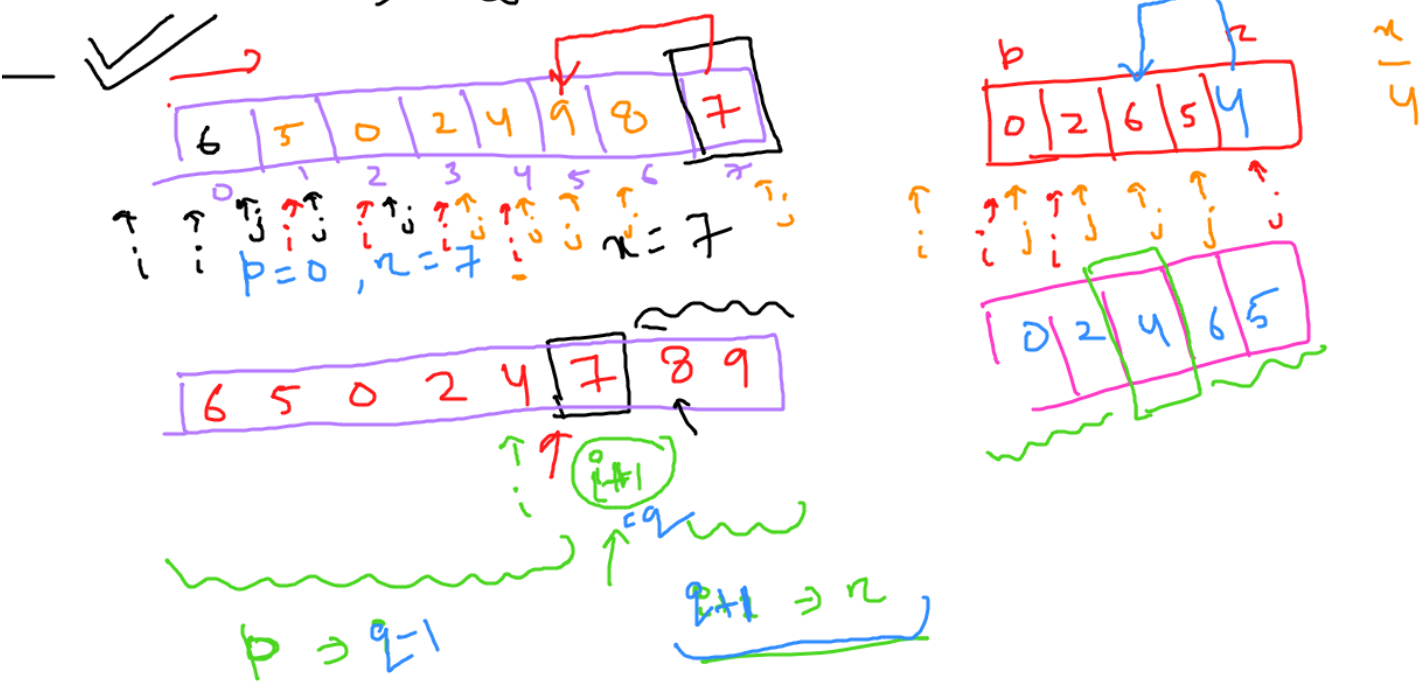
Quick sort ← Divide & Conquer Technique

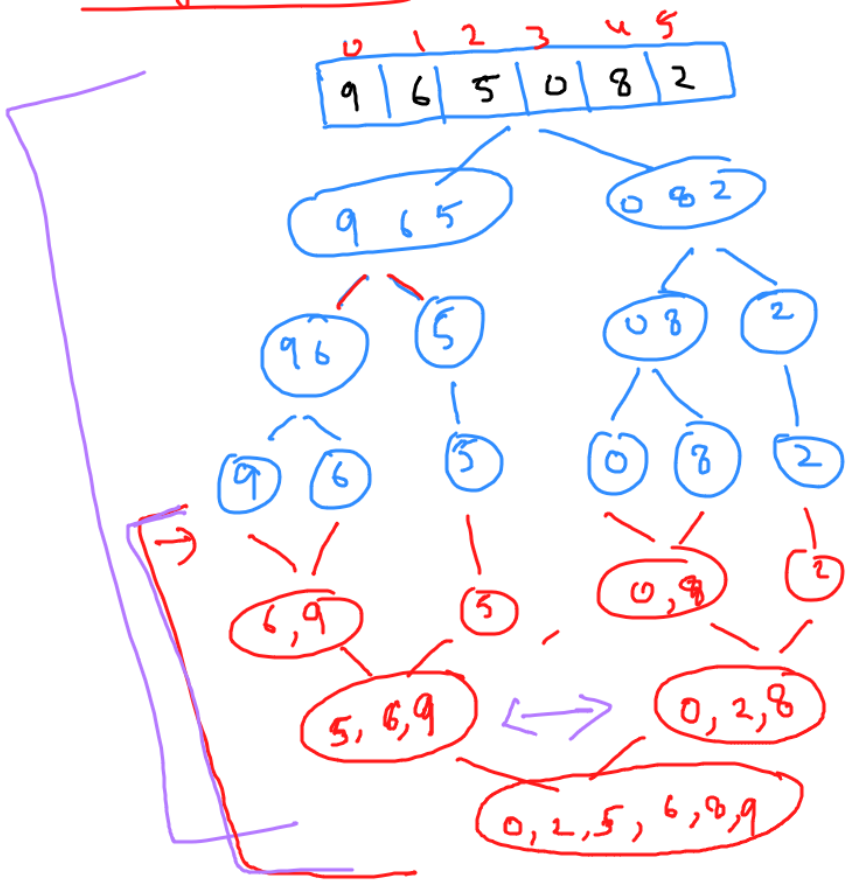
↳ partition ← Heart of the algorithm
↳ QuickSort




```
int partition (A, p, r)
{
    x = A[r]
    i = p-1
    for (j=p; j<r; j++)
    {
        if (A[j] <= x)
        {
            i++;
            swap(A[i], A[j]);
        }
    }
    swap (A[i+1], A[r]);
    return i+1;
}
```

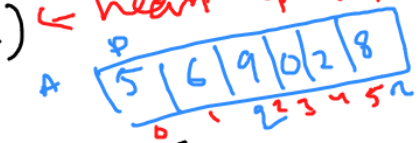
```
void quickSort(A, p, r)
{
    if (p < r)
    {
        q = partition(A, p, r)
        quicksort(A, p, q-1)
        quicksort(A, q+1, r)
    }
}
```

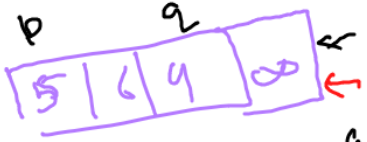
Merge sort ← D & C Technique



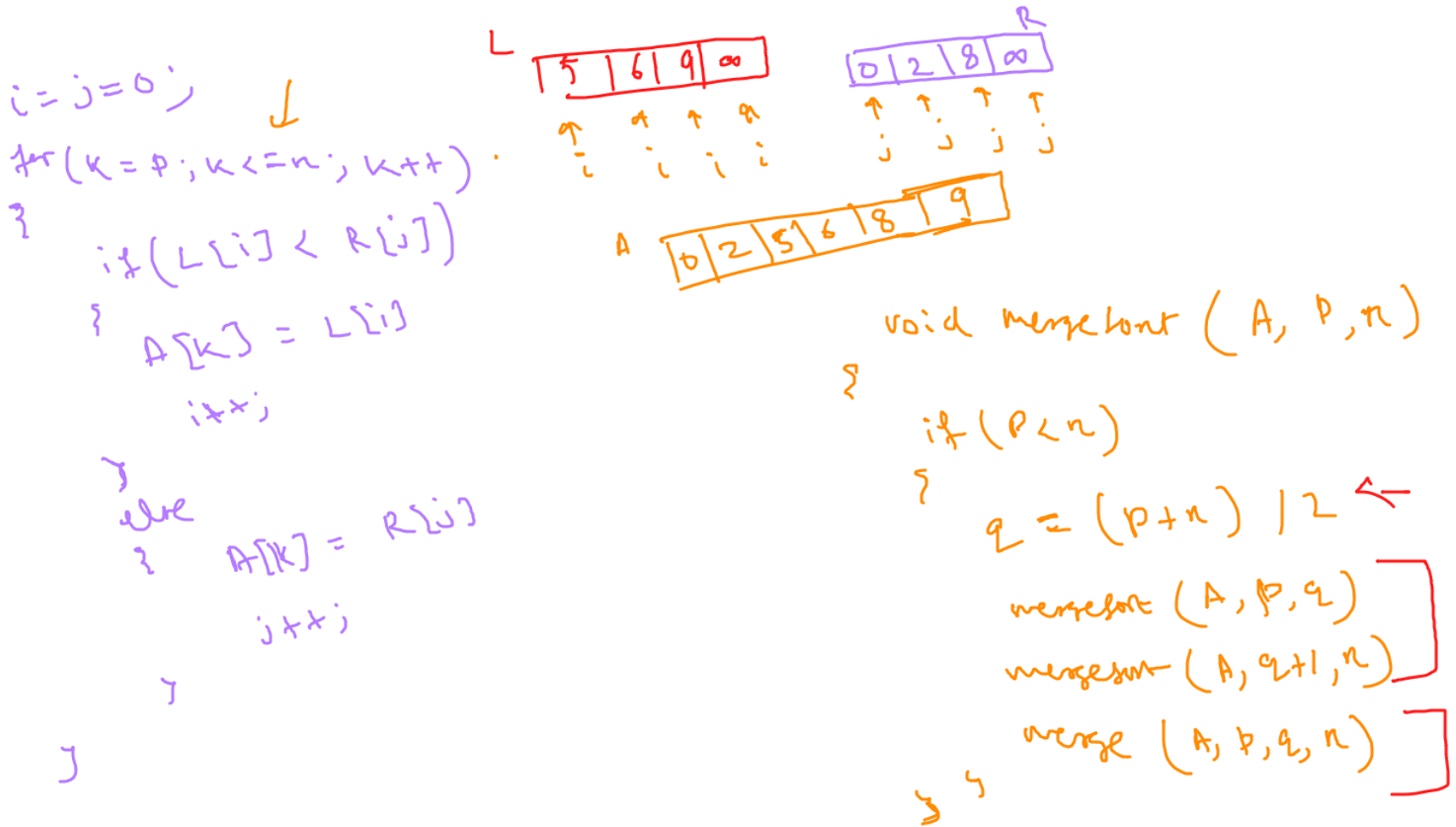
merge (A, p, q, r) ← heart of the algo

mergesort (A, p, r)






```
merge ( A, p, q, r )
{
    n1 = q - p + 1
    n2 = r - q
    Let L[n1+1] & R[n2+1] be two arrays.
    for (i=0; i<n1; i++)
    {   L[i] = A[p+i]
    }
    for (j=0; j<n2; j++)
    {   R[j] = A[q+1+j]
    }
    L[n1] = R[n2] = ∞;
```


i = j = 0;
for (k = p; k <= n; k++)
{
if (L[i] < R[j])
{
A[k] = L[i]
i++;
}
else
{
A[k] = R[j]
j++;
}
}
L
5 6 9 ∞
R
0 2 8 ∞
A
0 2 5 6 8 9
void mergeSort (A, p, n)
{
if (p < n)
{
q = (p+n) / 2
mergeSort (A, p, q)
mergeSort (A, q+1, n)
merge (A, p, q, n)
}
}