MERGE SORT ALGORITHM

It’s a sorting technique that **sequences data by continuously merging items in the list**. Every single item in the original unordered list is merged with another, creating groups of two. Every two-item group is merged, creating groups of four and so on until there is one ordered list.

Merge Sort Algorithm (Working)

mergeSort(arr [ ], I, r) [ arr = Array, l = leftmost index , r = rightmost index]

{

Step 1 – Find the middle point to divide the array into two halves:

Middle m = (I+r)/2

**Division**

Step 2 – Call merge sort for first half:

mergeSort(arr,I,m)

Step 3 – Call merge sort for second half:

mergeSort(arr,m+1,r)

**Recursion**

Step 4 – Merge the two halves sorted in step 2 and 3

merge (arr,I,m,r)

**Merging**

}

Merge Sort implementation

mergeSort(arr[],l,r)

{

if (l<r)

{

1. m=(l+r)/2

2. mergeSort(arr,l,m)

3. mergeSort(arr,m+1,r)

4. merge(arr,l,m,r)

}

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 9 | 7 | 3 | 6 | 2 |

|  |  |
| --- | --- |
| 2 | 6 |

|  |  |  |
| --- | --- | --- |
| 3 | 7 | 9 |

|  |
| --- |
| 6 |

|  |
| --- |
| 2 |

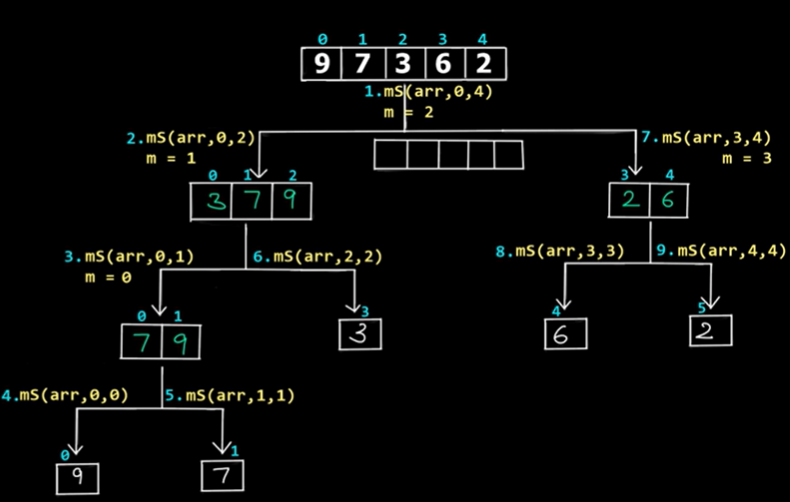
|  |  |
| --- | --- |
| 7 | 9 |

|  |
| --- |
| 3 |

|  |
| --- |
| 9 |

|  |
| --- |
| 7 |

Representation



Step 4 Implementation

merge(arr,l,m,r)

{

1.i=l,j=m+1,k=l // 3 variables

2.temp[] //create temp array

3.while (i<=m && j<=r)

3.1 if(arr[i]<= arr[j])

temp[k]=arr[i]

i++,k++

3.2 else

temp[k]=arr[j]

j++,k++

4.while (i<=m)

temp[k]=arr[i]

i++,k++

5.while(j<=r)

temp[k]=arr[j]

j++,k++

6.for(int p=l;p<=r;p++)

arr[p]=temp[p];

}