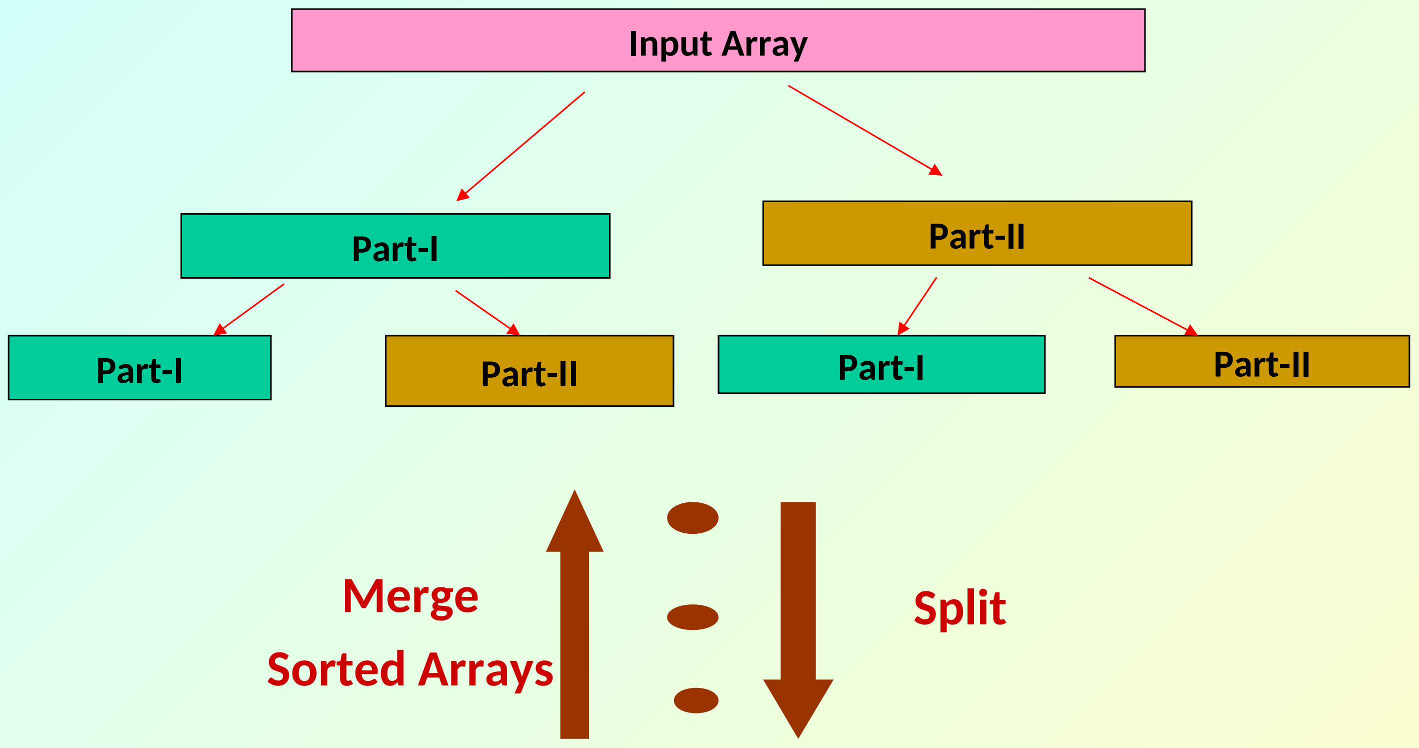
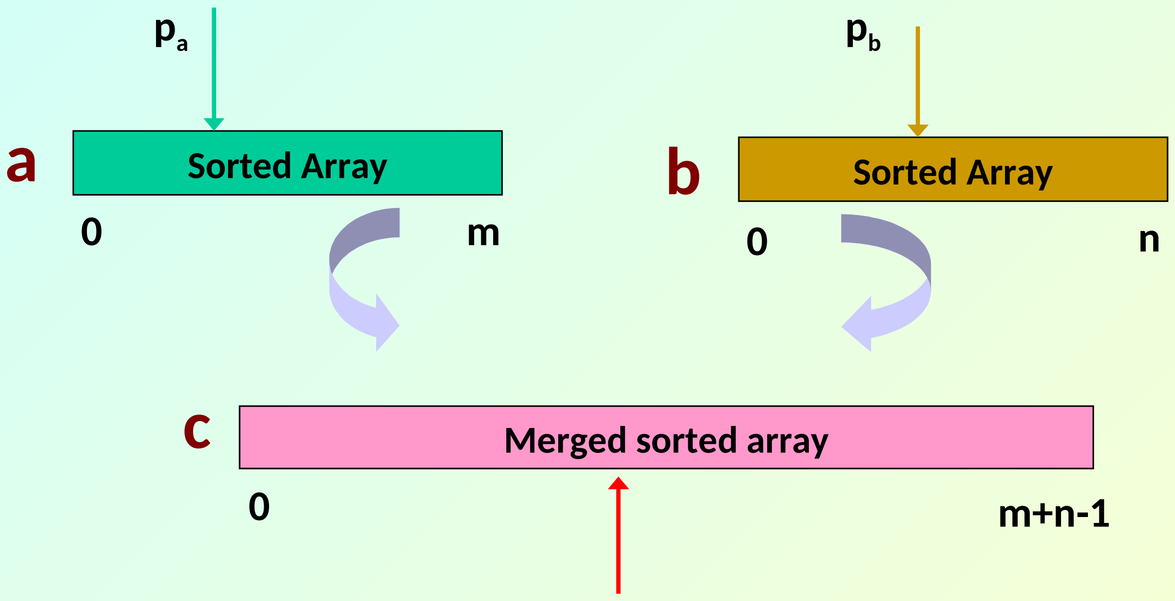
Merge Sort

Merge Sort


Input Array
Part-I
Part-II
Part-I
Part-II
Part-I
Part-II
Merge
Sorted Arrays
Split

Merging two sorted arrays



Move and copy elements pointed by p_a if its value is smaller than the element pointed by p_b in (m+n-1) operations; otherwise, copy elements pointed by p_b .

Example

- Initial array A contains 14 elements:
  - 66, 33, 40, 22, 55, 88, 60, 11, 80, 20, 50, 44, 77, 30
- Pass 1 :: Merge each pair of elements
  - (33, 66) (22, 40) (55, 88) (11, 60) (20, 80) (44, 50) (30, 70)
- Pass 2 :: Merge each pair of pairs
  - (22, 33, 40, 66) (11, 55, 60, 88) (20, 44, 50, 80) (30, 77)
- Pass 3 :: Merge each pair of sorted quadruplets
  - (11, 22, 33, 40, 55, 60, 66, 88) (20, 30, 44, 50, 77, 80)
- Pass 4 :: Merge the two sorted subarrays to get the final list
  - (11, 20, 22, 30, 33, 40, 44, 50, 55, 60, 66, 77, 80, 88)

```
void merge_sort (int *a, int n)
{
   int i, j, k, m;
   int *b, *c;

   if (n>1)  {
      k = n/2;
      m = n-k;
      b = (int *) malloc(k*sizeof(int));
      c = (int *) malloc(m*sizeof(int));

      for (i=0; i<k; i++)
         b[i]=a[i];
      for (j=k; j<n; j++)
         c[j-k]=a[j];

      merge_sort (b, k);
      merge_sort (c, m);
      merge (b, c, a, k, m);
      free(b); free(c);
   }
}
```

```
void merge (int *a, int *b, int *c, int m, int n)
{
   int i, j, k, p;
   i = j = k = 0;
   do  {
     if (a[i] < b[j])  {
       c[k]=a[i]; i++;
     }
     else  {
       c[k]=b[j]; j++;
     }
     k++;
   }  while ((i<m) && (j<n));
   if (i == m) {
      for (p=j; p<n; p++)  { c[k]=b[p]; k++; }
   }
   else  {
      for (p=i; p<m; p++)  { c[k]=a[p]; k++; }
   }
}
```

```
main()
{
    int i, num;
    int a[ ] = {-56,23,43,-5,-3,0,123,-35,87,56,75,80};

    num = 12;

    printf ("\n Original list: ");
    print (a, 0, num-1);

    merge_sort (a, 12);

    printf ("\n Sorted list: ");
    print (a, 0, num-1);
}
```

Time Complexity

- **Best/Worst/Average case:**

  n $\log_2 n$

- **Drawback:**
  - Needs double amount of space for storage.
  - For sorting ***n*** elements, requires another array of size ***n*** to carry out merge.

Example :: sorting arrays of structures (bubble sort)

```
#include <stdio.h>
struct stud
{
    int  roll;
    char  dept_code[25];
    float  cgpa;
};

main()
{
  struc  stud  class[100], t;
  int  j, k, n;

  scanf  ("%d", &n);
        /* no. of students */
```

```
for  (k=0; k<n; k++)
  scanf (”%d %s %f”, &class[k].roll,
                class[k].dept_code,
                &class[k].cgpa);
for  (j=0; j<n-1; j++)
  for  (k=1; k<n-j; k++)
  {
    if  (class[k-1].roll >
                    class[k].roll)
    {
       t = class[k-1];
       class[k-1] = class[k];
       class[k] = t;
     }
   }
    <<<< PRINT THE RECORDS >>>>
}
```