

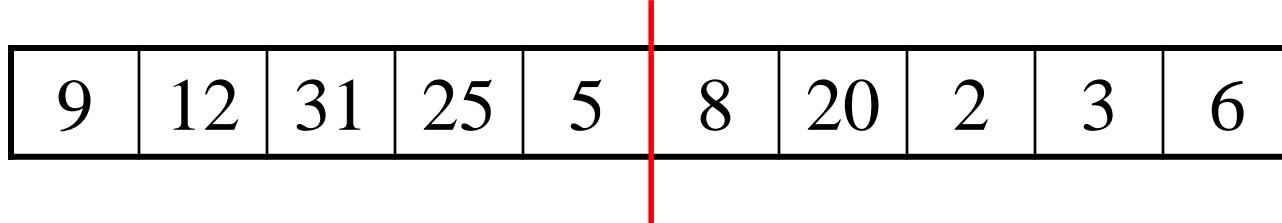
BM267 - Introduction to Data Structures

8. Mergesort

Ankara University
Computer Engineering Department
Bulent Tugrul

Mergesort

- Also a divide and conquer algorithm.
- Let's see how the mergesort works with a small array.
 - The first step divides the array into two equally sized groups.



- After dividing we have two smaller arrays with five elements in each array.

Mergesort

- Assume that we sort these arrays somehow with recursive calls.

5	9	12	25	31
---	---	----	----	----

2	3	6	8	20
---	---	---	---	----

- We have to merge these arrays to get the solution.

2	3	5	6	8	9	12	20	25	31
---	---	---	---	---	---	----	----	----	----

Mergesort

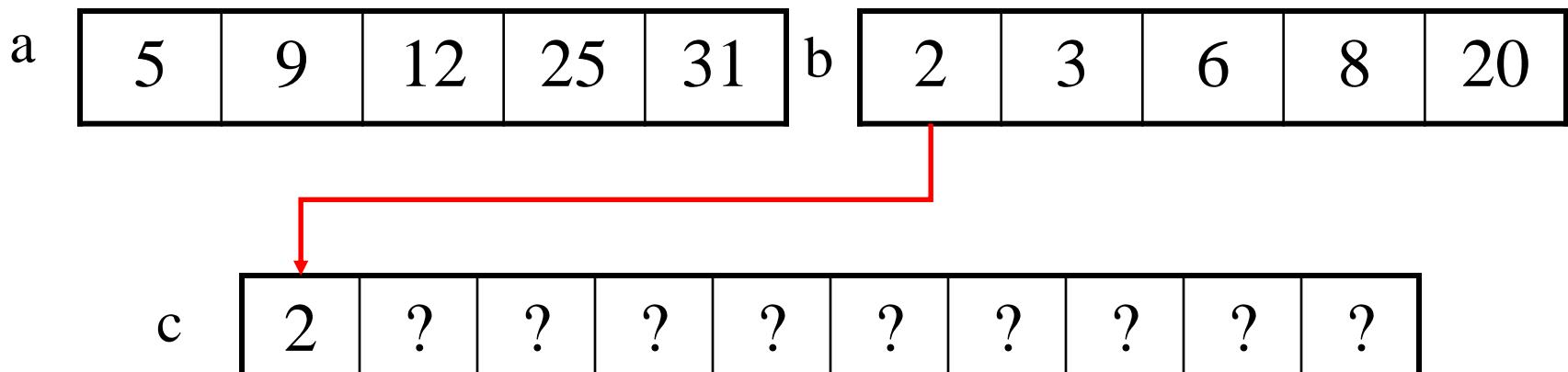
- As shown, the mergesort algorithm divides the array its midpoint, sorts the two half-arrays by recursive call.
- The stopping case for mergesort is when the array to be sorted consists of one element.
- If there is more than one element, mergesort carries out these steps:
 - Calculate the pieces of the two pieces of the array.
 - The last element of the first half, m is $(r + 1) / 2$
 - The first element of the second half m+1
 - Use recursive calls to sort the two halves. (l, m) and (m+1, r)
 - Finally, activate merge function to combine the two sorted halves.

Mergesort

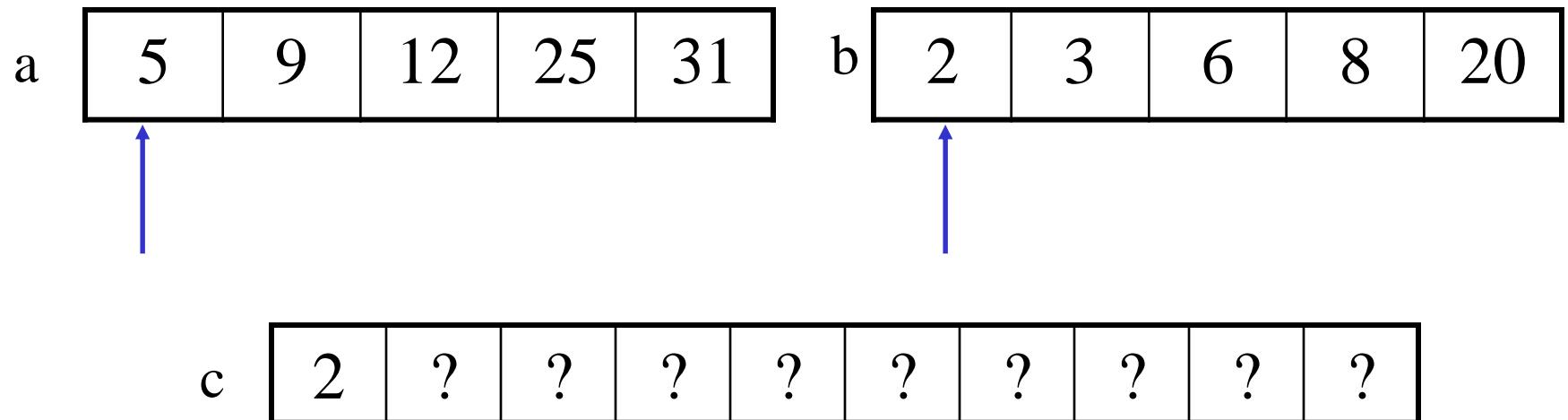
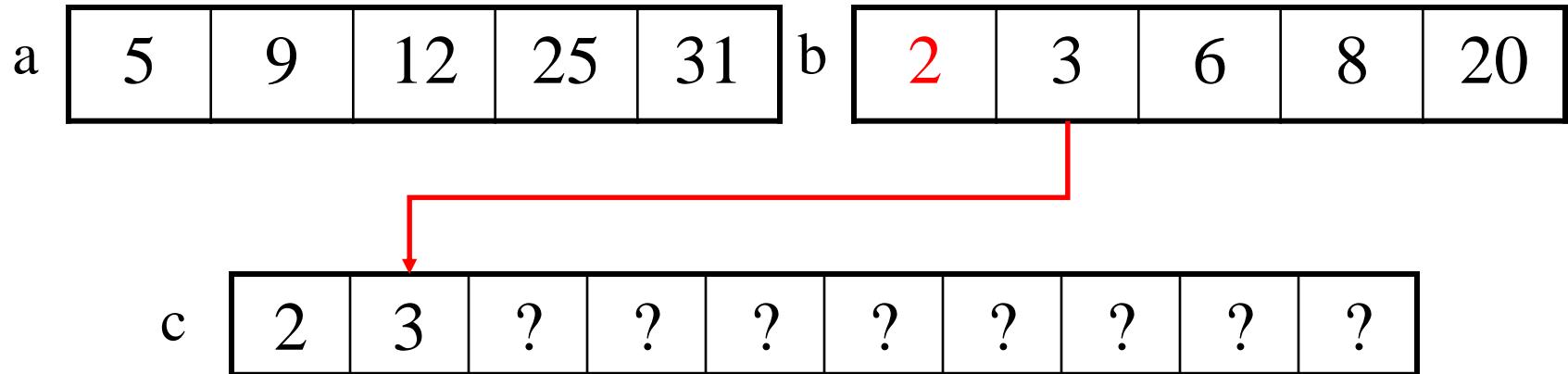
```
void mergesort(Item a[], int l, int r) {  
    int m = (l+r)/2;  
    if (r <= l) return;  
    mergesort(a, l, m);  
    mergesort(a, m+1, r);  
    merge(a, l, m, r);  
}
```

Merge Function

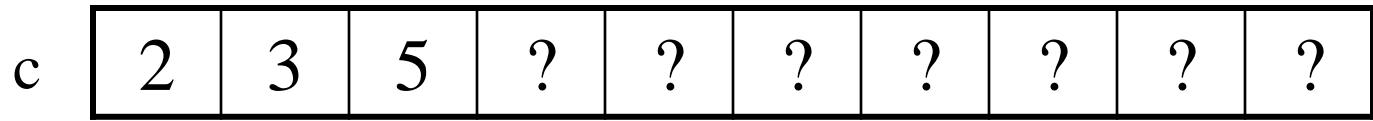
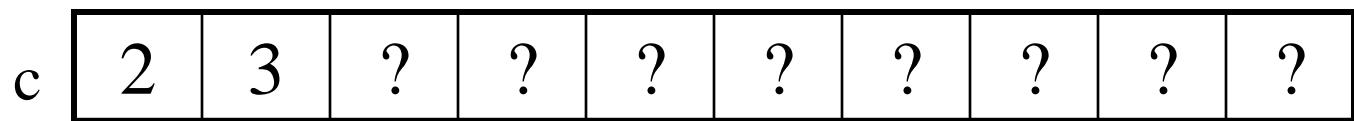
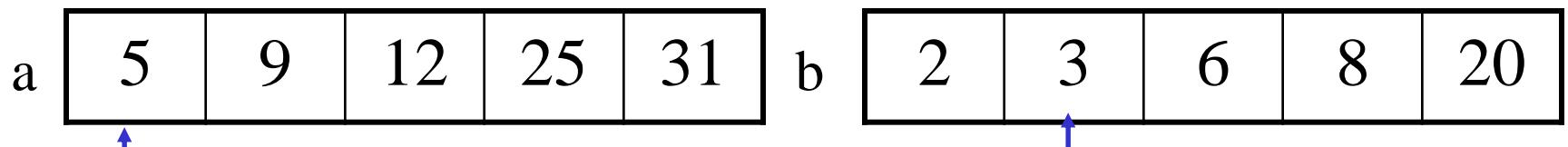
- Let's us suppose that we have two ordered arrays $a[0] \dots a[N-1]$ and $b[0] \dots b[M-1]$ of integers that we wish to merge into a third array $c[0] \dots c[N+M-1]$.
- The obvious strategy is to choose the smallest remaining element from a and b .



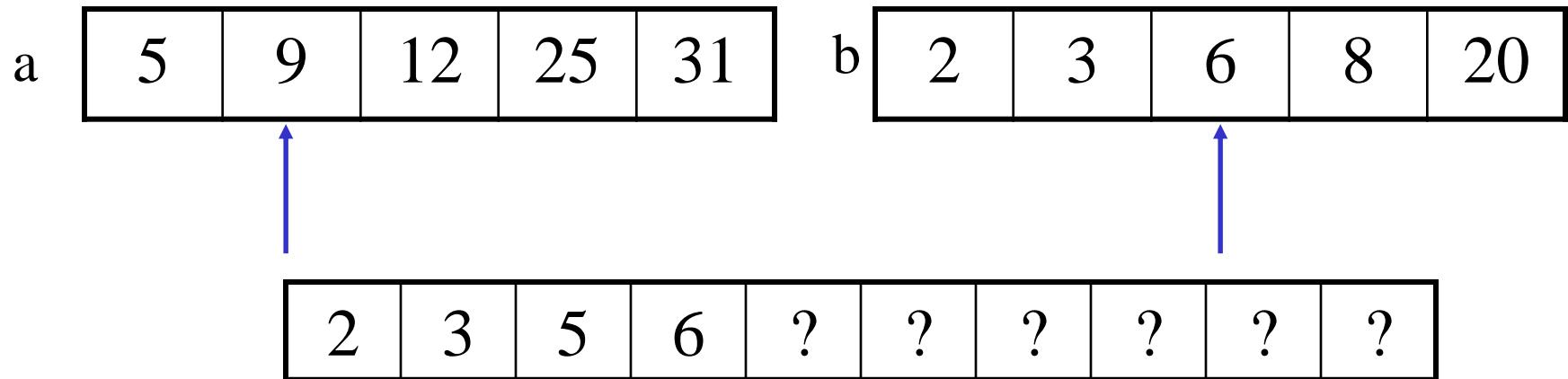
Merge Function



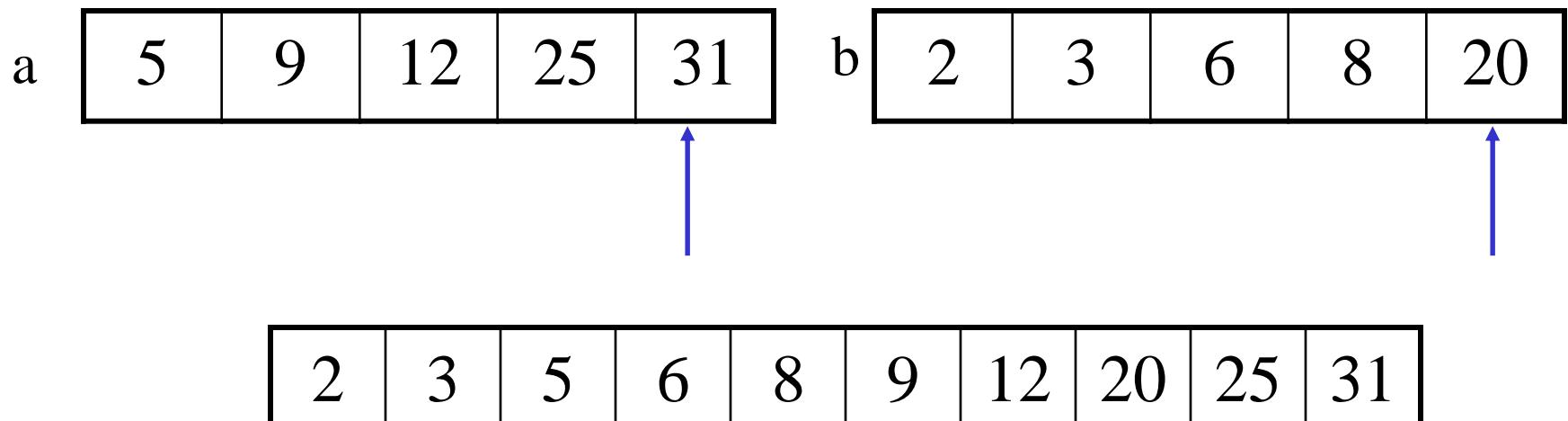
Merge Function



Merge Function



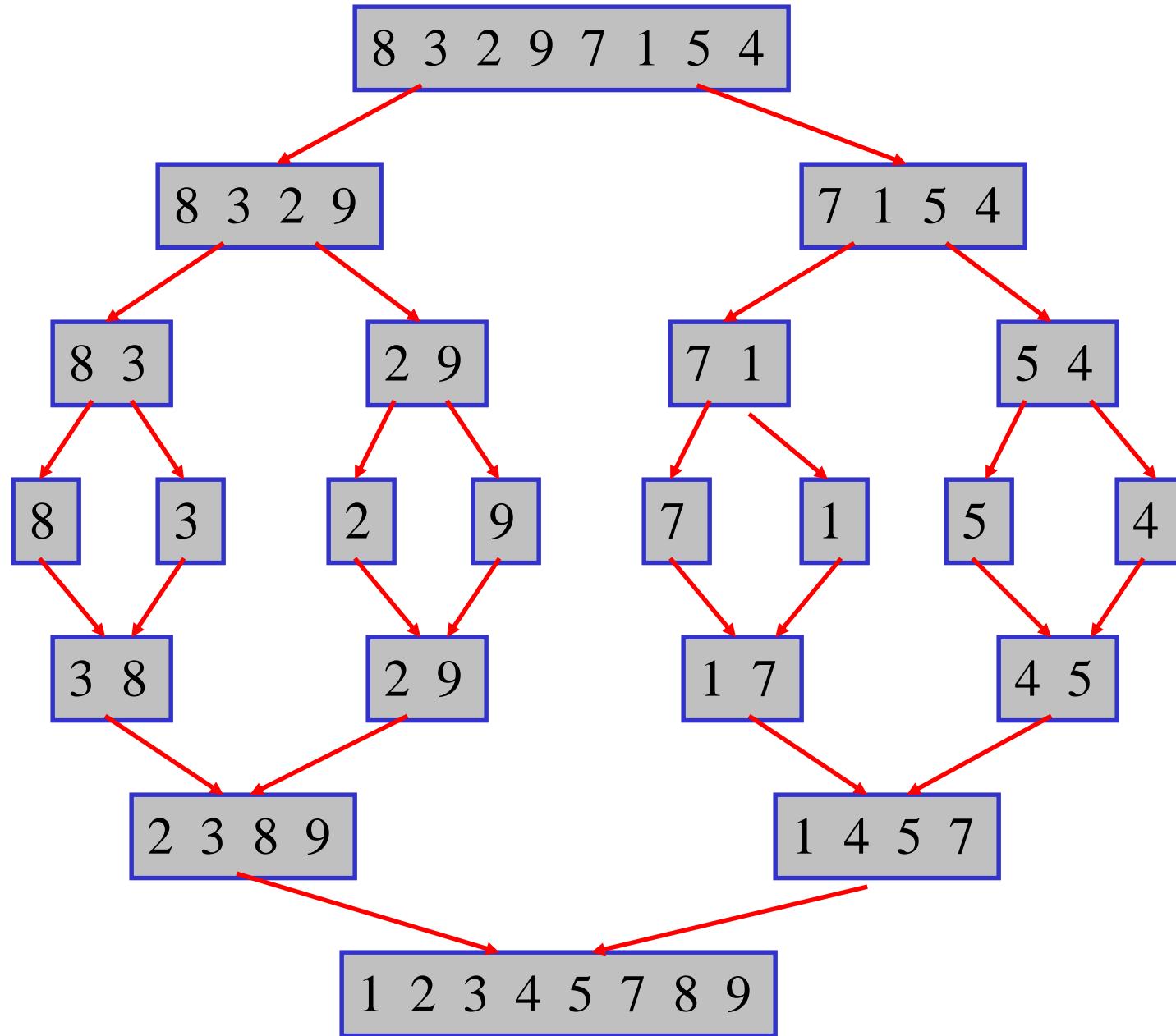
Finally



Merge Function

```
mergeAB(Item c[], Item a[], int N, Item b[], int M )  
{    int i, j, k;  
    for (i = 0, j = 0, k = 0; k < N+M; k++)  
    {  
        if (i == N) { c[k] = b[j++]; continue; }  
        if (j == M) { c[k] = a[i++]; continue; }  
        c[k] = ((a[i] < b[j])) ? a[i++] : b[j++];  
    }  
}
```

Mergesort



Analysis of Mergesort

- Mergesort requires about $n \log n$ comparisons to sort n elements.
 - $T(n) = 2T(n/2) + n-1$ (for simplicity assume $n = 2^k$)
- Mergesort uses extra space proportional n .
- Mergesort is stable, if the underlying merge is stable.