tarafından sıralanan alt diziler birleştirilir
                    for(int i=0; i < Uzunluk; i+=maxlim){
                              int mid = i == 0? 0 : i-1;
                              int remain = (Uzunluk)-i;
                              int end = remain < maxlim? i+(remain-1): i+(maxlim-1);

                              merge(array, 0, mid, end);
                    }

          }
// Merge sort algoritması
          public static void mergeSort(Integer[] array, int begin, int end){
              // Dizi parçalarının başlangıç ve bitiş indisleri arasında en az 2 eleman olduğu sürece devam
eder
                    if (begin<end){
                        // Dizinin orta noktası bulunur
                              int mid = (begin+end)/2;
                              // İlk yarısını sıralamak için mergeSort çağrılır
                              mergeSort(array, begin, mid);
                              // İlk yarısını sıralamak için mergeSort çağrılır
                              mergeSort(array, mid+1, end);
                              // İki sıralı alt dizi birleştirilir
                              merge(array, begin, mid, end);
                    }
          }
```

```
          public static void merge(Integer[] array, int begin, int mid, int end){
              // Geçici bir dizi oluşturulur. Bu dizi, birleştirme işlemi sırasında kullanılacak.
                    Integer[] temp = new Integer[(end-begin)+1];

                    int i = begin, j = mid+1;
                    int k = 0;
                    // İki alt dizi arasındaki elemanları karşılaştırarak geçici diziye yerleştirir.

                    while(i<=mid && j<=end){
                              if (array[i] <= array[j]){
                                        temp[k] = array[i];
                                        i+=1;
                              }else{
                                        temp[k] = array[j];
                                        j+=1;
                              }
                              k+=1;
                    }
          // İlk alt diziye kalan elemanları geçici diziye ekler.
                    while(i<=mid){
                              temp[k] = array[i];
                              i+=1; k+=1;
                    }
                    // İkinci alt diziye kalan elemanları geçici diziye ekler.
                    while(j<=end){
                              temp[k] = array[j];
                              j+=1; k+=1;
                    }
          // Geçici dizideki elemanları asıl diziye geri kopyalar
                    for(i=begin, k=0; i<=end; i++,k++){
                              array[i] = temp[k];
                    }
          }
}

public class Driver{
          // Rastgele sayılar üretmek için Random sınıfı kullanılır
          private static Random random = new Random();
          // Dizinin boyutu rastgele olarak belirlenir
          private static final int size = random.nextInt(100);
          // Boyutu belirlenen rastgele sayılarla doldurulmuş bir Integer dizisi oluşturulur
          private static final Integer liste[] = new Integer[size];

          static {
          for(int i=0; i<size; i++){
          // liste dizisi rastgele sayılarla doldurulur
                    liste[i] = random.nextInt(size+(size-1))-(size-1);
          }
          }

          public static void main(String[] args){
              // Girdi dizisinin değerleri ekrana yazdırılır
          System.out.print("Input = [");
          for (Integer each: liste)
                    System.out.print(each+", ");
          System.out.print("] \n" +"Input.length = " + liste.length + '\n');
// Beklenen çıktıyı elde etmek için girdi dizisi kopyalanır
          Integer[] dizi1 = Arrays.copyOf(liste, liste.length);
// Arrays.sort yöntemi kullanılarak girdi dizisi standart Java sıralama yöntemiyle sıralanır
          Arrays.sort(dizi1, (a,b)->a>b? 1: a==b? 0: -1);

// Merge Sort algoritması kullanılarak girdi dizisi sıralanır
          Integer[] dizi2 = Arrays.copyOf(liste, liste.length);

          MergeSort.mergeSort(dizi2, 0, dizi2.length-1);
// Çoklu iş parçacığı kullanarak Merge Sort algoritmasıyla girdi dizisi sıralanır
```

```
        Integer[] dizi = Arrays.copyOf(liste, liste.length);
        MergeSort.threadedSort(dizi);
        // Çıktı dizisi ekrana yazdırılır
        System.out.print("Output = [");
        for (Integer each: dizi)
                System.out.print(each+", ");
        System.out.print("]\n");
        }
}
```

Ekran Çıktısı :



Kaynakça :

https://www.geeksforgeeks.org/merge-sort/

https://erdincuzun.com/ileri-python/multithreading-programlama/#:~:text=Multithreading%2C%20%C3%A7o%C4%9Fu%20yaz%C4%B1l%C4%B1m%20dili%20taraf%C4%B1ndan,kod%20par%C3%A7as%C4%B1n%C4%B1n%20da%20%C3%A7al%C4%B1%C5%9Fmas%C4%B1%20demektir.