Data Structures

Sorting Technique – Merge Sort

Team Emertxe

Introduction

Introduction

Merge Sort:

- Merge Sort is usually done Recursively
- It works on Divide and Conquer algorithm

Introduction

Merge Sort:

- Merge Sort is usually done Recursively
- It works on Divide and Conquer algorithm
- A sort algorithm that splits the items to be sorted into two groups, recursively sorts each group, and merges them into a final, sorted sequence

Merge Sort

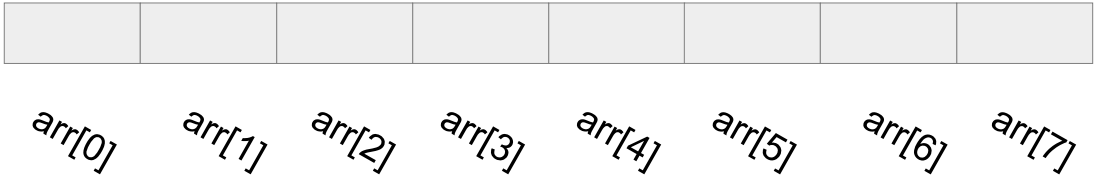
- arr[SIZE]

SIZE = 8

Merge Sort

- arr[SIZE]

SIZE = 8



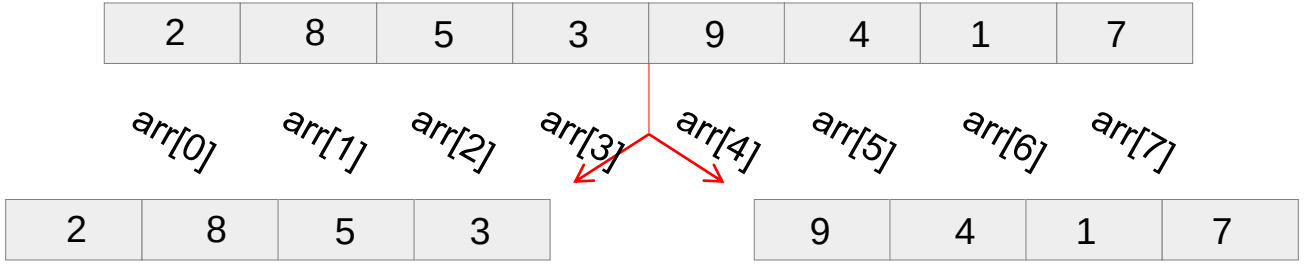
Merge Sort

- arr[SIZE]

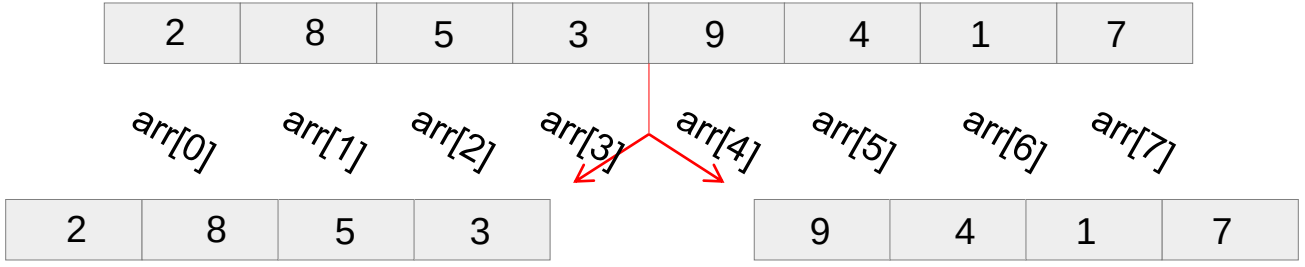
SIZE = 8

2	8	5	3	9	4	1	7
arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]	arr[7]

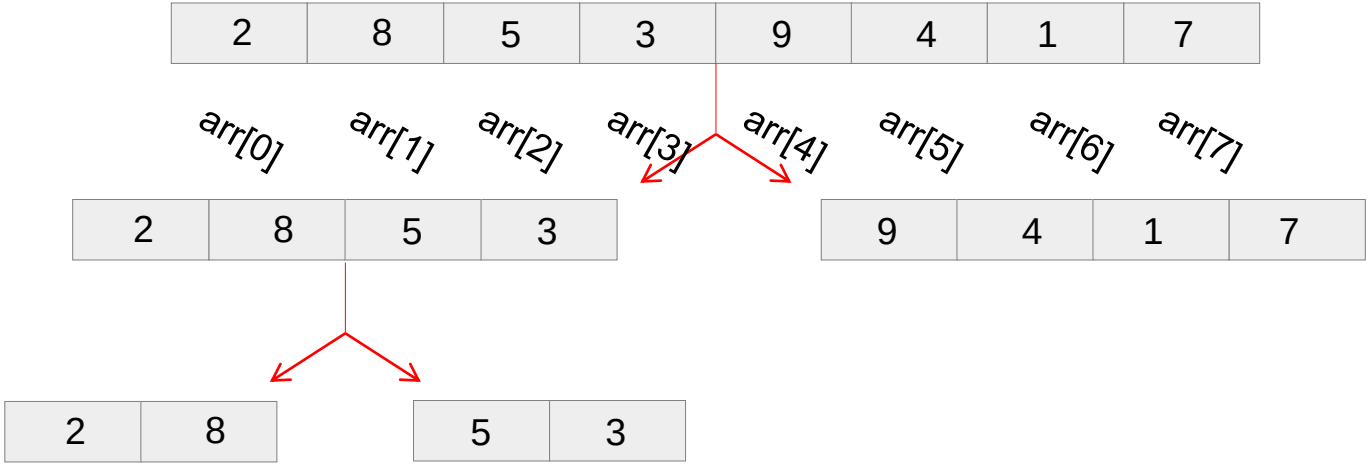
Merge Sort



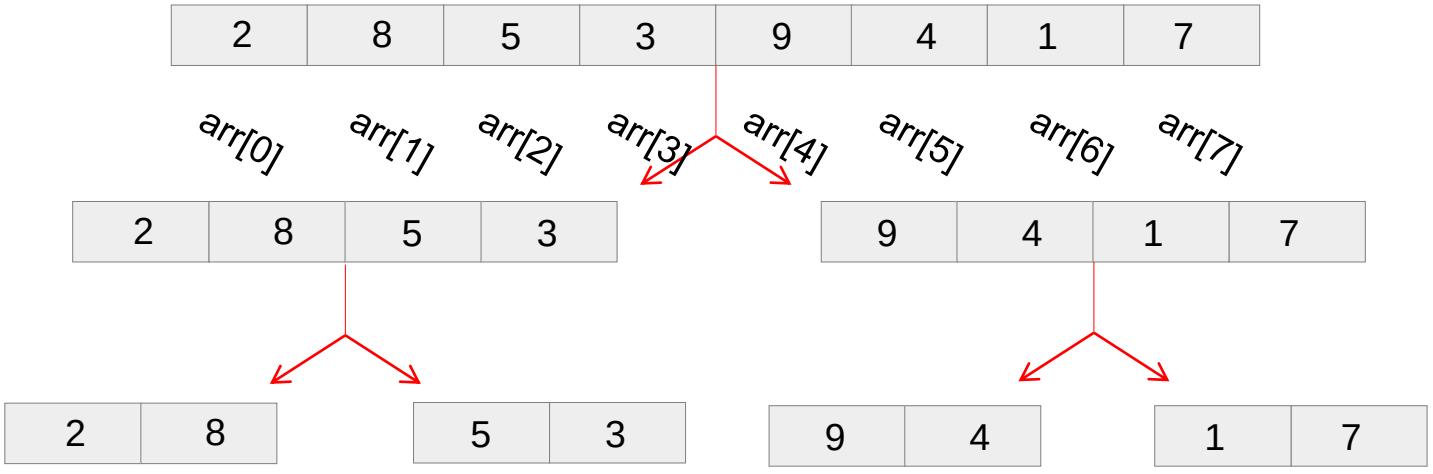
Merge Sort



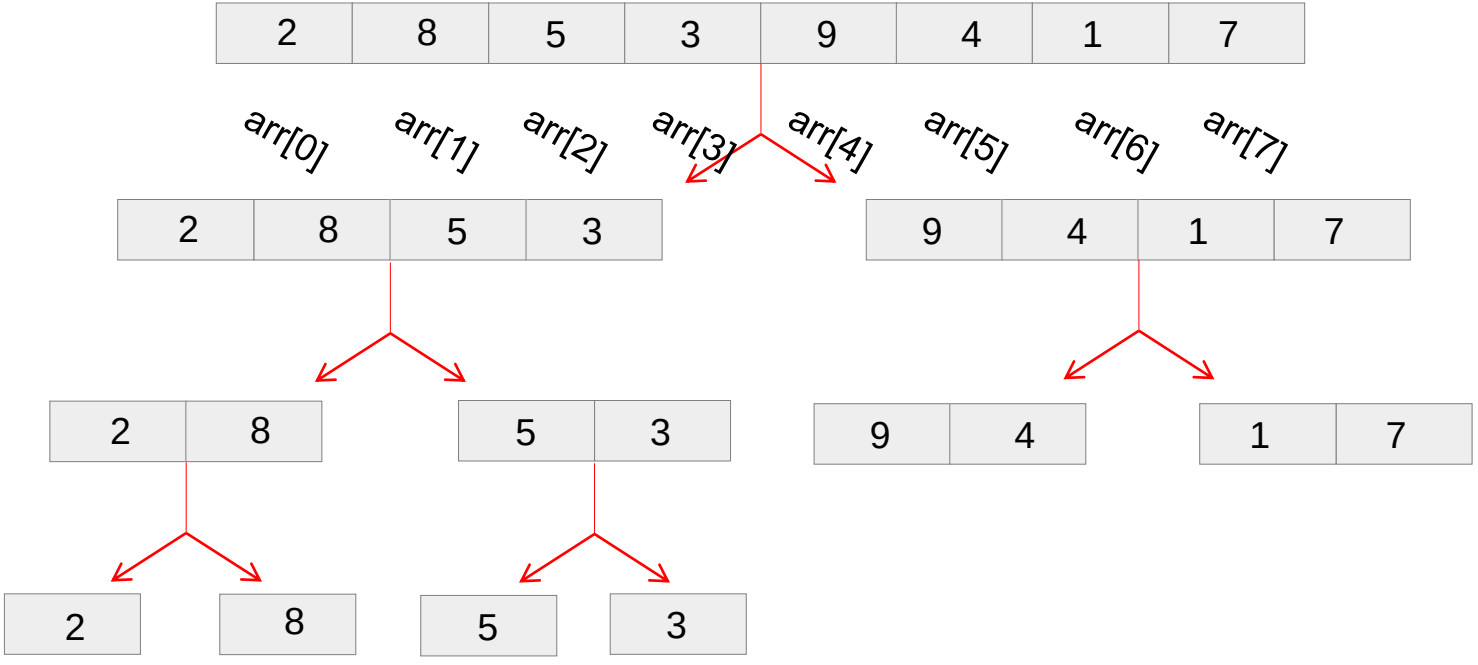
Merge Sort


2
8
5
3
9
4
1
7
arr[0]
arr[1]
arr[2]
arr[3]
arr[4]
arr[5]
arr[6]
arr[7]
2
8
5
3
9
4
1
7
2
8
5
3

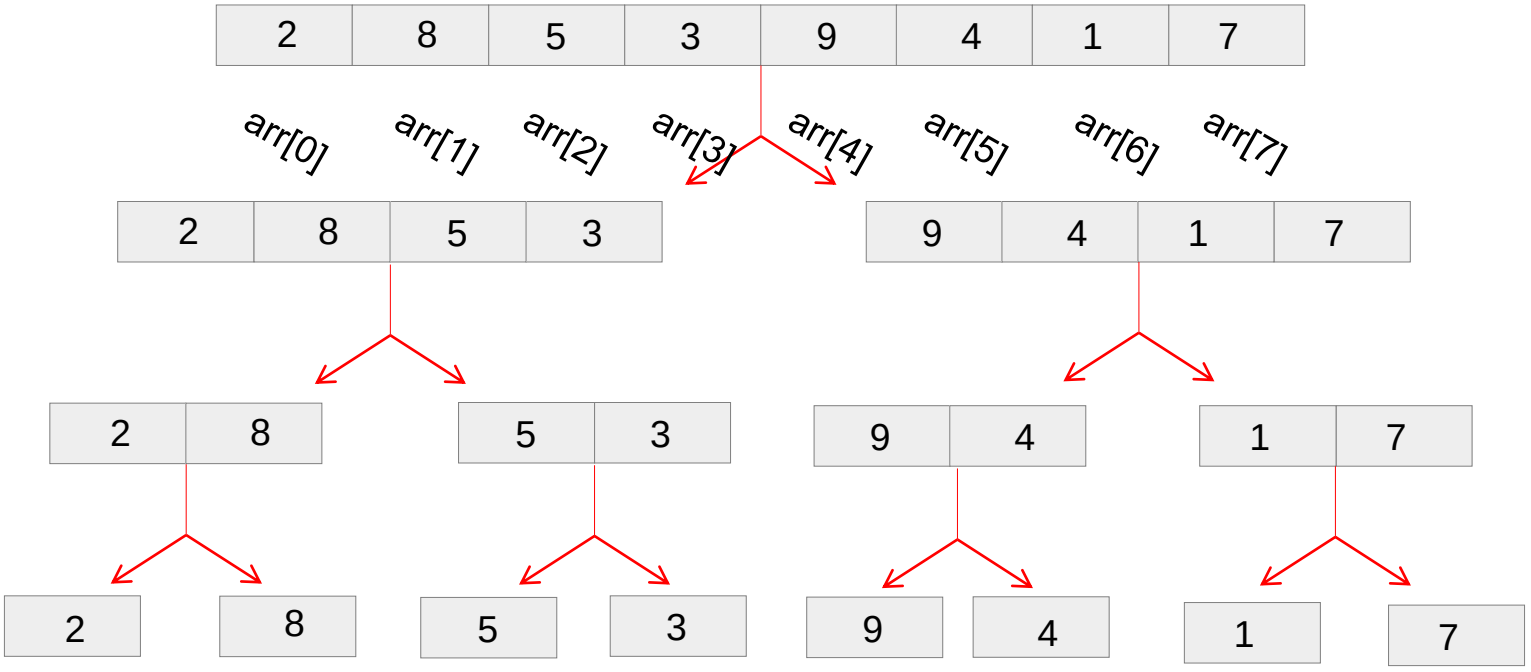
Merge Sort


2 8 5 3 9 4 1 7
arr[0] arr[1] arr[2] arr[3] arr[4] arr[5] arr[6] arr[7]
2 8 5 3
9 4 1 7
2 8
5 3
9 4
1 7

Merge Sort

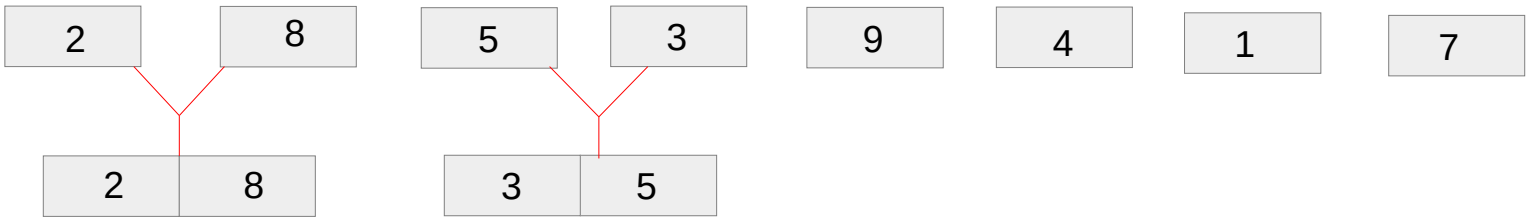

2 8 5 3 9 4 1 7
arr[0] arr[1] arr[2] arr[3] arr[4] arr[5] arr[6] arr[7]
2 8 5 3
9 4 1 7
2 8
5 3
9 4
1 7
2
8
5
3

Merge Sort

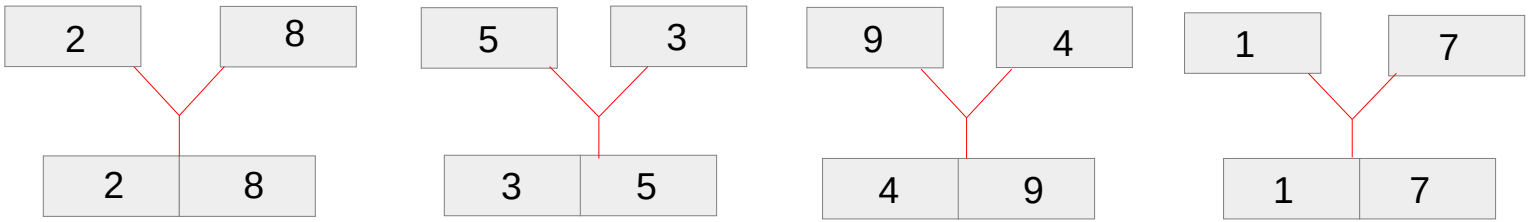

2 8 5 3 9 4 1 7
arr[0] arr[1] arr[2] arr[3] arr[4] arr[5] arr[6] arr[7]
2 8 5 3
9 4 1 7
2 8
5 3
9 4
1 7
2
8
5
3
9
4
1
7

Merge Sort

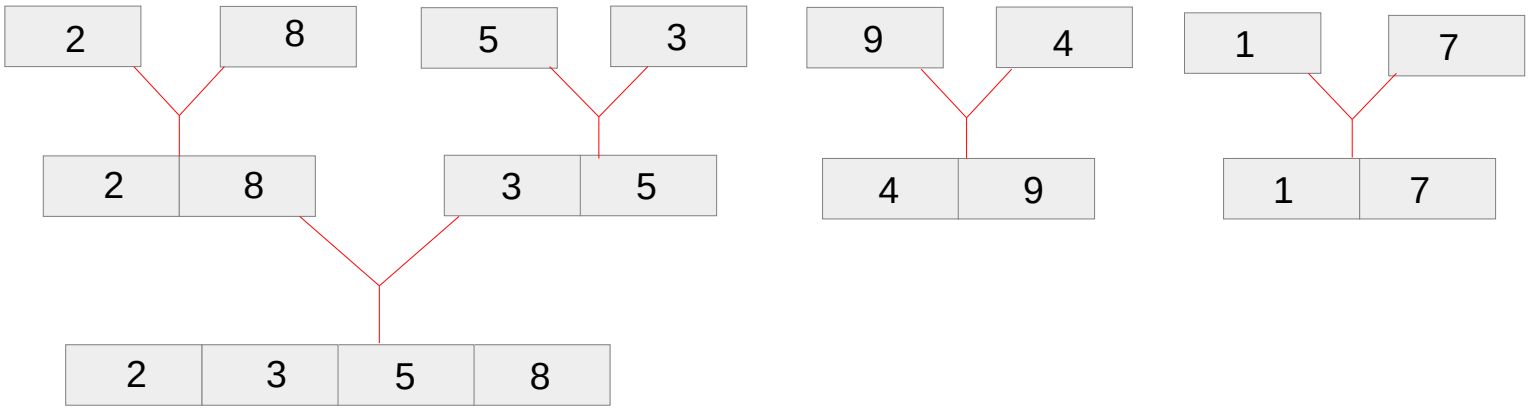
Merge Sort



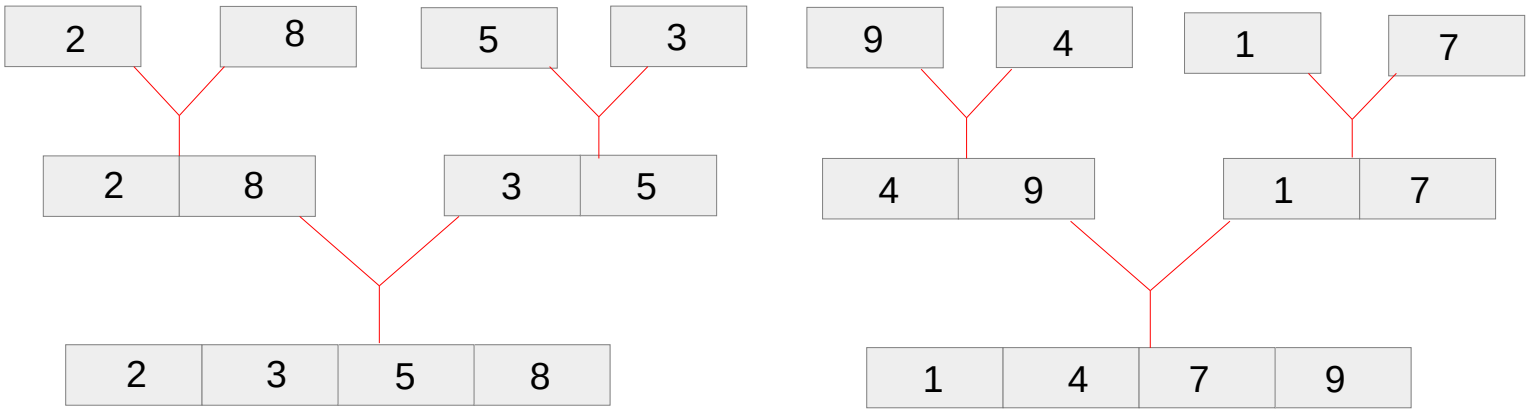
Merge Sort



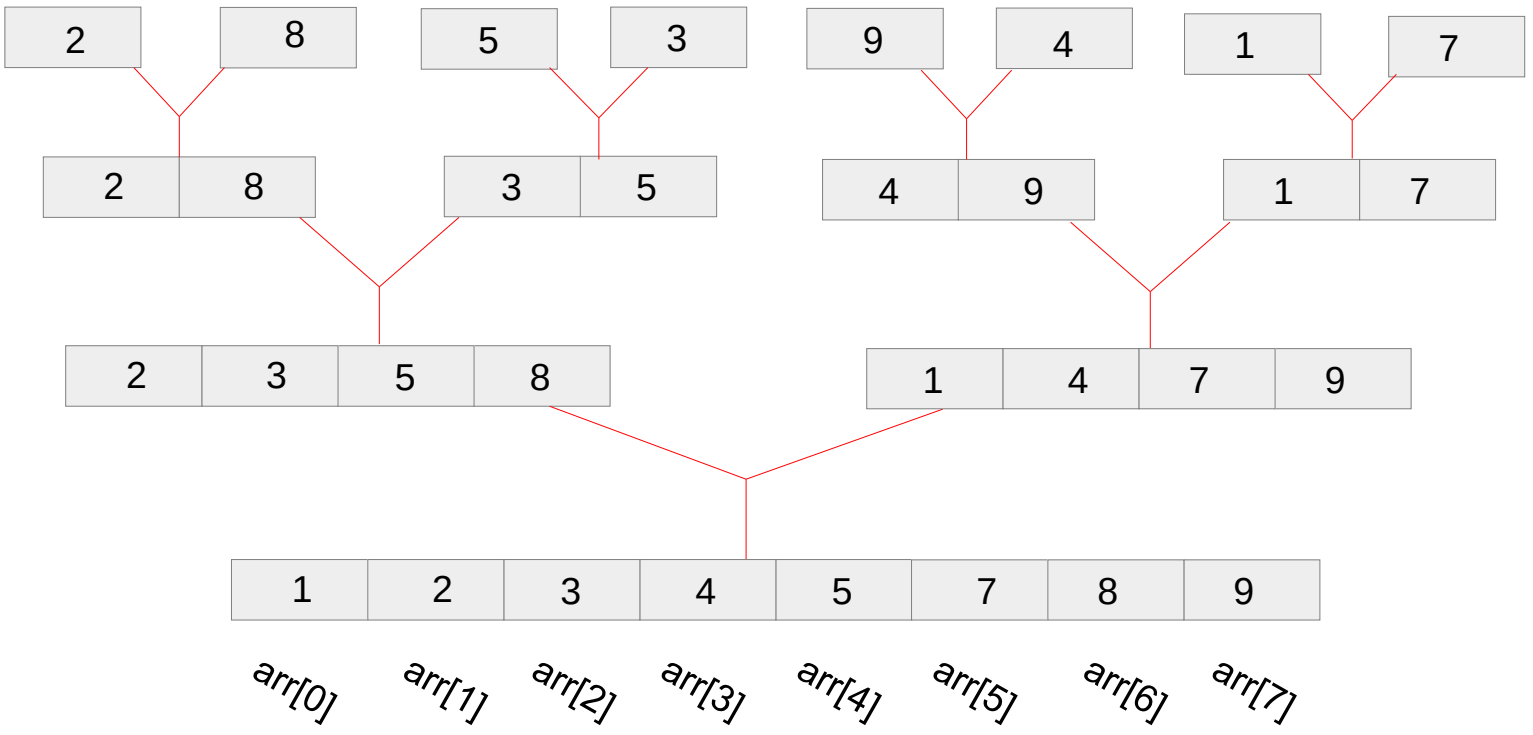
Merge Sort



Merge Sort



Merge Sort



Algorithm

Merge_Sort(arr,size)

If (size = 1)

  return 1

mid = size/2

L_{SA} = malloc(mid *sizeof int)

For i =0 upto mid-1

  L_{SA}[i] = arr[i]

R_{SA} = malloc(size – mid * sizeof(int))

For i = mid to size-1

  R_{SA}[i-mid] = arr[i]

Merge_Sort(L_{SA},mid)

Merge_Sort(R_{SA},size-mid)

merge(arr,size,L_{SA},mid,R_{SA},size-mid)

Algorithm

Merge_Sort(arr,size)

If (size = 1)
  return 1
mid = size/2
L_{SA} = malloc(mid *sizeof int)
For i =0 upto mid-1
  L_{SA}[i] = arr[i]
R_{SA} = malloc(size – mid * sizeof(int))
For i = mid to size-1
  R_{SA}[i-mid] = arr[i]
Merge_Sort(L_{SA},mid)
Merge_Sort(R_{SA},size-mid)
merge(arr,size,L_{SA},mid,R_{SA},size-mid)

Merge(arr,size,L_{SA},mid,R_{SA},size-mid)

i = j = k = 0
1. while(i < mid AND j < (size-mid)
  If (L_{SA}[i] < R_{SA}[j])
    arr[k] = L_{SA}[i]
    Increment i
  Else
    arr[k] = R_{SA}[j]
    Increment j
  Increment k
2. while(j < (size – mid))
    arr[k] = R_{SA}[j]
    Increment j
    Increment k
3. while(i < mid)
    arr[k] = L_{SA}[j]
    Increment i
    Increment k

Merge Sort

Advantages

- It is used for External Sorting
- It can be applied to files of any size.
- Used to implement Stable Sort

Merge Sort

Advantages

- It is used for External Sorting
- It can be applied to files of any size
- Used to implement Stable Sort

Disadvantages

- Requires extra space ,more than other sorting technique

Merge Sort

Advantages

- It is used for External Sorting
- It can be applied to files of any size.
- Used to implement Stable Sort

Disadvantages

- Requires extra space ,more than other sorting technique

Time Complexity

- O(nlogn)

Code - Merge Sort