Activation Records, Divide and Conquer, Recursive Algorithms

Ira A. Fulton Schools of Engineering
Arizona State University

Sorting

- Sorting is the process of arranging a sequence of objects into order (either increasing or decreasing)

- Mergesort is a sorting algorithm

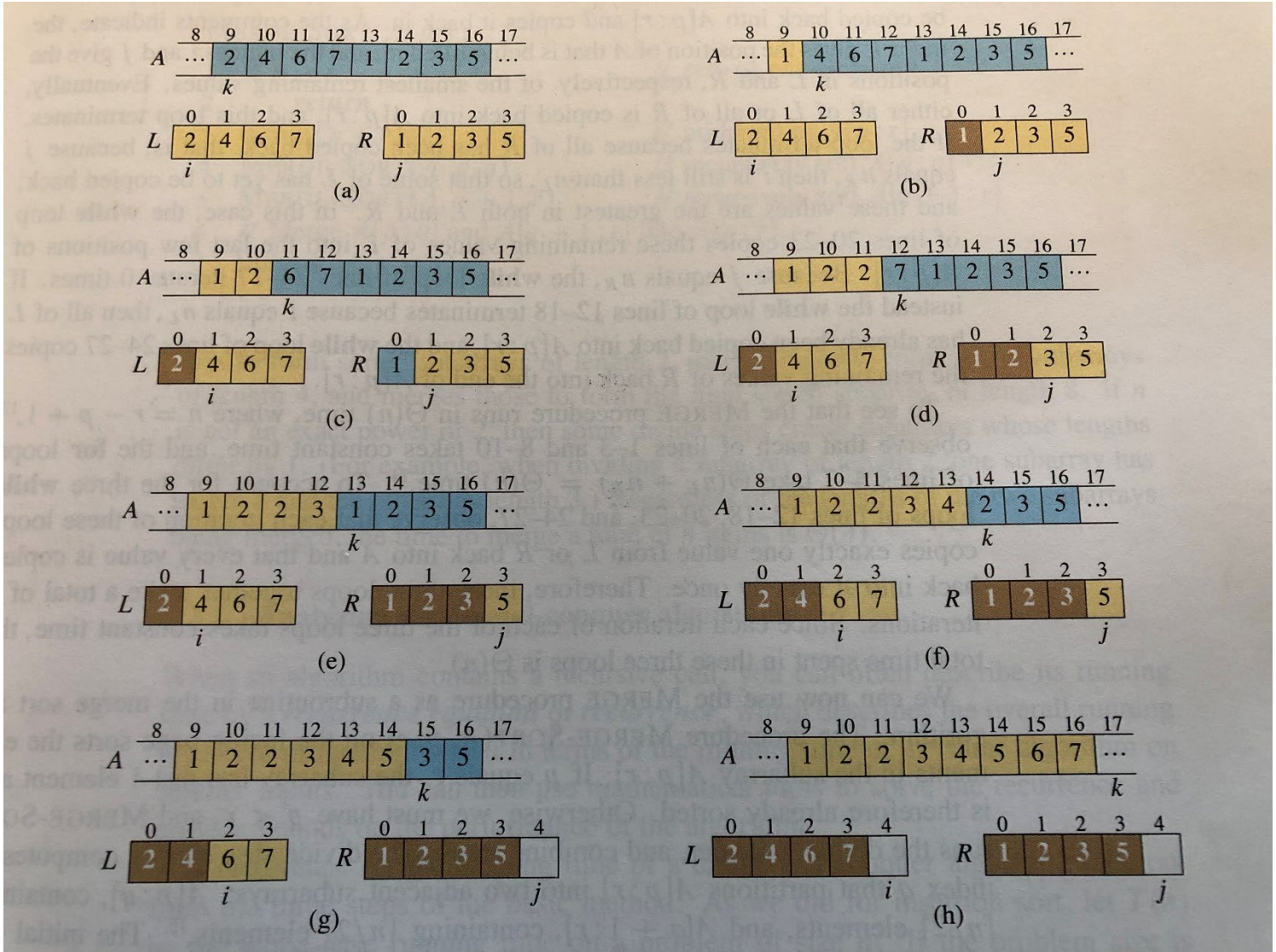
Divide and Conquer, Merge Sort, Recursion

- **Divide and Conquer**
  - **Divide**: break an instance of a problem into several smaller instances of the same problem
  - **Conquer**: Compute solutions for the smaller instances
  - **Combine**: Use the solutions to the smaller instances to obtain the solution of the original instance
- **Merge Sort**: A sorting algorithm that uses divide and conquer.
- **Recursive Algorithm**: An algorithm that calls itself.

Merge

```
Merge(A, p, q, r)                    // A[p:q]and A[q+1:r] are already sorted
01: nL = q - p + 1;                  // nL is the length of A[p:q]
02: nR = r - q;                      // nR is the length of A[q+1:r]
03: new L[0:nL-1]; new R[0:nR-1];    // two new arrays
04: for i=0 to nL-1                  //
05:   L[i] = A[p+i];                // L[0:nL-1] = A[p:q]
06: for j=0 to nR-1                  //
07:   R[j] = A[q+j+1];               // R[0:nR-1] = A[q+1:r]
08: i = 0;                           // i points to the start of L[]
09: j = 0;                           // j points to the start of R[]
10: k = p;                           // k points to the start of A[]
12: while i < nL and j < nR          //
13:   if L[i] <= R[j];               //
14:     A[k] = L[i];                 // L[i] is k-th smallest
15:     i = i+1;                     //
16:   else A[k] = R[j];              // R[j] is k-th smallest
17:     j = j+1;                     //
18:   k = k+1;                       //
20: while i < nL                     // 20-23 and 24-27 are mutually exclusive
21:   A[k] = L[i];                   // copy rest of L[] to A[]
22:   i = i+1;                       //
23:   k = k+1;                       //
24: while j < nR                     //
25:   A[k] = R[j];                   // copy rest of R[] to A[]
26:   j = j+1;                       //
27:   k = k+1;                       //
```

Example of Merge



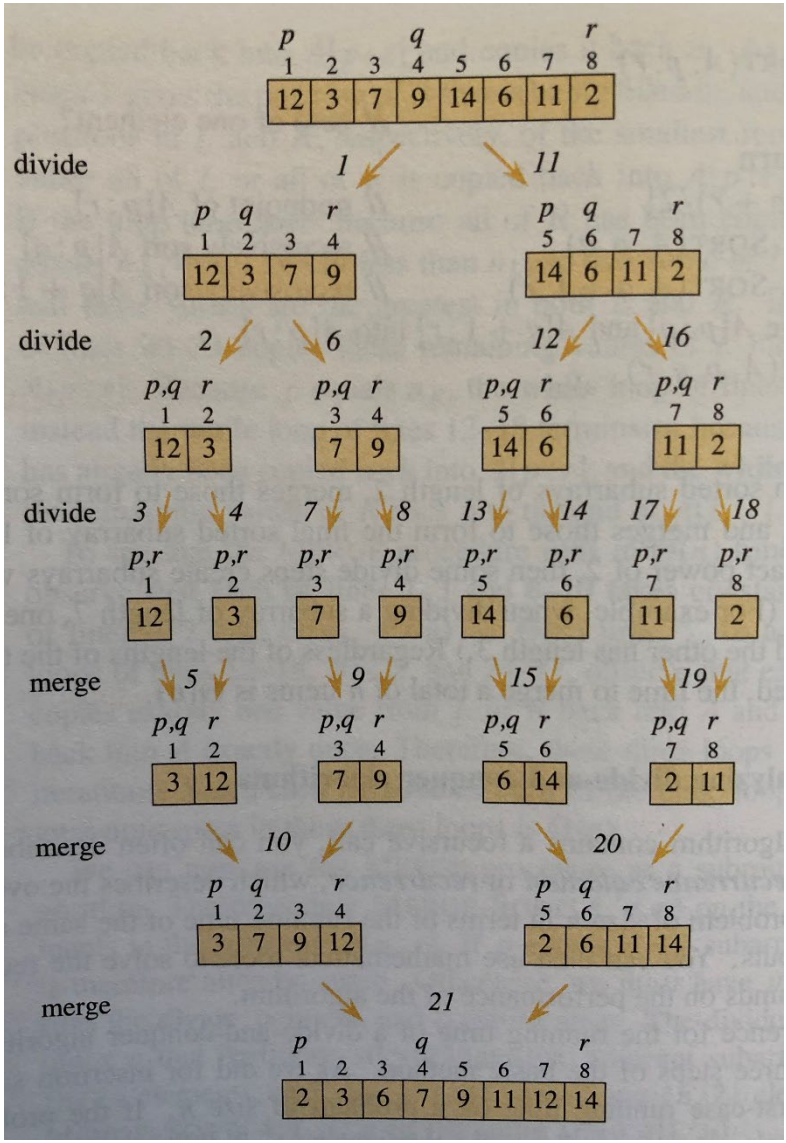
Mergesort

```
Merge-Sort(A, p, r)
1: if p >= r
2:    return;
3: q = (p+r)/2;
4: Merge-Sort(A, p, q);
5: Merge-Sort(A, q+1, r);
7: Merge(A, p, q, r);
```

Note for line 1:
When p=r, we have 1 element.
When p>r, we have 0 element.
We may call Merge-Sort with p>r.

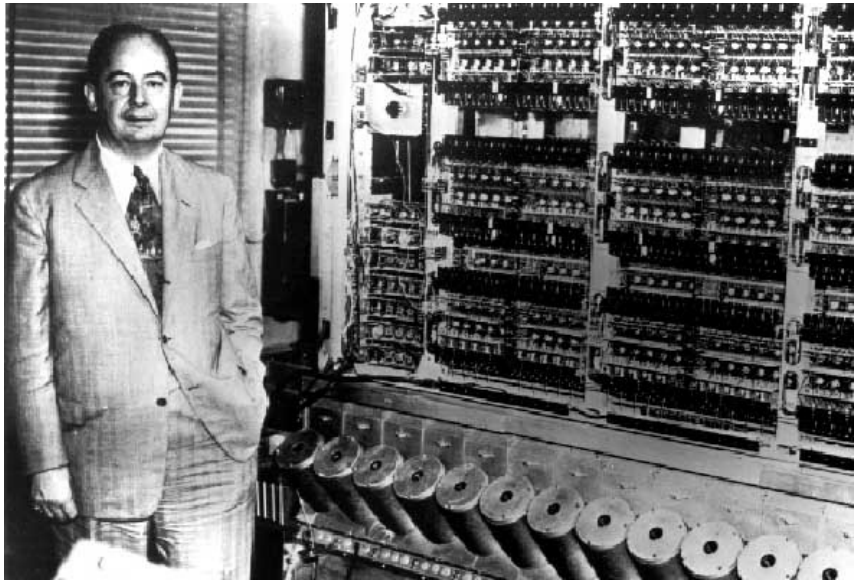
Note for line 3:
q is the floor of (p+r)/2.
Integer division always the floor.

Note for lines 4-5:
Conquer left, then conquer right.



Mergesort

- Merge sort was invented by John von Neumann in 1945
- Divide: break A[p:r] into A[p:q] and A[q+1:r], where q=(p+r)/2
- Conquer: sort A[p:q] and A[q+1:r], using merge sort
- Combine: merge A[p:q] and A[q+1:r]



John von Neumann

Activation Records and Recursive Calls

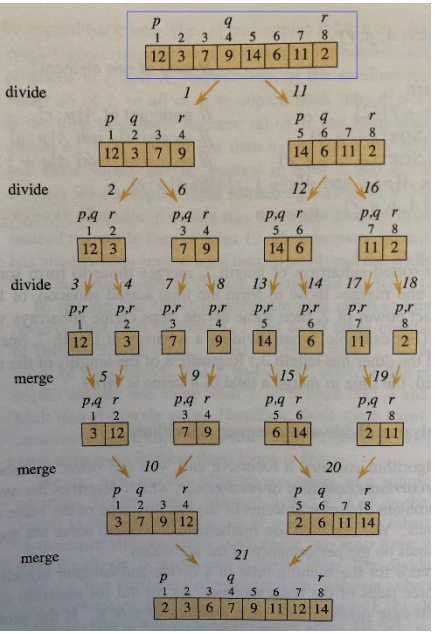
- Run-time stack
- Activation record (AR)
  - One activation record per function call
  - The activation records are managed via a stack
  - The AR of the callee is on top of the AR of the caller
  - When a function completes, its AR is popped off
- Recursive vs iterative algorithms

Activation Records and Recursive Calls

- For each function call, there is an activation record
- The activation record reserves space for the program, the variables, and intermediate results

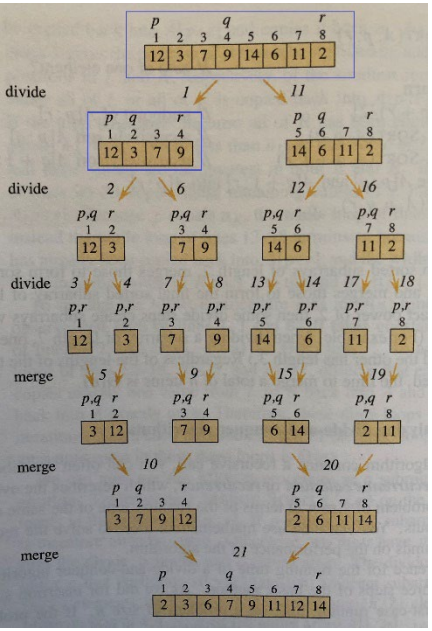
- The activation records form the run-time stack
- When a function is called, we push an AR to the stack.
- When a function is completed, we pop the AR off.

Activation Records and Recursive Calls

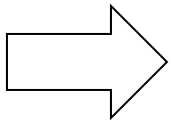


Merge-Sort(A, 1, 8)

Activation Records and Recursive Calls



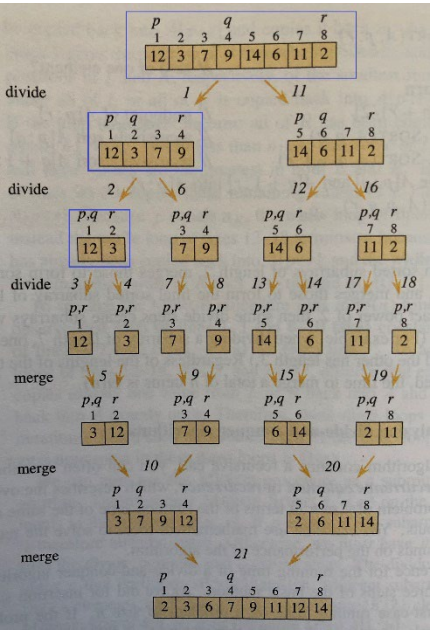
Merge-Sort(A, 1, 8)



Merge-Sort(A, 1, 4)

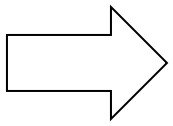
Merge-Sort(A, 1, 8)

Activation Records and Recursive Calls



Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

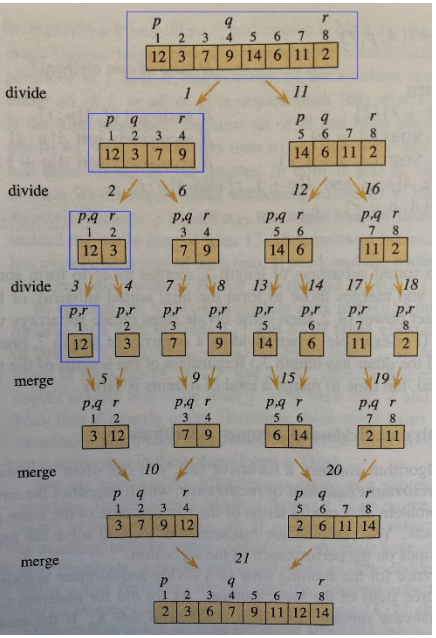


Merge-Sort(A, 1, 2)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

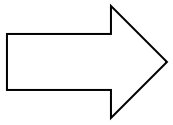
Activation Records and Recursive Calls



Merge-Sort(A, 1, 2)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)



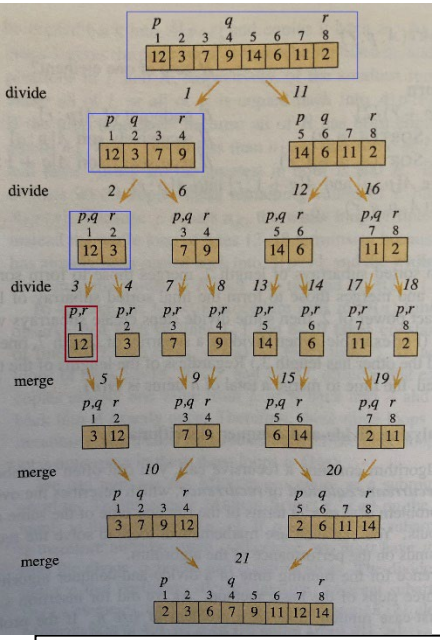
Merge-Sort(A, 1, 1)

Merge-Sort(A, 1, 2)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

Activation Records and Recursive Calls

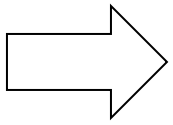


Merge-Sort(A, 1, 1)

Merge-Sort(A, 1, 2)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

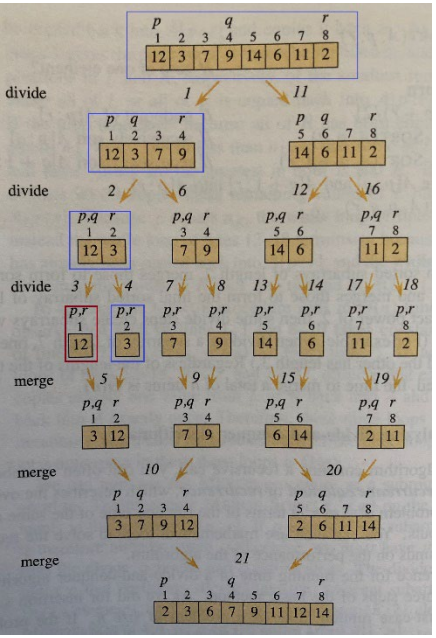


Merge-Sort(A, 1, 2)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

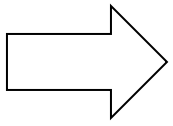
Activation Records and Recursive Calls



Merge-Sort(A, 1, 2)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)



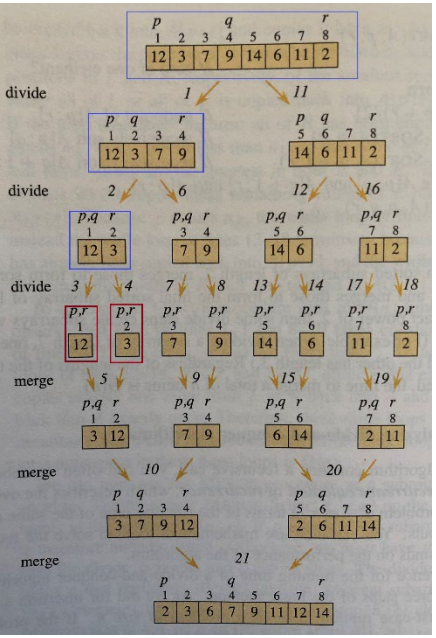
Merge-Sort(A, 2, 2)

Merge-Sort(A, 1, 2)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

Activation Records and Recursive Calls

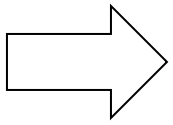


Merge-Sort(A, 2, 2)

Merge-Sort(A, 1, 2)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

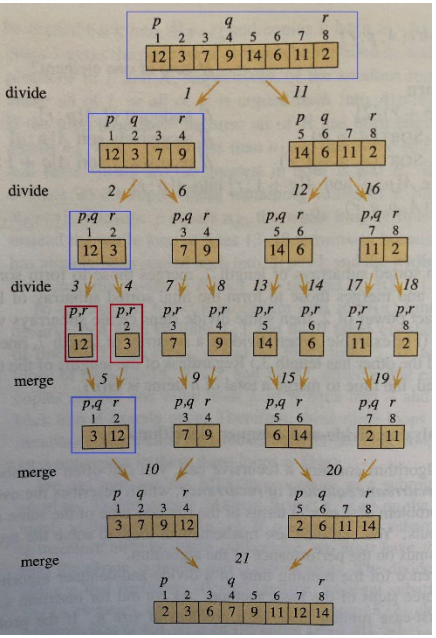


Merge-Sort(A, 1, 2)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

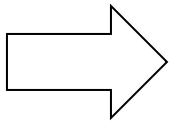
Activation Records and Recursive Calls



Merge-Sort(A, 1, 2)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)



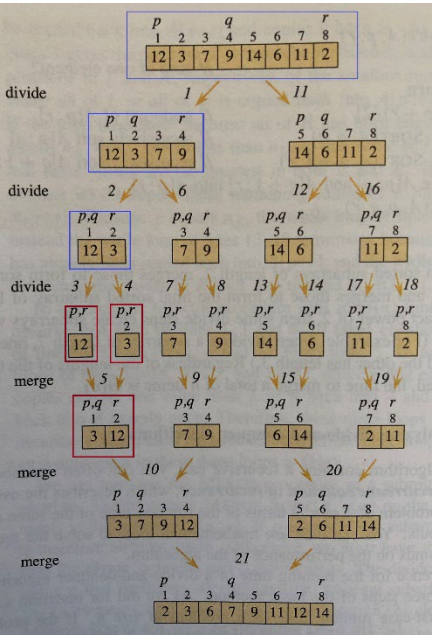
Merge(A, 1, 1, 2)

Merge-Sort(A, 1, 2)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

Activation Records and Recursive Calls

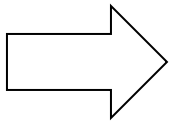


Merge(A, 1, 1, 2)

Merge-Sort(A, 1, 2)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

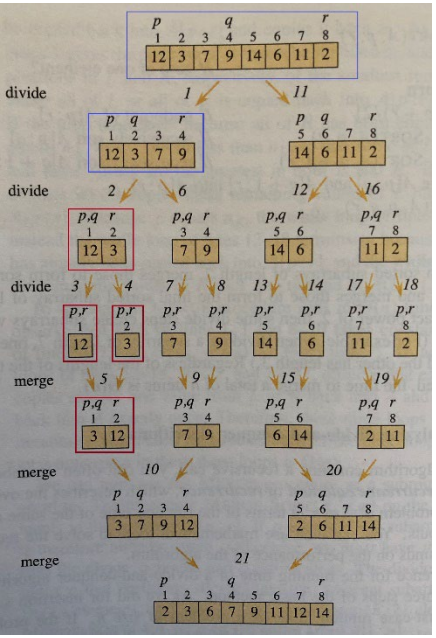


Merge-Sort(A, 1, 2)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

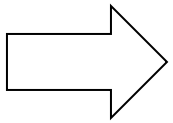
Activation Records and Recursive Calls



Merge-Sort(A, 1, 2)

Merge-Sort(A, 1, 4)

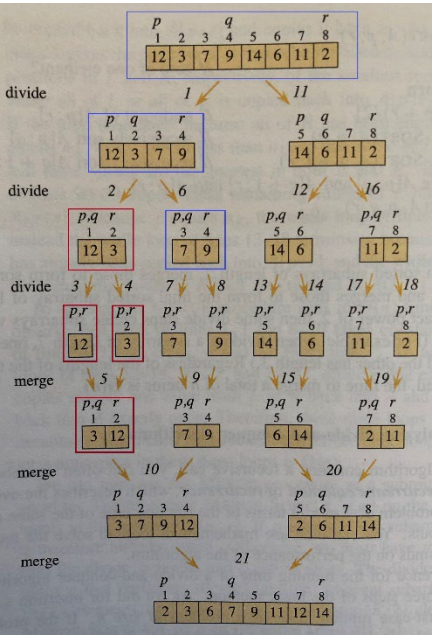
Merge-Sort(A, 1, 8)



Merge-Sort(A, 1, 4)

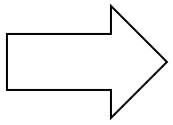
Merge-Sort(A, 1, 8)

Activation Records and Recursive Calls



Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

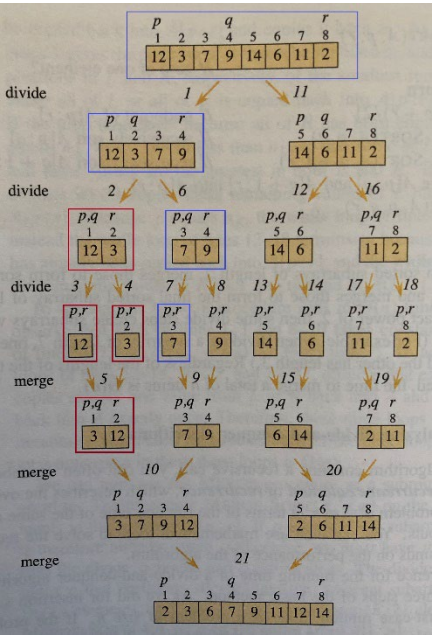


Merge-Sort(A, 3, 4)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

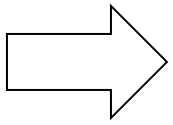
Activation Records and Recursive Calls



Merge-Sort(A, 3, 4)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)



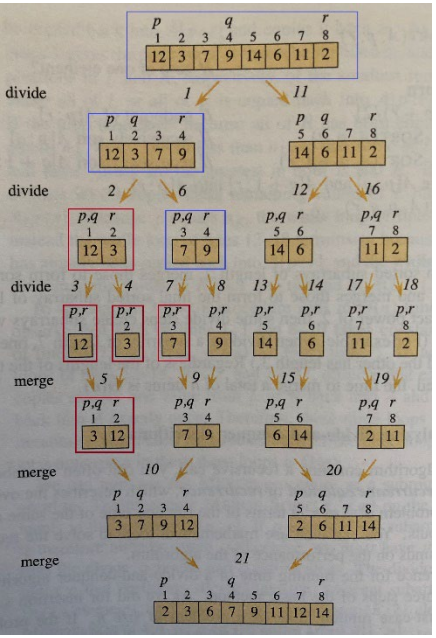
Merge-Sort(A, 3, 3)

Merge-Sort(A, 3, 4)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

Activation Records and Recursive Calls

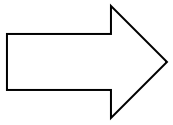


Merge-Sort(A, 3, 3)

Merge-Sort(A, 3, 4)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

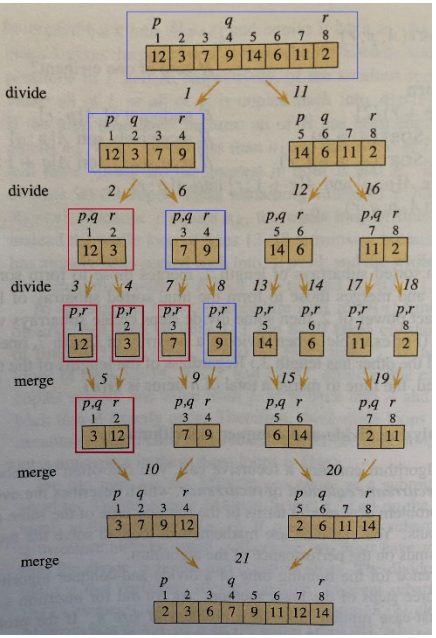


Merge-Sort(A, 3, 4)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

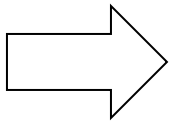
Activation Records and Recursive Calls



Merge-Sort(A, 3, 4)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)



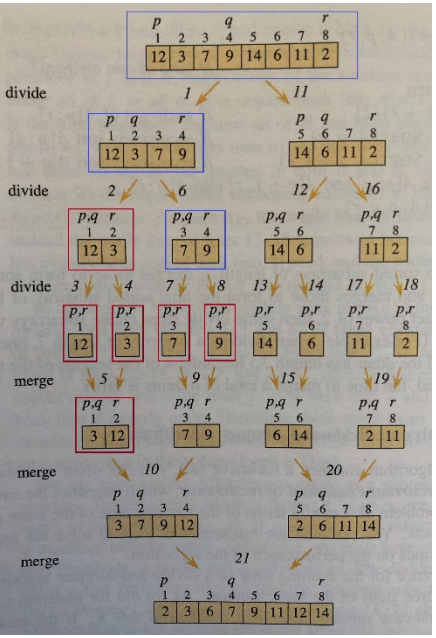
Merge-Sort(A, 4, 4)

Merge-Sort(A, 3, 4)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

Activation Records and Recursive Calls

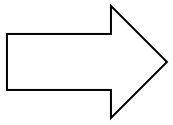


Merge-Sort(A, 4, 4)

Merge-Sort(A, 3, 4)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

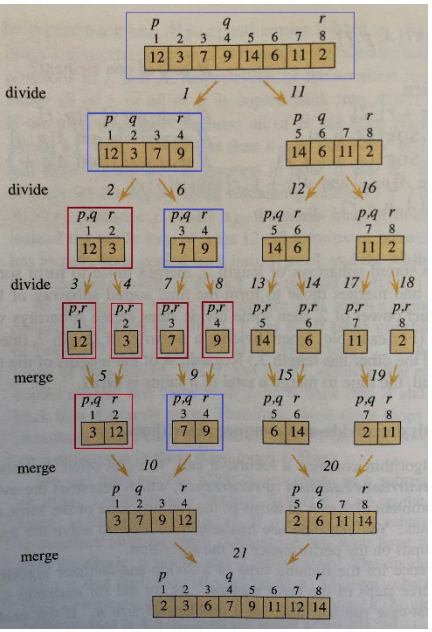


Merge-Sort(A, 3, 4)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

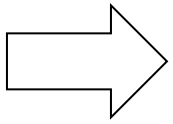
Activation Records and Recursive Calls



Merge-Sort(A, 3, 4)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)



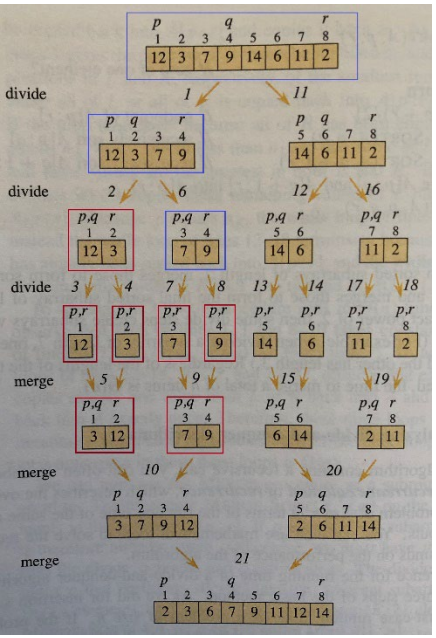
Merge(A, 3, 3, 4)

Merge-Sort(A, 3, 4)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

Activation Records and Recursive Calls

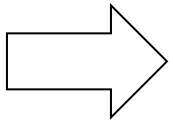


Merge(A, 3, 3, 4)

Merge-Sort(A, 3, 4)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

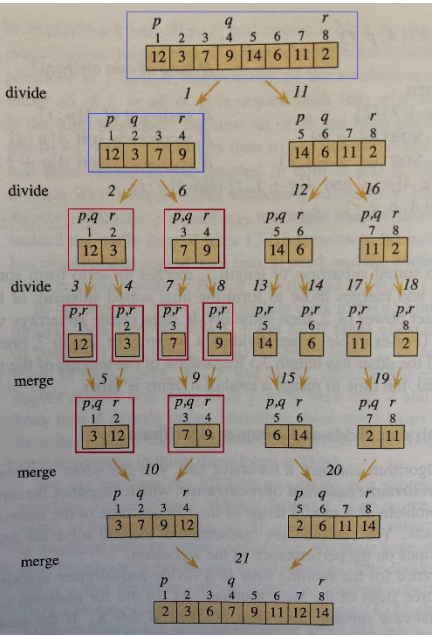


Merge-Sort(A, 3, 4)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

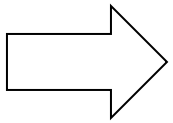
Activation Records and Recursive Calls



Merge-Sort(A, 3, 4)

Merge-Sort(A, 1, 4)

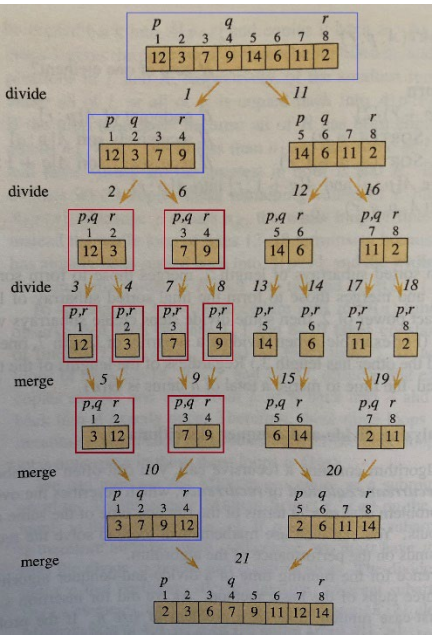
Merge-Sort(A, 1, 8)



Merge-Sort(A, 1, 4)

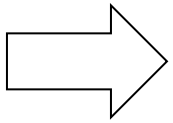
Merge-Sort(A, 1, 8)

Activation Records and Recursive Calls



Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

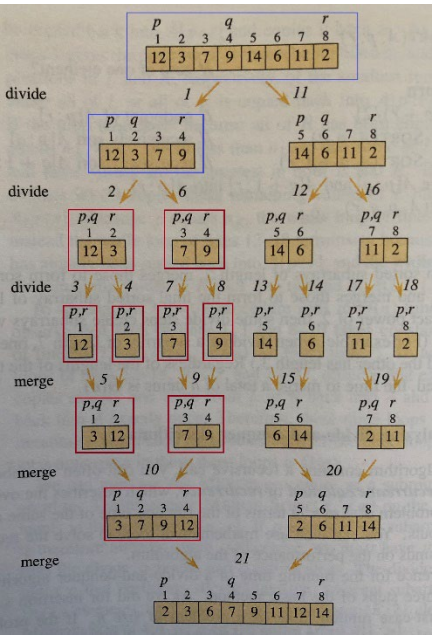


Merge(A, 1, 2, 4)

Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

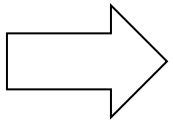
Activation Records and Recursive Calls



Merge(A, 1, 2, 4)

Merge-Sort(A, 1, 4)

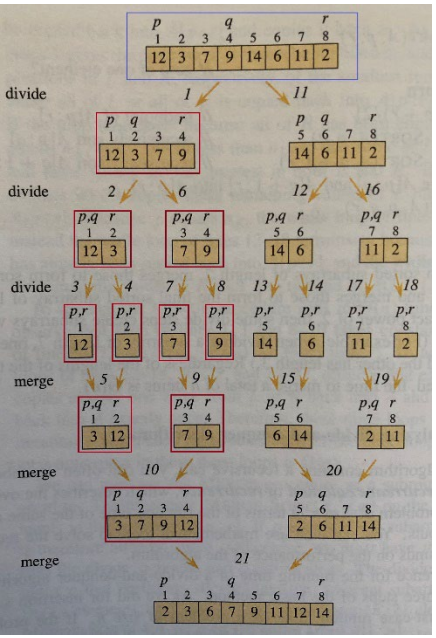
Merge-Sort(A, 1, 8)



Merge-Sort(A, 1, 4)

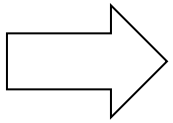
Merge-Sort(A, 1, 8)

Activation Records and Recursive Calls



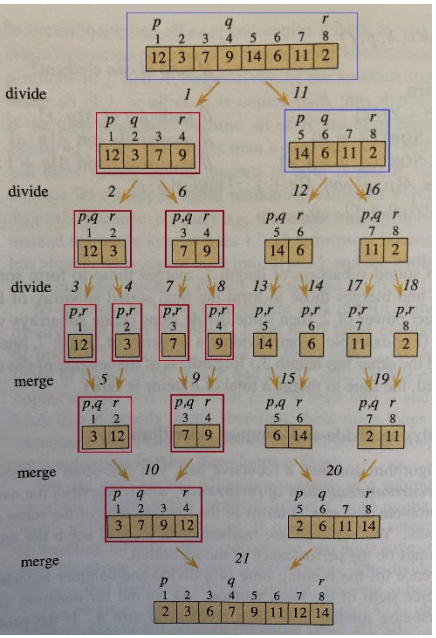
Merge-Sort(A, 1, 4)

Merge-Sort(A, 1, 8)

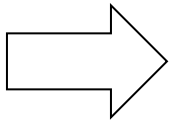


Merge-Sort(A, 1, 8)

Activation Records and Recursive Calls



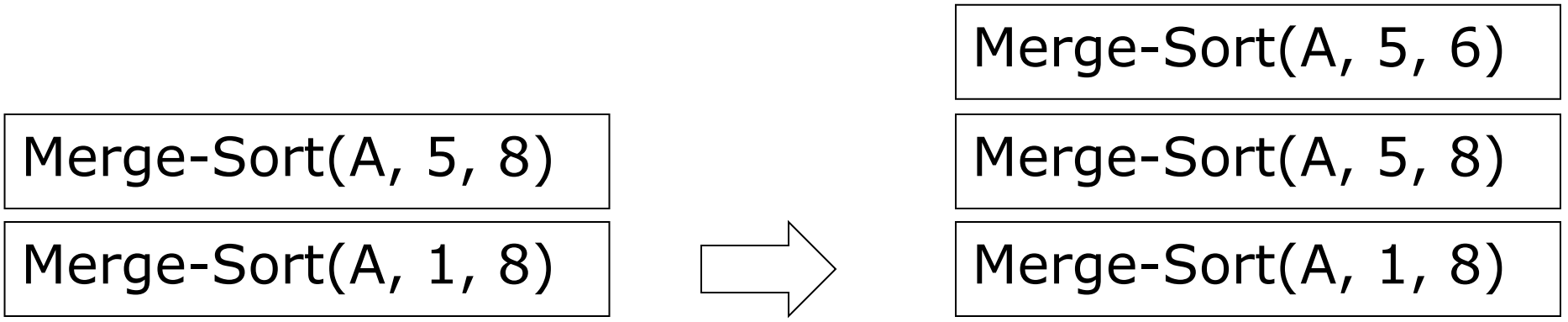
Merge-Sort(A, 1, 8)



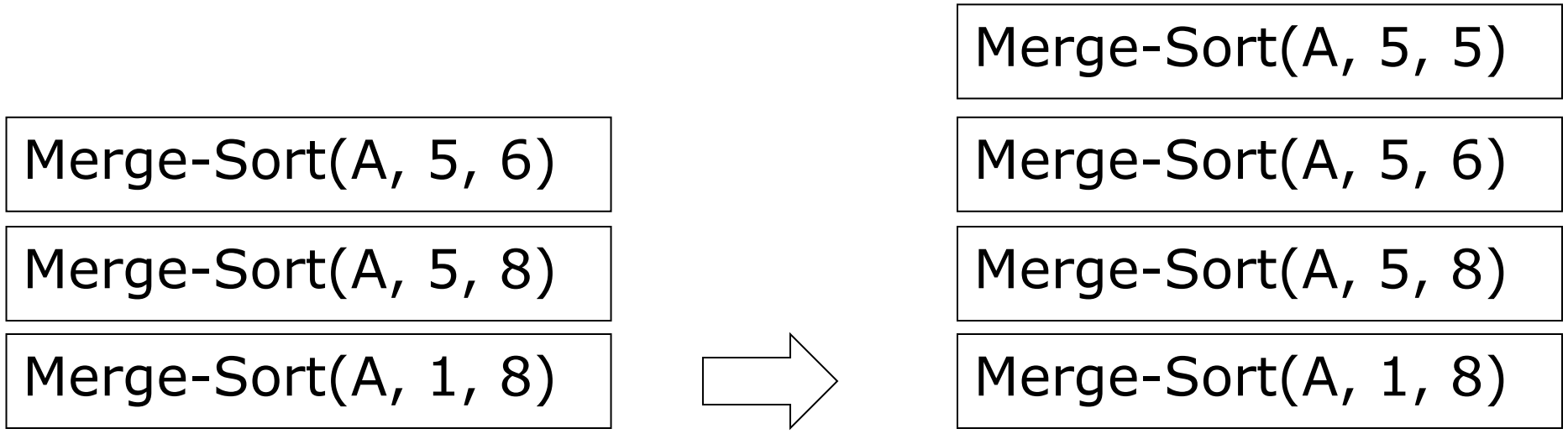
Merge-Sort(A, 5, 8)

Merge-Sort(A, 1, 8)

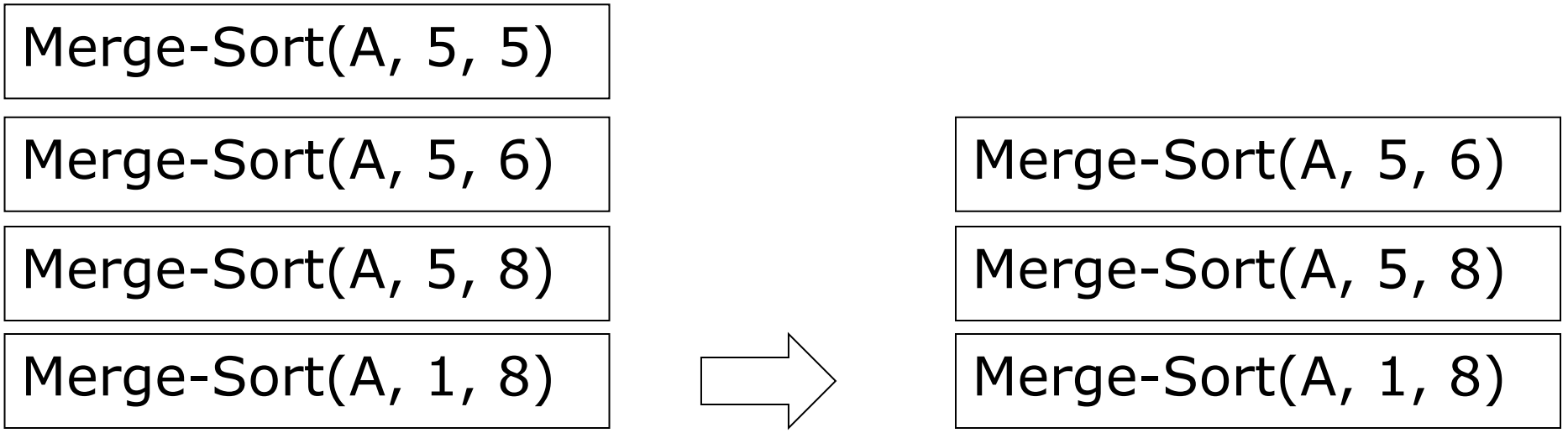
Activation Records and Recursive Calls



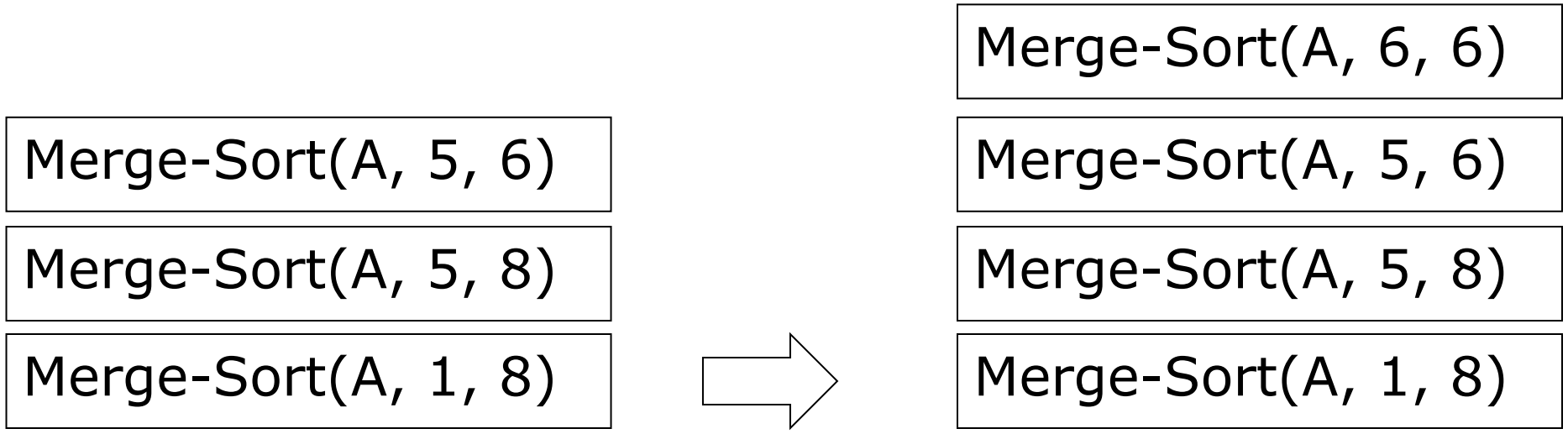
Activation Records and Recursive Calls



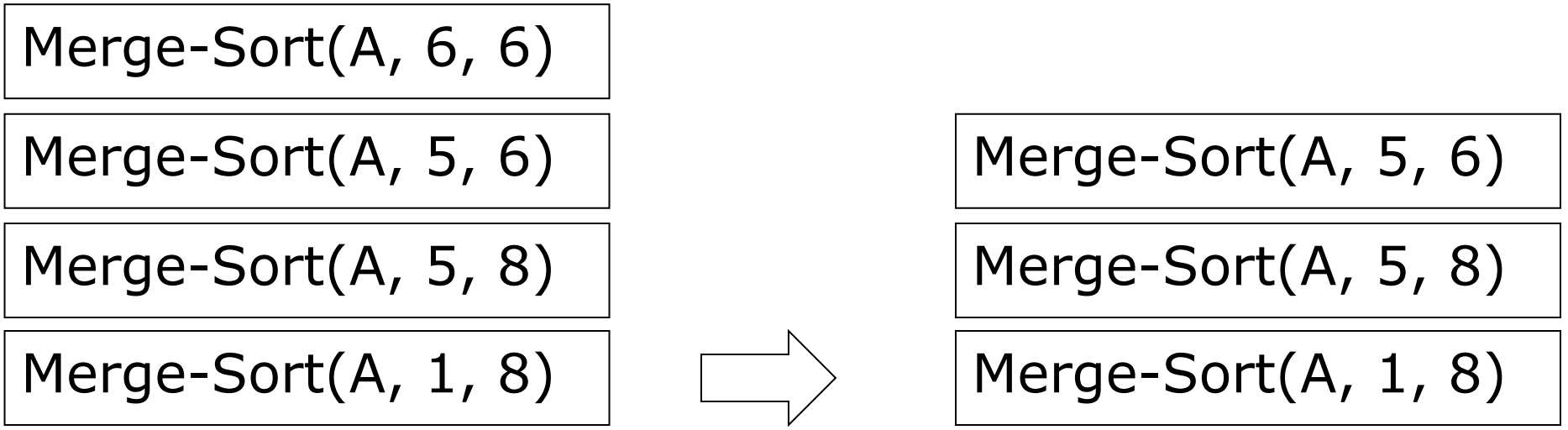
Activation Records and Recursive Calls



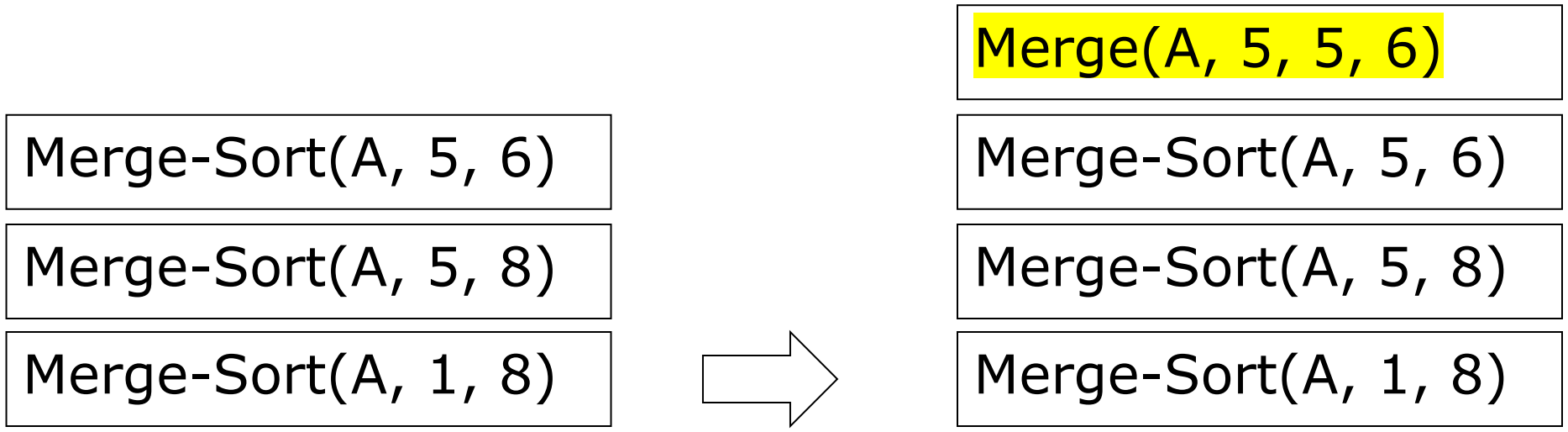
Activation Records and Recursive Calls



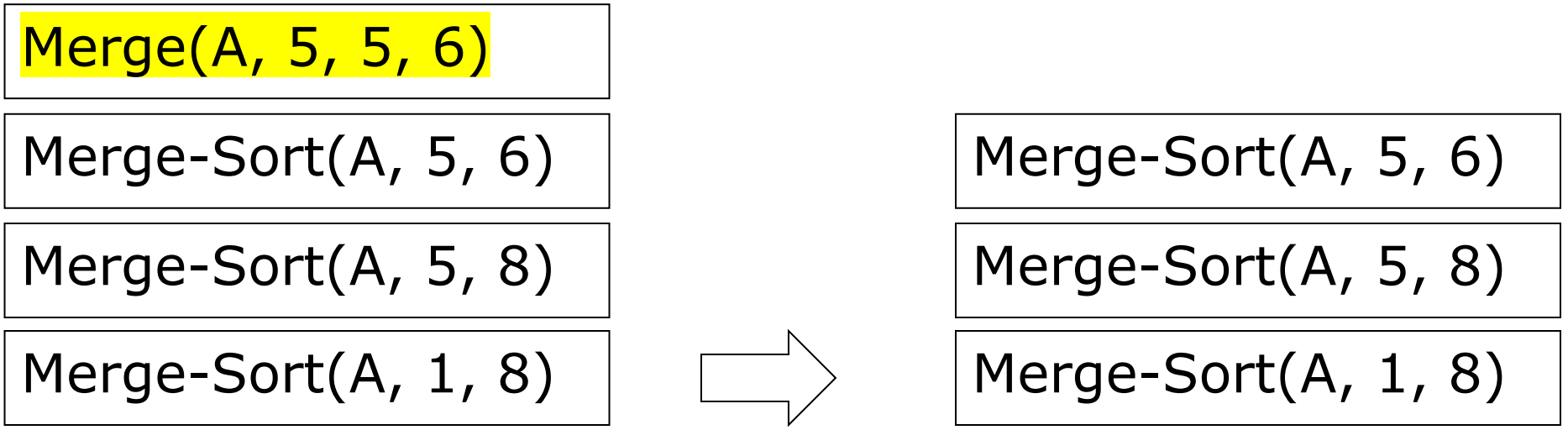
Activation Records and Recursive Calls



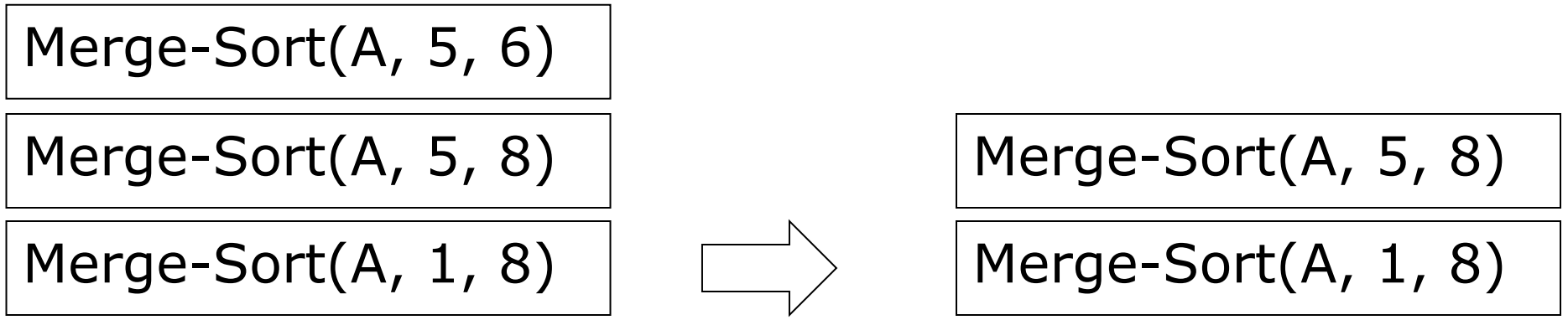
Activation Records and Recursive Calls



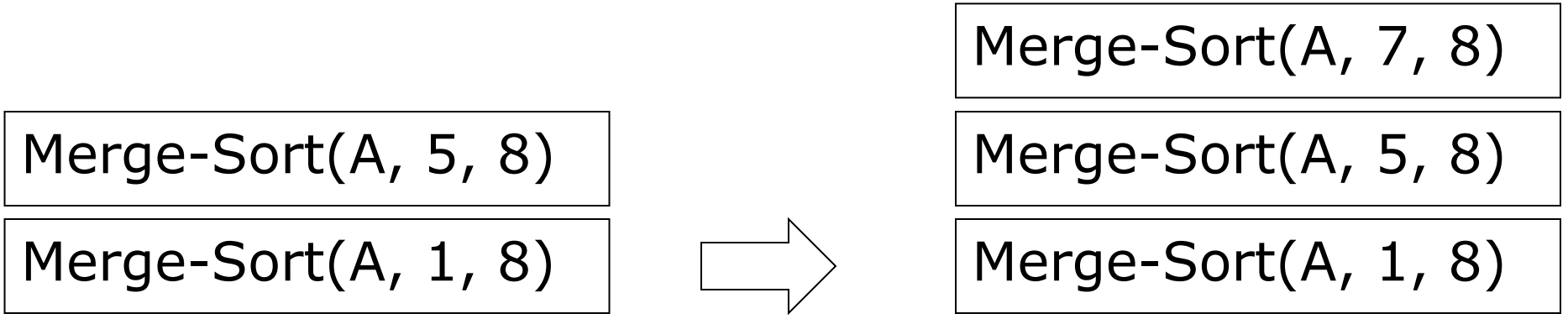
Activation Records and Recursive Calls



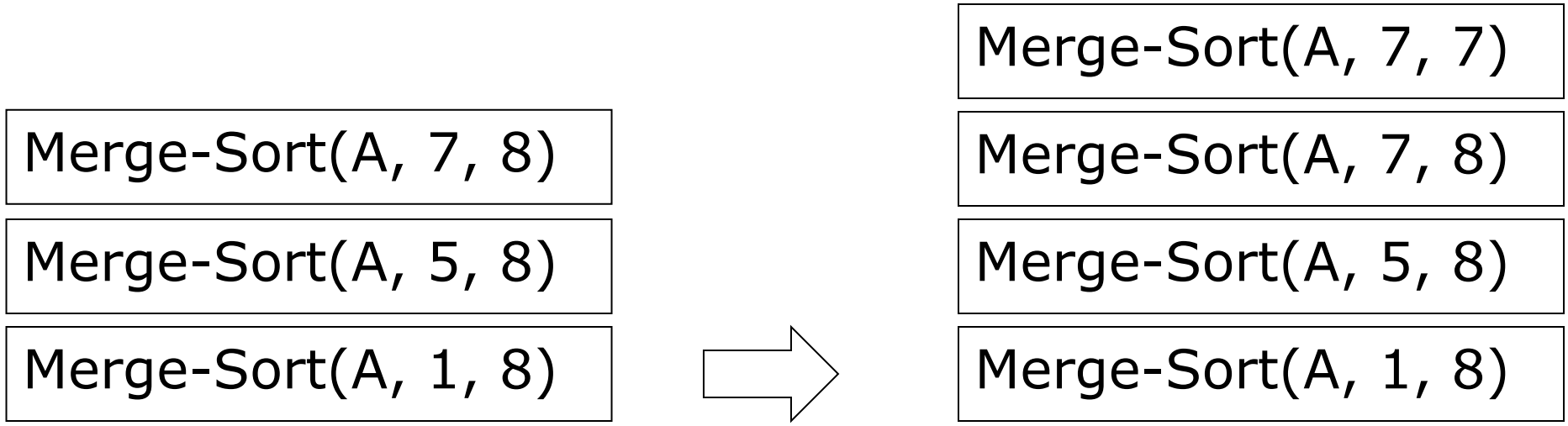
Activation Records and Recursive Calls



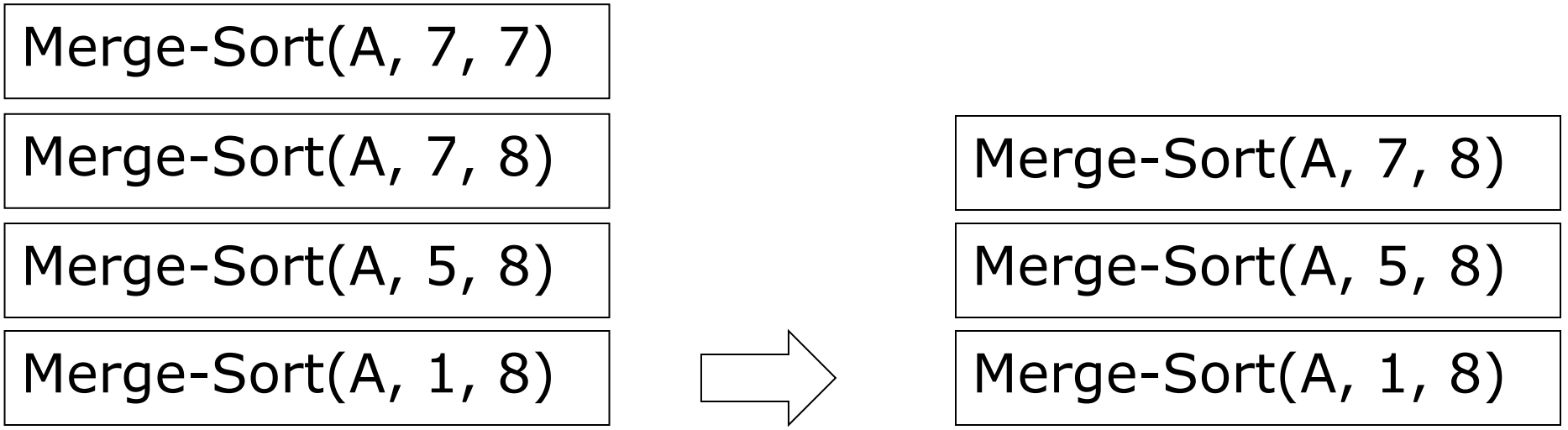
Activation Records and Recursive Calls



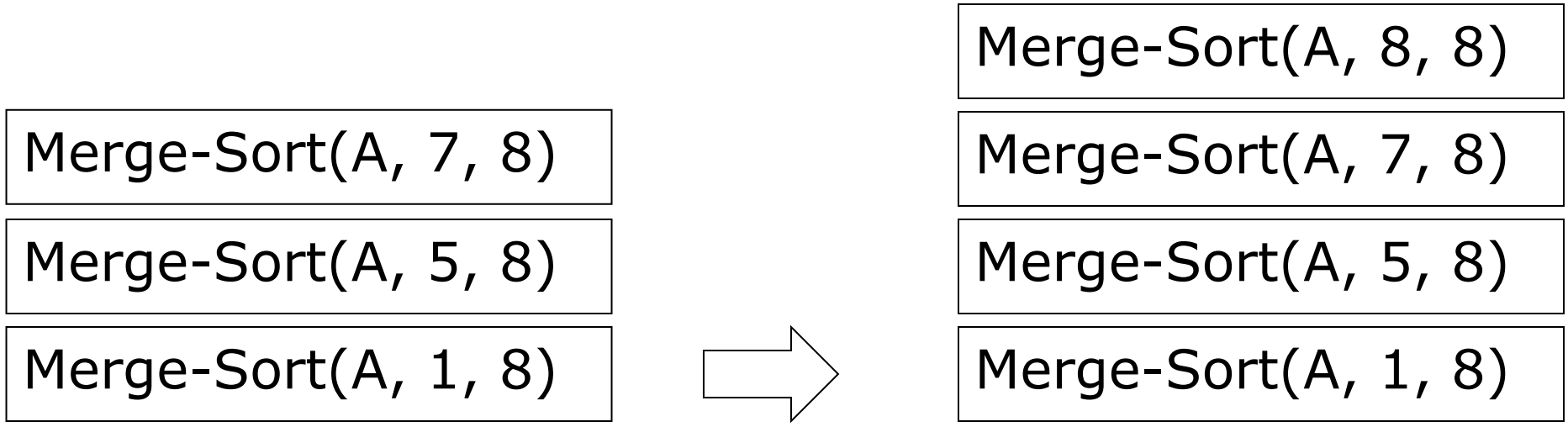
Activation Records and Recursive Calls



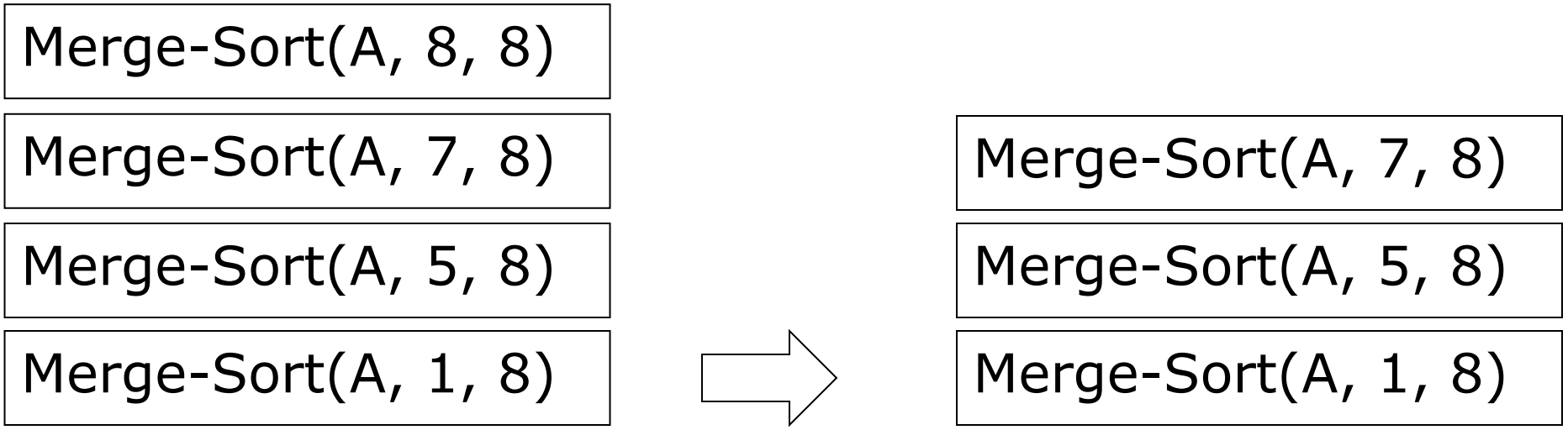
Activation Records and Recursive Calls



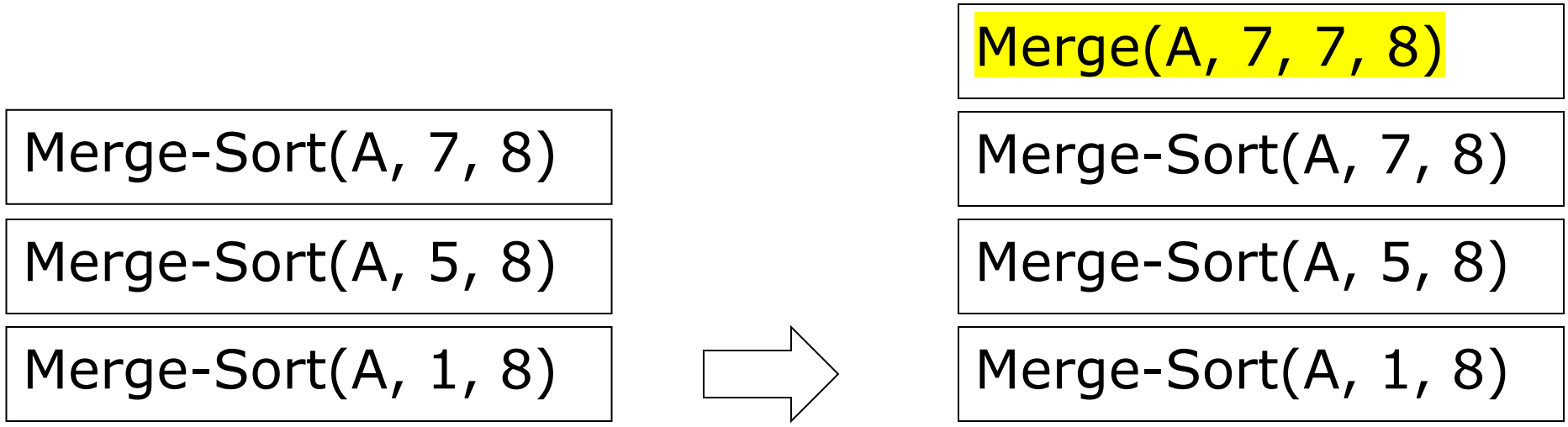
Activation Records and Recursive Calls



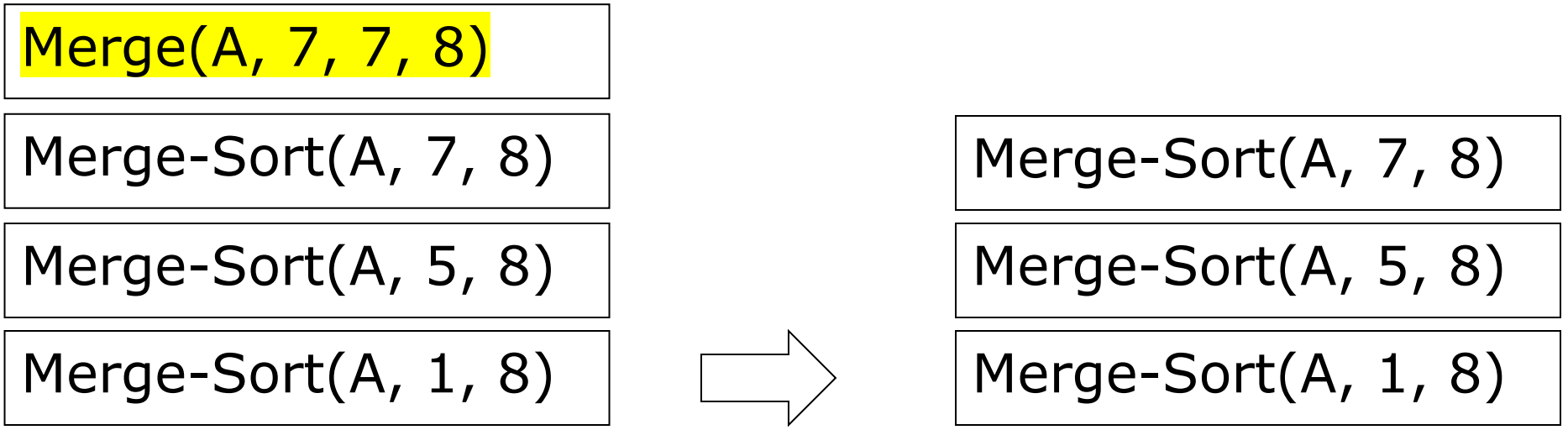
Activation Records and Recursive Calls



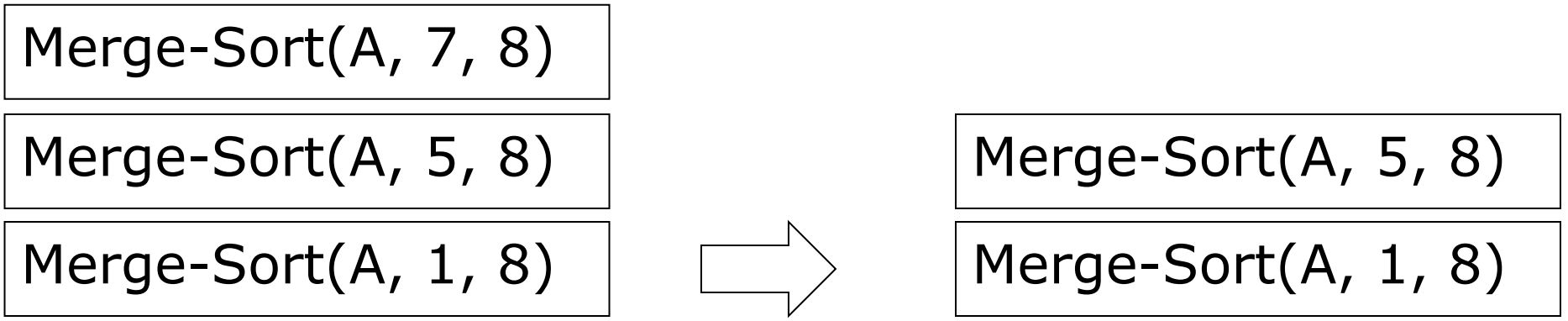
Activation Records and Recursive Calls



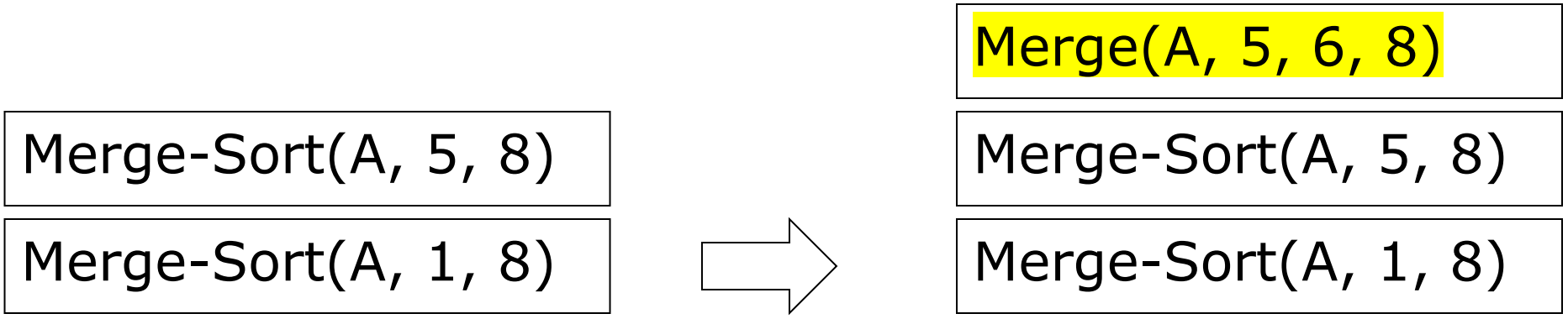
Activation Records and Recursive Calls



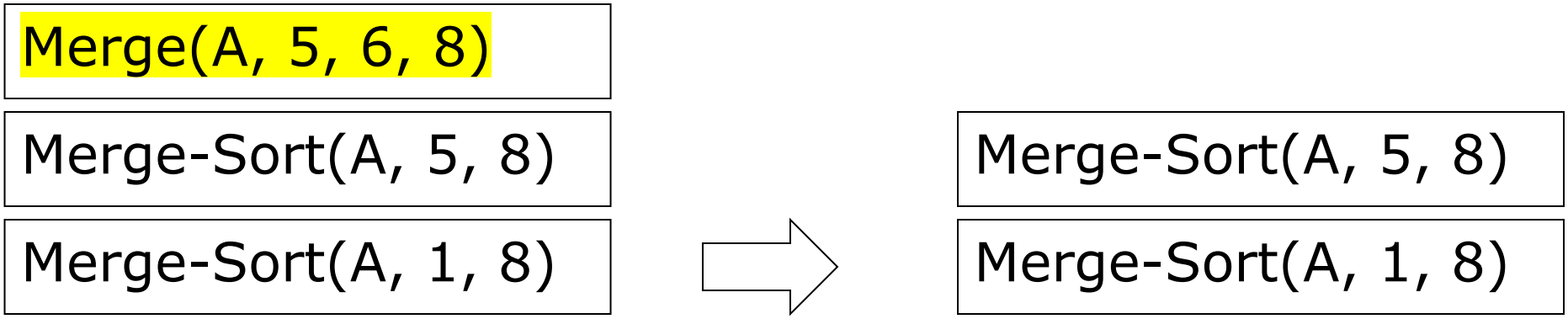
Activation Records and Recursive Calls



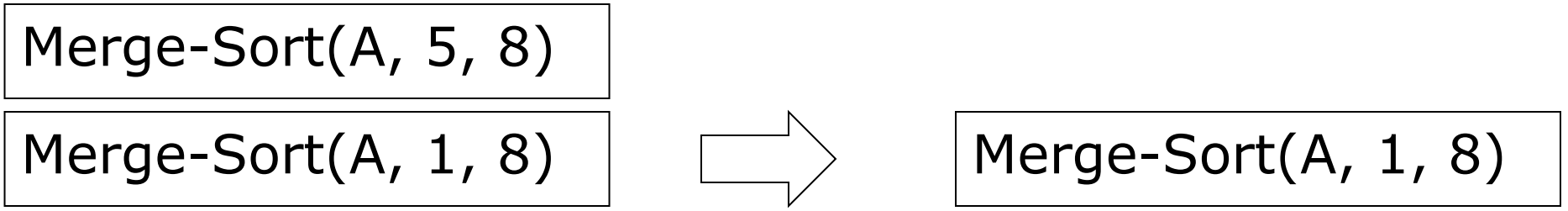
Activation Records and Recursive Calls



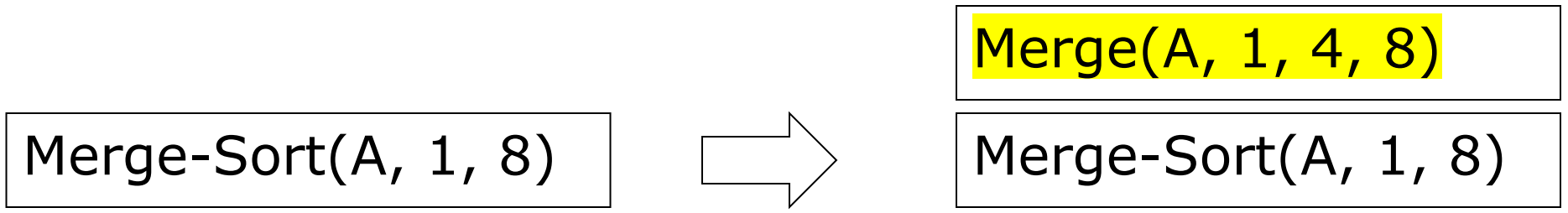
Activation Records and Recursive Calls



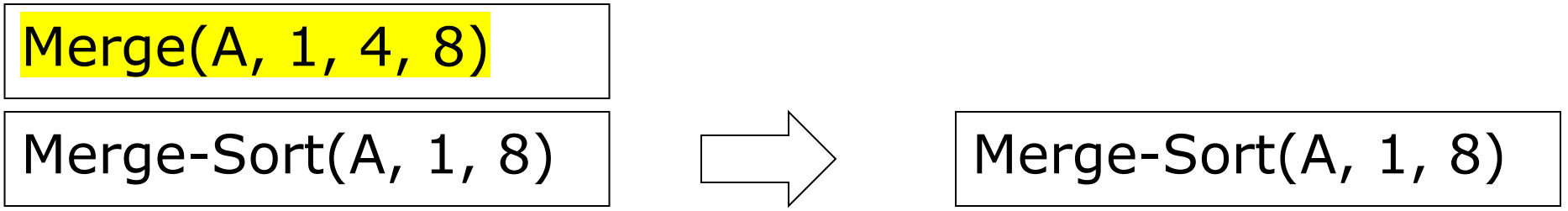
Activation Records and Recursive Calls



Activation Records and Recursive Calls

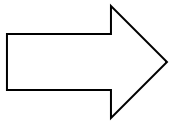


Activation Records and Recursive Calls



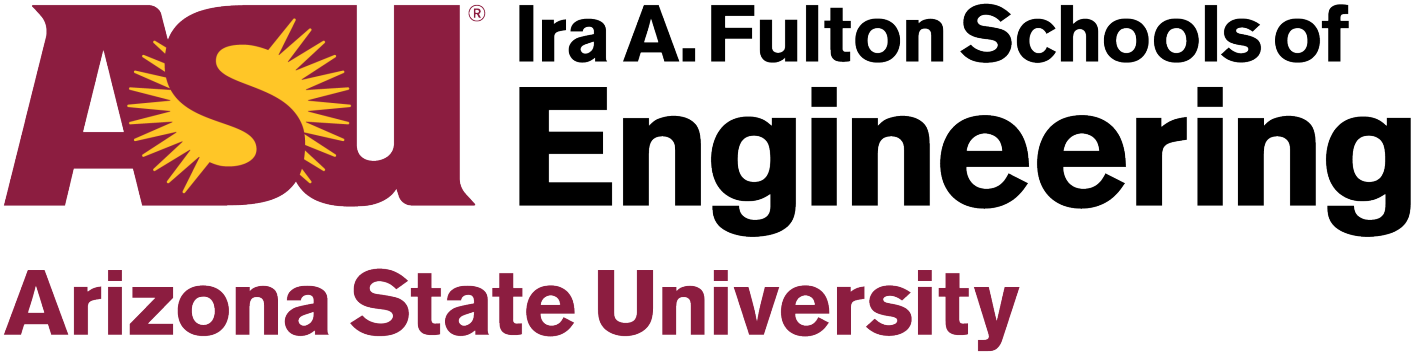
Activation Records and Recursive Calls

Merge-Sort(A, 1, 8)



Running Time of Mergesort

- Time needed for size n instance is T(n)
- The base case (n = 1): $\Theta(1)$
- Time needed to divide the instance: $D(n) = \Theta(1)$
- Time needed for 2 sub-instances is $2T(n/2)$
- $T(n) = 2T(n/2) + n$
- $T(n) = \Theta(n \log(n))$


Ira A. Fulton Schools of
Engineering
Arizona State University