**KINGDOM OF SAUDI ARABIA**

**Ministry of Education**

**Taibah University**

**College of Computer Science & Engineering**

---

**CS424 – Introduction to Parallel Computing**

**Semester II 2019/2020**

---

# Merge Sort Algorithm

**Submit By:**

**Basem Awwad Al-Moziny**
**No. 3706683**

**Hamed Abed Karim**
**No. 3611356**

**Yasser Faisal Al-Muhammadi**
**No. 3692722**

## 1. Description of the problem and the sequential solution

Given an array arr[] of size n, its prefix sum array is another array prefixSum[] of same size such that the value of prefixSum[i] is :
arr[0] + arr[1] + arr[2] … arr[i].

- **Example :**

```
Input  : arr[] = {10, 20, 10, 5, 15}
Output : prefixSum[] = {10, 30, 40, 45, 60}

Explanation : While traversing the array, update
the element by adding it with its previous element.
prefixSum[0] = 10,
prefixSum[1] = prefixSum[0] + arr[1] = 30,
prefixSum[2] = prefixSum[1] + arr[2] = 40 and so on.
```

- **Java Code :**

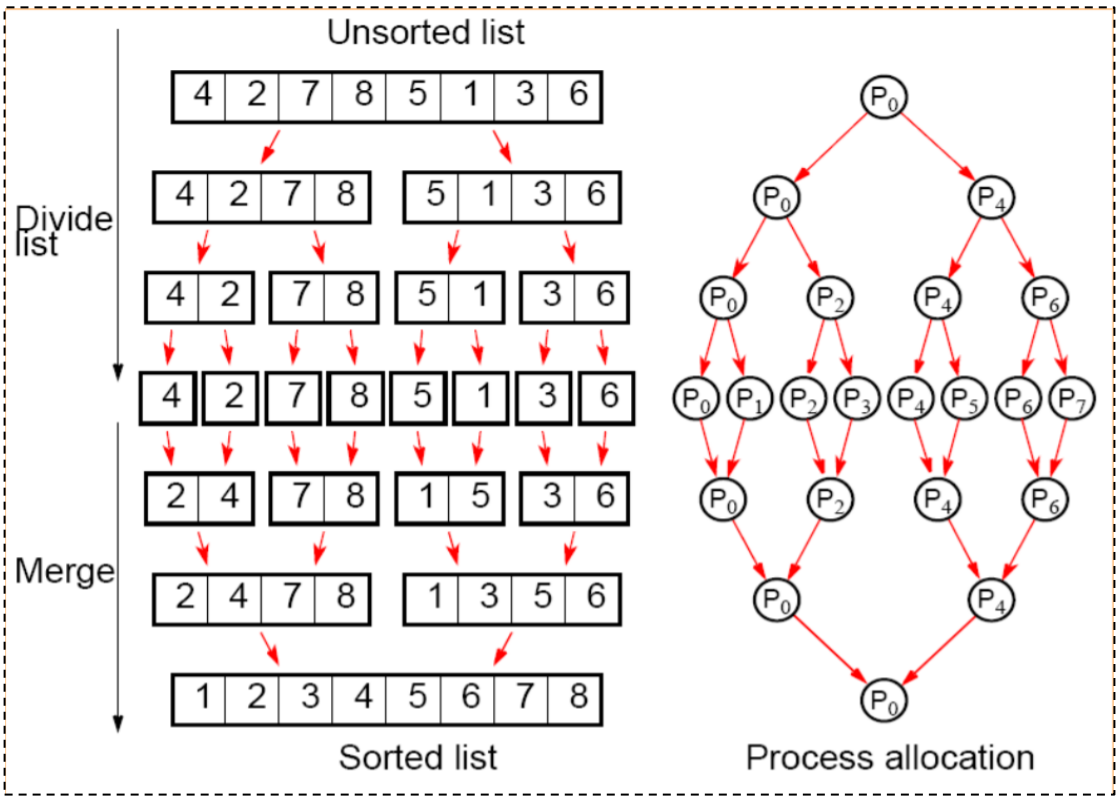
```
// Java Program for Implementing
// prefix sum arrayclass

class Prefix
{
   // Fills prefix sum array
   static void fillPrefixSum(int arr[], int n,
   int prefixSum[])
  {
    prefixSum[0] = arr[0];
    // Adding present element
    // with previous element
    for( int i = 1; i < n; ++i )
    prefixSum[i] = prefixSum[i-1] + arr[i];
  }
```

```
        // Driver code
        public static void main(String[] args)
        {
            int arr[] = { 10, 4, 16, 20 };
            int n = arr.length;
            int prefixSum[] = new int[n];

            fillPrefixSum(arr, n, prefixSum);

            for (int i = 0; i < n; i++)
            System.out.print(prefixSum[i] + " ");
            System.out.println("");
        }

}
```

- **Output :**

```
10  14  30  50
```

## 2. Parallel algorithm design



## 3. Parallel code

```
package parallelmergesort;

public class ParallelMergeSort {

    public static void main(String[] args) {

        long alltime = System.nanoTime();
        int[] arr = {9, 85, 6, 56, 5, 4, 35, 89, 23};
        long Right = 0;
        long Left = 0;

        MyThread task1 = new MyThread(arr, 0, arr.length / 2);
        MyThread task2 = new MyThread(arr, arr.length / 2, arr.length);
        Thread t1 = new Thread(task1);
        Thread t2 = new Thread(task2);

        Left = System.nanoTime();
        t1.start();
        Right = System.nanoTime();

        double timet1 = Right - Left;

        Left = System.nanoTime();
        t2.start();
        Right = System.nanoTime();

        double timet2 = Right - Left;

        System.out.println("bubble sort array");
        for (int i = 0; i < arr.length; i++) {
            System.out.print(" " + arr[i]);         }

        System.out.println("");

        double tp1 = timet1 / 1000000000;
        double tp2 = timet2 / 1000000000;
        System.out.println("the time of t1   :  " + tp1);
        System.out.println("the time of t2   :  " + tp2);
        if (timet1 > timet2) {
            System.out.println("the parallel time    :  " + tp1);
        } else {
            System.out.println("the parallel time   :  " + tp2);
        }

        float total = System.nanoTime() - alltime / 1000000000;
        System.out.println("the total time :" + total);
        System.out.println("");
      System.out.println("");
      double speed = 0;
      if (timet1 > timet2) {
          speed = total / tp1;
          System.out.println("Speedup= " + total / tp1);
      } else {
      speed = total / tp2;
      System.out.println("Speedup= " + total / tp2);
      } System.out.println("Effeciency = " + speed / 2);
      }

    public static class MyThread implements Runnable {

        int let;
        int rit;
        int[] arr;
        int temp;

        MyThread(int[] arr, int let, int rit) {
            this.let = let;
            this.rit = rit;
            this.arr = arr;
            }
        @Override
        public void run() {
            for (int i = 0; i <= arr.length; i++) {
            for (int j = 1; j < (rit - i); j++) {
            if (arr[j - 1] > arr[j]) {
            //swap elements
            temp = arr[j - 1];
            arr[j - 1] = arr[j];
            arr[j] = temp;

                    }
                }
            }
        }
```

## 4. Sample output

```
Speedup= 1.1963034324498156E19
Effeciency = 5.9815171622490778E18
```