
Fundamentals of
Data Structures using C

# Merge Sort

**B.Bhuvaneswaran, AP (SG) / CSE**

9791519152

bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI
ENGINEERING COLLEGE

# Merge Sort

- The most common algorithm used in external sorting is the merge sort.
- This algorithm follows divide and conquer strategy.
  - In dividing phase, the problem is divided into smaller problem and solved recursively.
  - In conquering phase, the partitioned array is merged together recursively.
- Merge sort is applied to the first half and second half of the array.
- This gives two sorted halves, which can then be recursively merged together using the merging algorithm.

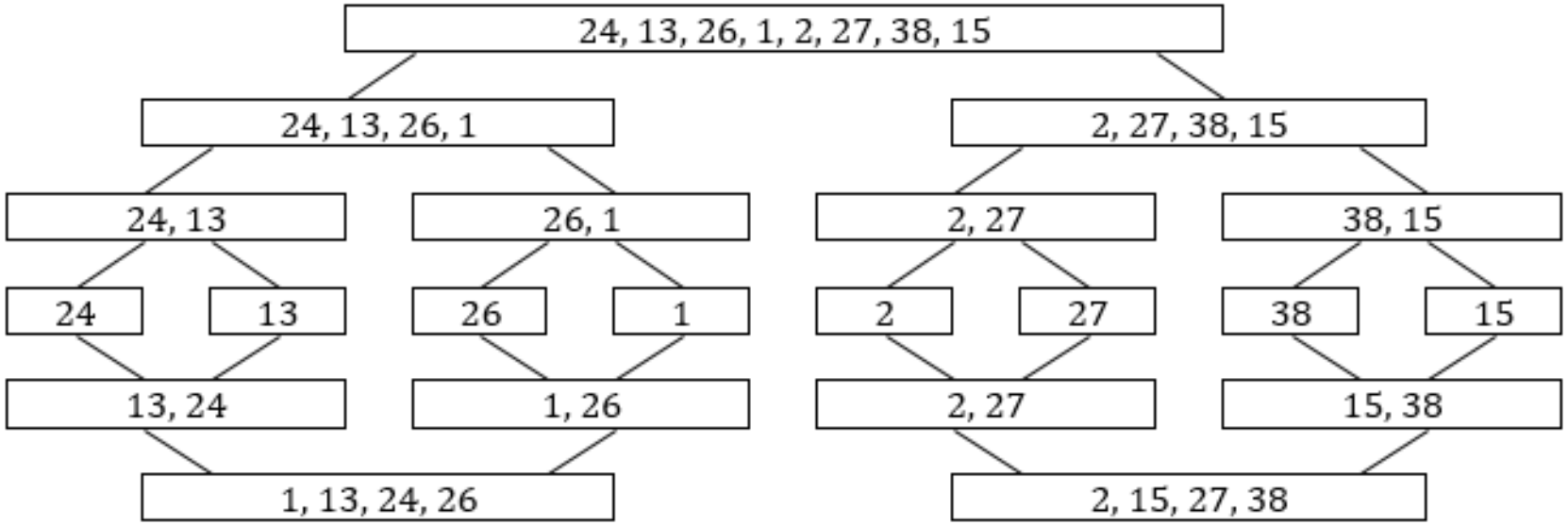
# Merge Sort

- The basic merging algorithm takes two input arrays A and B and an output array C.
- The first element of A array and B array are compared, then the smaller element is stored in the output array C and the corresponding pointer is incremented.
- When either input array is exhausted the remainder of the other array is copied to an output array C.

# Example

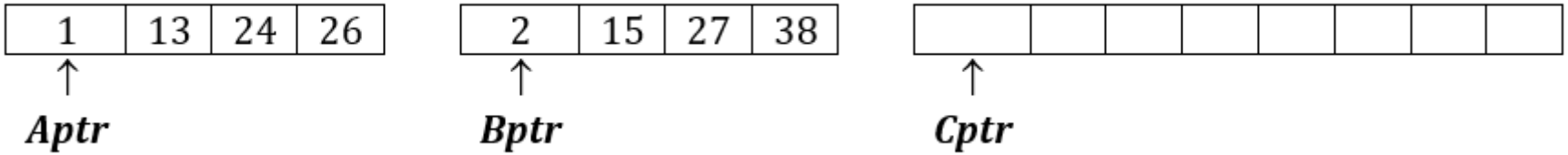
- For instance, to sort the eight element array 24, 13, 26, 1, 2, 27, 38, 15, we recursively sort the first four and last four elements, obtaining 1, 13, 24, 26, 2, 15, 27, 38 then these array is divided into two halves and the merging algorithm is applied to get the final sorted array.

# Example


24, 13, 26, 1, 2, 27, 38, 15
24, 13, 26, 1
2, 27, 38, 15
24, 13
26, 1
2, 27
38, 15
24
13
26
1
2
27
38
15
13, 24
1, 26
2, 27
15, 38
1, 13, 24, 26
2, 15, 27, 38

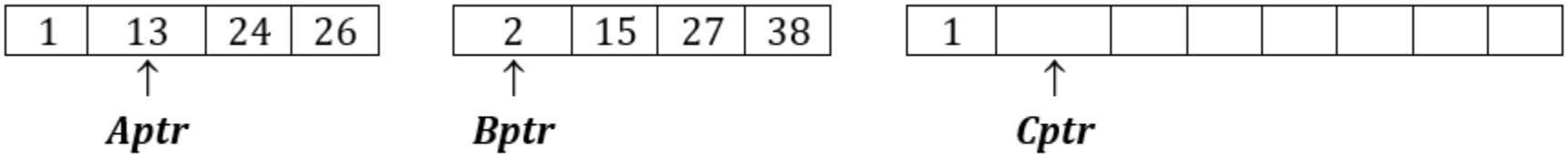
# Merge

- Let us consider first 4 elements 1, 13, 24, 26 as A array and the next four elements 2, 15, 27, 38 as B array.



# Merge

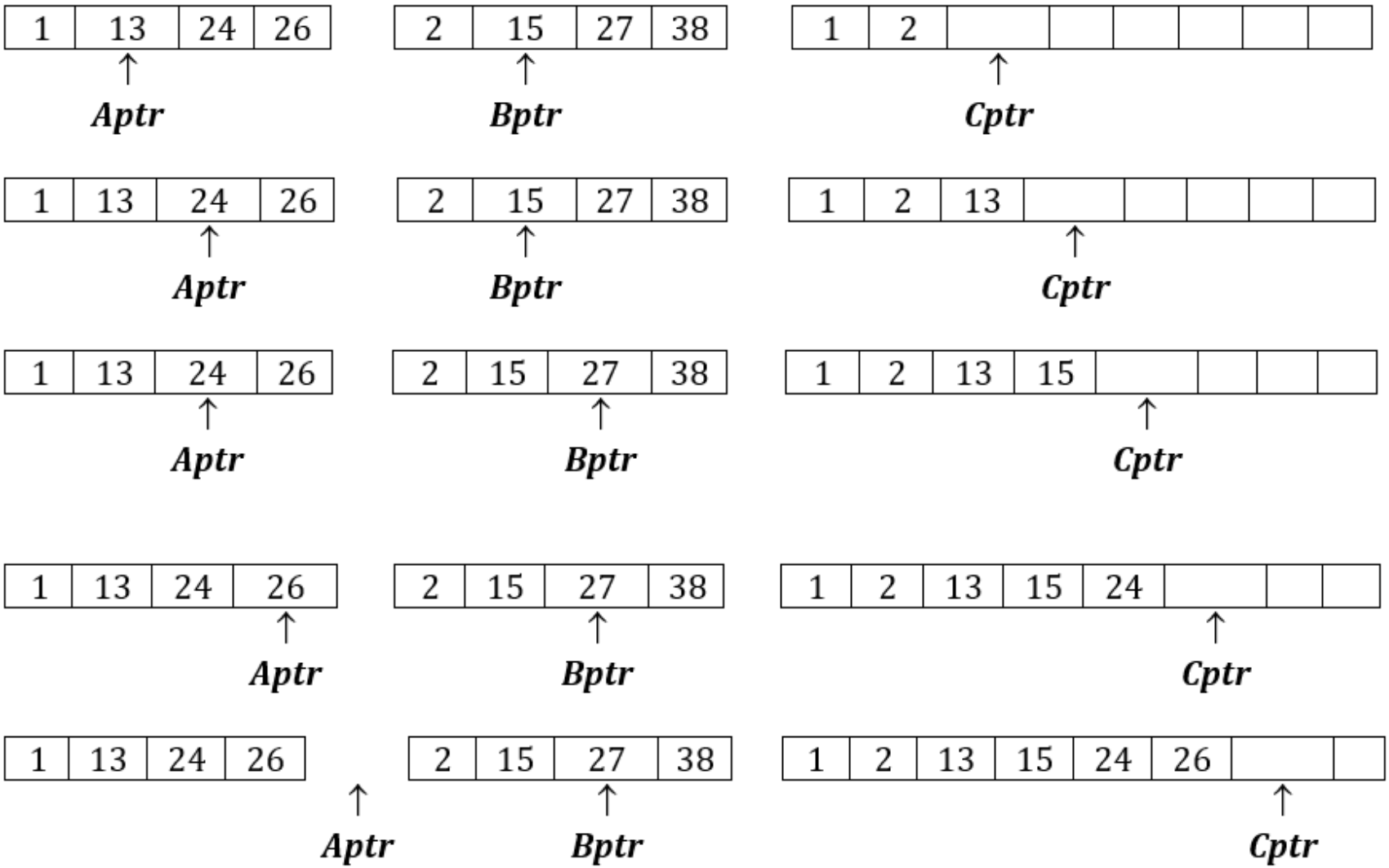
- First, the element 1 from A array and element 2 from B array is compared, then the smallest element 1 from A array is copied to an output array C.
- Then the pointers Aptr and Cptr is incremented by one.



# Merge

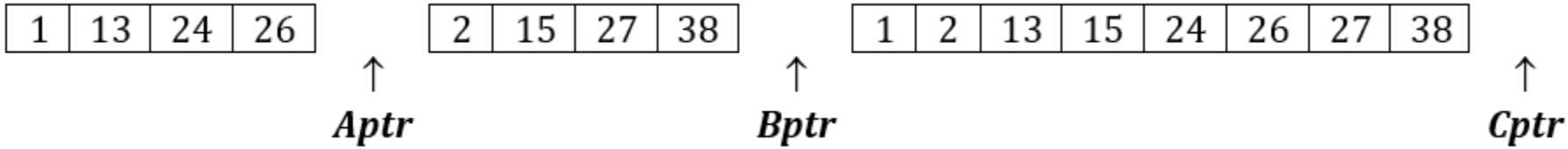
- Next, 13 and 2 are compared and the smallest element 2 from B array is copied to C array and the pointers Bptr and Cptr gets incremented by one.
- This proceeds until A array and B array are exhausted, and all the elements are copied to an output array C.

# Merge

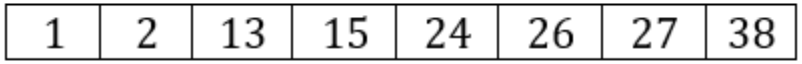

1 13 24 26
Aptr
2 15 27 38
Bptr
1 2
Cptr
1 13 24 26
Aptr
2 15 27 38
Bptr
1 2 13
Cptr
1 13 24 26
Aptr
2 15 27 38
Bptr
1 2 13 15
Cptr
1 13 24 26
Aptr
2 15 27 38
Bptr
1 2 13 15 24
Cptr
1 13 24 26
Aptr
2 15 27 38
Bptr
1 2 13 15 24 26
Cptr

# Merge

- Since A array is exhausted, the remaining elements of B array is then copied to C array.




- Final sorted array:



# Routine

```
void MergeSort(int arr[], int left, int right)
{
        int center;
        if (left < right)
        {
                center = (left + right) / 2;
                MergeSort(arr, left, center);
                MergeSort(arr, center + 1, right);
                Merge(arr, left, center, right);
        }
}
```

# Routine

```
void Merge(int arr[], int left, int center, int right)
{
        int a[20], b[20], n1, n2, aptr, bptr, cptr, i, j;
        n1 = center - left + 1;
        n2 = right - center;
        for (i = 0; i < n1; i++)
                a[i] = arr[left + i];
        for (j = 0; j < n2; j++)
                b[j] = arr[center + 1 + j];
        aptr = 0;
        bptr = 0;
        cptr = left;
```

# Routine

```
while (aptr < n1 && bptr < n2)
{
        if (a[aptr] <= b[bptr])
        {
                arr[cptr] = a[aptr];
                aptr++;
        }
        else
        {
                arr[cptr] = b[bptr];
                bptr++;
        }
        cptr++;
}
```

# Routine

```
        while (aptr < n1)
        {
                arr[cptr] = a[aptr];
                aptr++;
                cptr++;
        }
        while (bptr < n2)
        {
                arr[cptr] = b[bptr];
                bptr++;
                cptr++;
        }
}
```

# Analysis of Merge Sort

- Worst case analysis : O(n log n)
- Best case analysis : O(n log n)
- Average case analysis : O(n log n)

# Advantages of Merge Sort

- It is a fast and stable sorting method.
- It always ensures an efficiency of O(n log n).
- It has better cache performance.
- Merge Sort is a Stable Sort.

# Limitations of Merge Sort

- It requires additional memory space to perform sorting. The size of the additional space is in direct proportion to the size of the input list.
- Even though the number of comparisons made by merge sort are nearly optimal, its performance is slightly lesser than that of quick sort.
- Merge sort sorts the larger amount of data making use of external storage device.
- It requires extra memory space.

# Queries?

# Thank You!