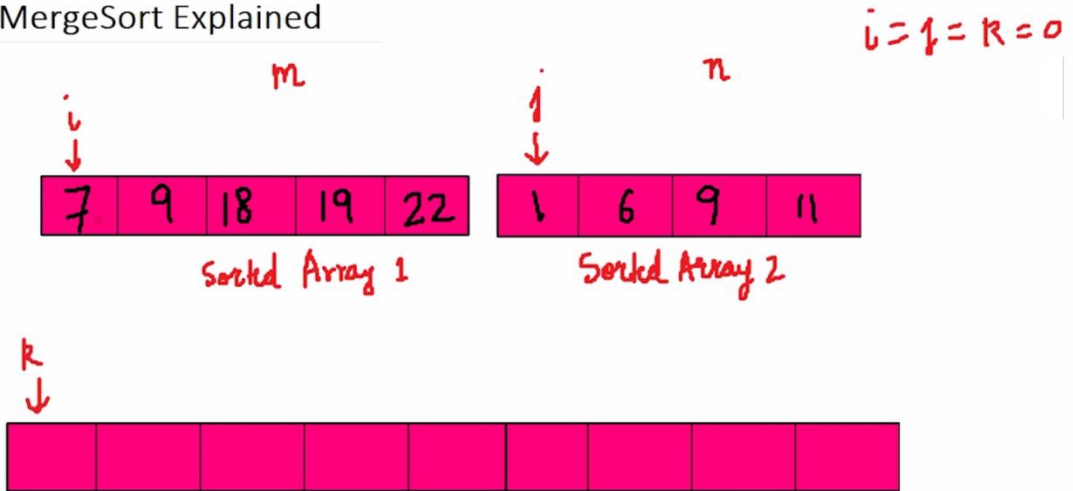
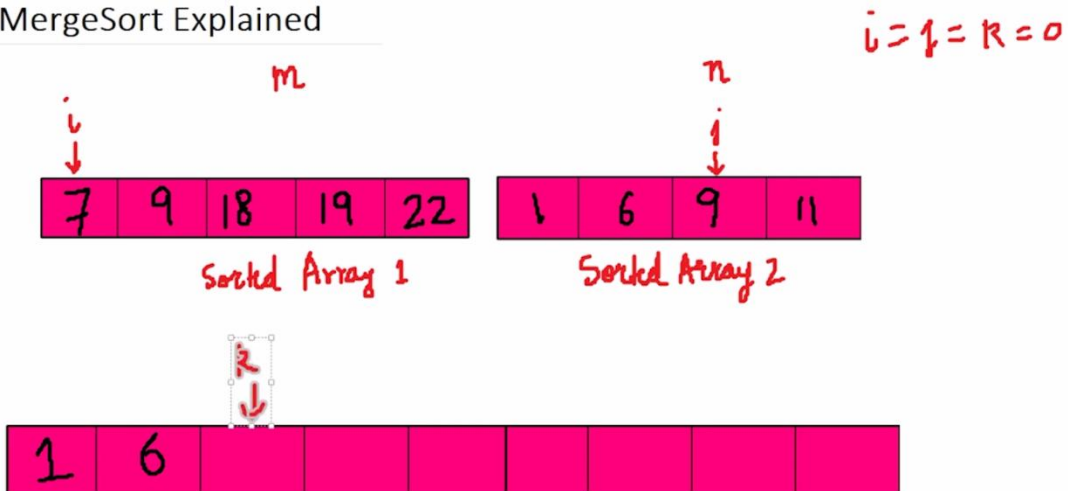


f

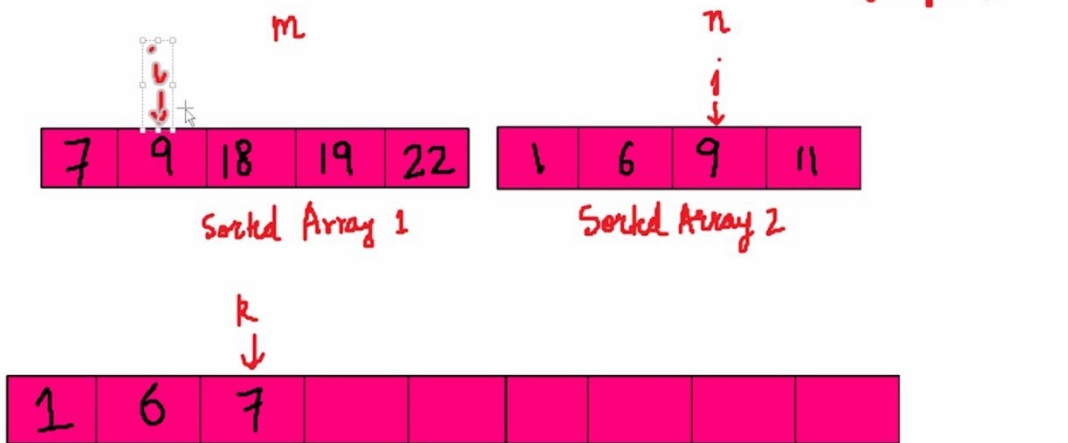
MergeSort Explained



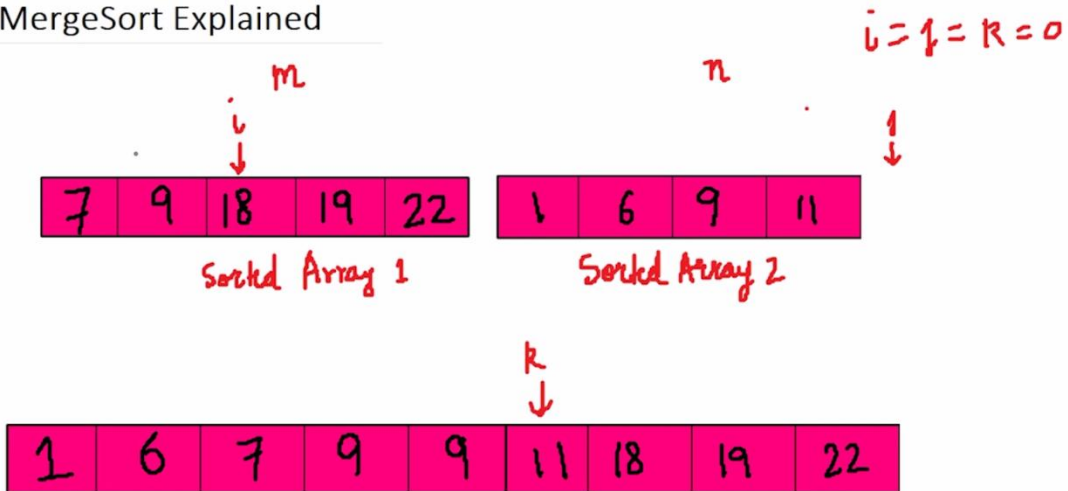
MergeSort Explained



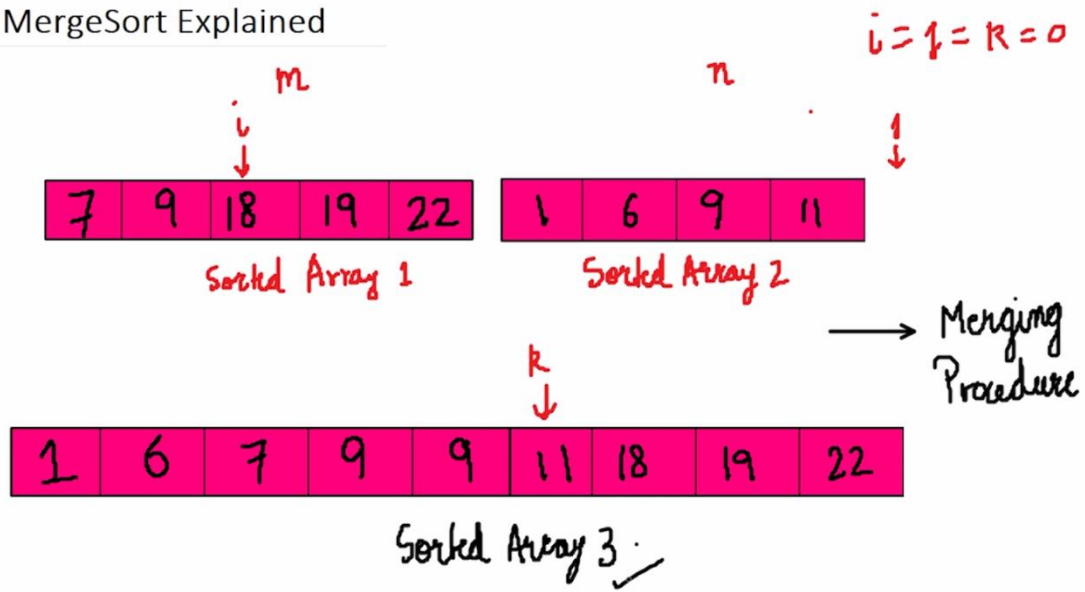
MergeSort Explained



MergeSort Explained

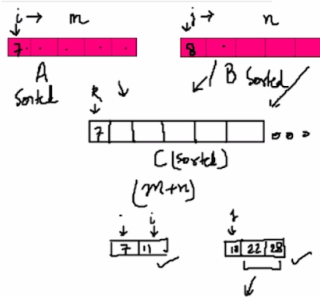


MergeSort Explained



Merge Procedure for code

Merging Procedure - Code!



Void Merge($A[]$, $B[]$, $C[]$, m , n) {

```
int i, j, k;
i = j = k = 0;
while (i < m && j < n) {
    if (A[i] < B[j]) {
        C[k] = A[i];
        i++; k++;
    }
    else {
        C[k] = B[j];
        j++; k++;
    }
}
```

```
while (i < m) {
    C[k] = A[i];
    k++; i++;
}
```

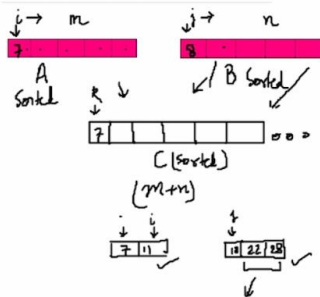
→ Copy all remaining elements from A to C

```
while (j < n) {
    C[k] = B[j];
    k++; j++;
}
```

→ from B to C

✓ ~~1~~, ~~2~~, 7, 8, 11, ✓
 ✓ ~~1~~, ~~2~~, ~~7~~, 8, 11, ✓

Merging Procedure - Code!



Void Merge($A[]$, $B[]$, $C[]$, m , n) {

```
int i, j, k;
i = j = k = 0;
while (i < m && j < n) {
    if (A[i] < B[j]) {
        C[k] = A[i];
        i++; k++;
    }
    else {
        C[k] = B[j];
        j++; k++;
    }
}
```

```
while (i < m) {
    C[k] = A[i];
    k++; i++;
}
```

✓ → Copy all remaining elements from A to C

```
while (j < n) {
    C[k] = B[j];
    k++; j++;
}
```

✓ → from B to C

✓ ~~2~~, ~~4~~, ~~5~~, 40, 50, 80
 ↓
 2, 4, 5, 40, 50, 80

Merging in a single array

low: 0, Mid: 1, High: 4

A: [7, 15, 2, 8, 10]

B: [, , , ,]

```

Void Merge(A[], mid, lo, hi) {
    int i, t, k;
    i = lo, t = mid + 1, k = lo;
    while (i <= mid && t <= hi) {
        if (A[i] < A[t]) {
            B[k] = A[i];
            i++; k++;
        }
        else {
            B[k] = A[t];
            t++; k++;
        }
    }
    while (i <= mid) {
        B[k] = A[i];
        k++; i++;
    }
    while (t <= hi) {
        B[k] = A[t];
        k++; t++;
    }
    for (i = lo; i <= hi; i++) A[i] = B[i];
}
        
```

Copy all remaining elements from A to C

from B to C

Merging in a Single Array

low: 0, Mid: 1, High: 4

A: [7, 15, 2, 8, 10]

B: [, , , ,]

```

Void Merge(A[], mid, lo, hi) {
    int i, t, k;
    i = lo, t = mid + 1, k = lo;
    while (i <= mid && t <= hi) {
        if (A[i] < A[t]) {
            B[k] = A[i];
            i++; k++;
        }
        else {
            B[k] = A[t];
            t++; k++;
        }
    }
    while (i <= mid) {
        B[k] = A[i];
        k++; i++;
    }
    while (t <= hi) {
        B[k] = A[t];
        k++; t++;
    }
    for (i = lo; i <= hi; i++) A[i] = B[i];
}
        
```

Copy all remaining elements from A to C

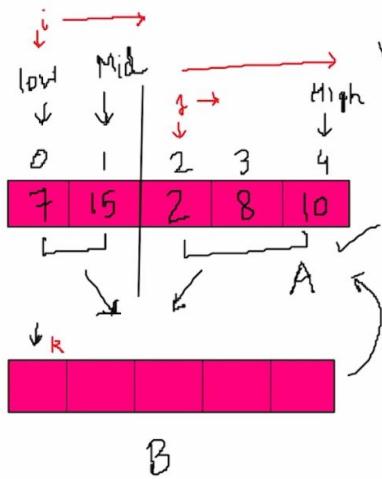
from B to C

7 8 11 | 2 3

1 2 3 7 8 11

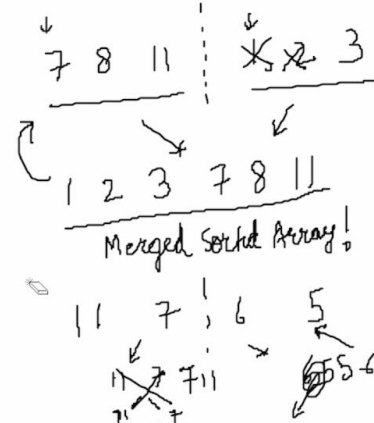
Merged Sorted Array!

Merging in a Single Array

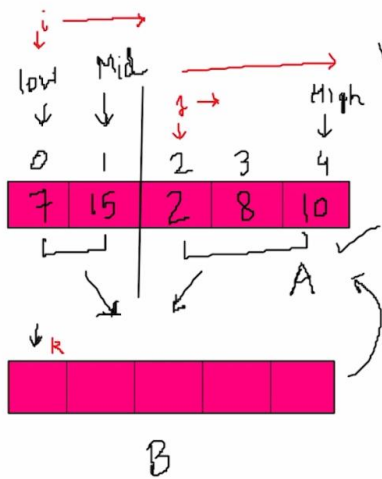


```

Void Merge(A[], mid, lo, hi) {
    int i, j, k;
    i = lo, j = mid + 1, k = lo;
    while (i <= mid && j <= hi) {
        if (A[i] < A[j]) {
            B[k] = A[i];
            i++; k++;
        }
        else {
            B[k] = A[j];
            j++; k++;
        }
    }
    while (i <= mid) {
        B[k] = A[i];
        k++; i++;
    }
    while (j <= hi) {
        B[k] = A[j];
        k++; j++;
    }
}
    
```

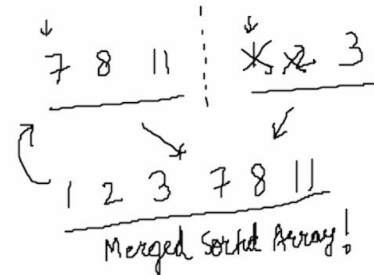


Merging in a Single Array

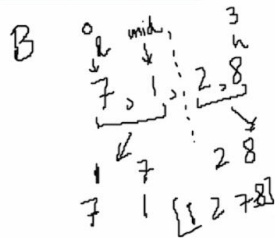
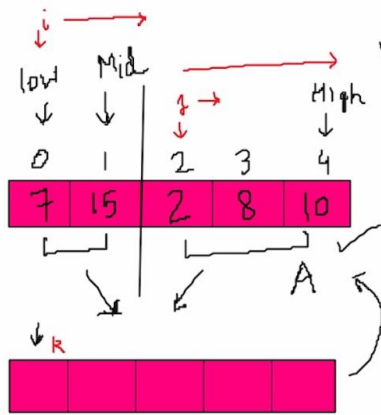


```

Void Merge(A[], mid, lo, hi) {
    int i, j, k; int B[hi+1];
    i = lo, j = mid + 1, k = lo;
    while (i <= mid && j <= hi) {
        if (A[i] < A[j]) {
            B[k] = A[i];
            i++; k++;
        }
        else {
            B[k] = A[j];
            j++; k++;
        }
    }
    while (i <= mid) {
        B[k] = A[i];
        k++; i++;
    }
    while (j <= hi) {
        B[k] = A[j];
        k++; j++;
    }
}
    
```



Merging in a Single Array



Void Merge(A[], mid, lo, hi)

```
int i, t, k; int B[mid+1]
i = lo, j = mid+1, k = lo
while (i <= mid & j <= hi) {
    if (A[i] < A[j]) {
        B[k] = A[i];
        i++; k++;
    }
    else {
        B[k] = A[j];
        j++; k++;
    }
}
```

```
while (i <= mid) {
    B[k] = A[i];
    k++; i++;
}
```

```
while (j <= hi) {
    B[k] = A[j];
    k++; j++;
}
```

Recursive Merge Sort

Void MS(int A[], int l, int h)

```
if (l < h) {
    mid = (l+h)/2;
    MS(A, l, mid);
    MS(A, mid+1, h);
    Merge(A, l, mid, h);
}
```

copy all remaining elements from A to C

from B to C