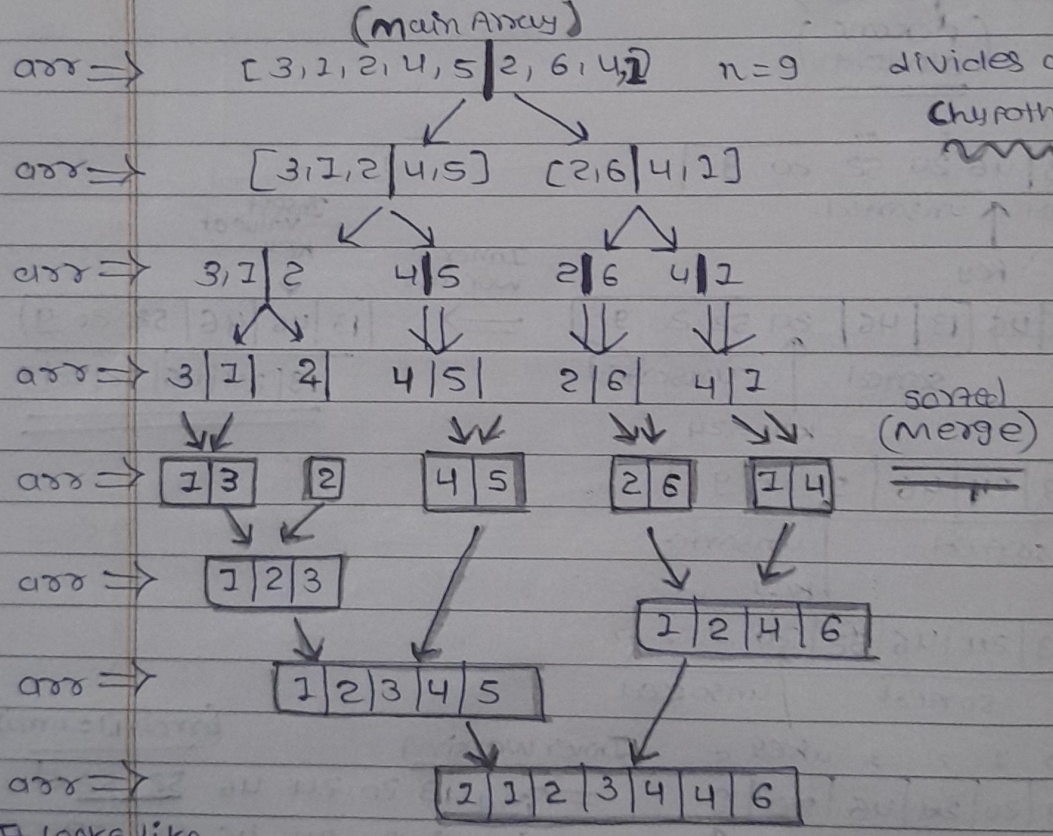
Sorting - II

Merge Sort

* Divide and conquer (merge)
* Much more optimized sorting algo.



It looks like Recursion tree
(so here we will use Recursion)

Recursively [divide & merge]

TC :- O(n log n)
SC :- O(n)

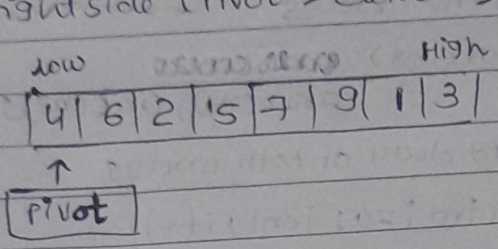
```
mergeSort(arr, left, right)
{   if (left < right)  ——→ base condition
    {   int mid = left + (right-left)/2
        mergeSort(arr, left, mid);
        mergeSort(arr, mid+1, right);
        merge(arr, left, mid, right);
    }
}
```

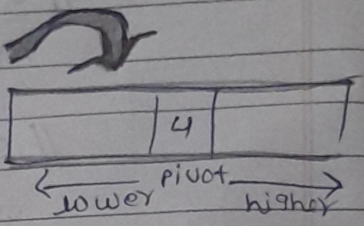
```
merge(arr, left, mid, right)
{   int n1 = mid - left + 1;
    int n2 = right - mid;
// (size of left & right sub arrays)
// 2 temp arrays
    vector<int>
            arr1(n1), arr2(n2);

// copy data of both arrays
    for(int i=0; i<n1; i++)
        arr1[i] = arr[left+i];
    for(int i=0; i<n2; i++)
        arr2[i] = arr[mid+1+i];
// merge Temp arrays
        (Logic of merge 2 sorted Arrays)
    int i=0, j=0, k=left
    while(i<n1 && j<n2)
    {   if(arr1[i]<arr2[j])
        {   arr[k] = arr1[i];
            i++;
        }
        else
        {   arr[k] = arr2[j];
            j++;
        }
        k++;
    }
    while(j<n2)
    {   arr[k] = arr2[j];
        j++;
    } k++;
    while(i<n1)
    {   arr[k] = arr[i];
        i++;
    } k++;
}
```

Quick Sort

* Pivot element → sort it
  left side (pivot < element)
  right side (pivot > element)

- You can take any element as pivot





Code :-

```
Partition (arr, low, high).
{   int pivot = arr[low]
    int i = low+1 ;

    for(int j = low+1 ; j<=high ; j++)
    {   if(arr[j] < pivot)
        {   swap(arr[i], arr[j]);
            i++;
        }
    }
    swap (arr[low], arr[i-1]);
    return i-1;
}

quicksort (arr, low, high)
{   if (low<high)
    {   int pi = partition (arr, low, high);
        quicksort (arr, low, pi-1);
        quicksort (arr, pi, high);
    }
}
```

Recursive Bubble Sort

```
function (arr, n)
{   if (n<=1)  ---> Base condition
      {return;}
    for (int j=0; j<n-1; j++)
    { if (arr[j] > arr[j+1])
      { swap (arr[j], arr[j+1]); }
    }
    function (arr, n-1);
}
```

Recursive Insertion Sort

```
function (arr, n)
{   if (n<=1)  ---> Base condition
      {return;}
    function (arr, n-1)   // sort first n-1 elements
                          // backtracking

    int last = arr[n-1];
    int j = n-2;
    while (j>=0 && arr[j]>last)
    { arr[j+1] = arr[j];
      j--;
    }
    arr[j+1] = last;
}
```

If you don't understand / remember,
(1) Learn Normal sorting logic & code first
(2) Do the Dry run...