Side Talk: Memory Management in Merge Sort

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

In the class, we discussed a recursive algorithm named merge sort. In this talk, we will discuss how the operating system allocates memory as the algorithm makes recursive calls.

Recall:

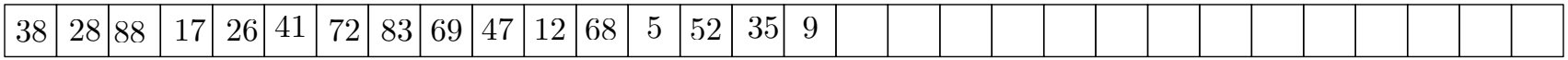
Merge Sort

Inductive Case. The algorithm runs in three steps:

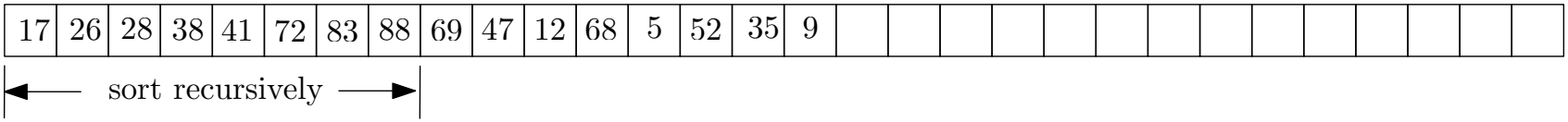
1. Recursively sort the first half of the array S.
2. Recursively sort the second half of the array.
3. Merge the two halves of the array into the final sorted sequence.

Example

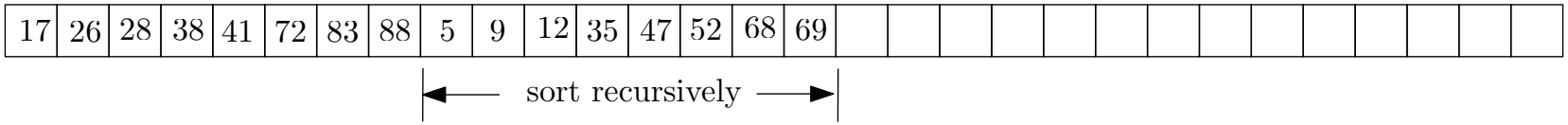
Input:



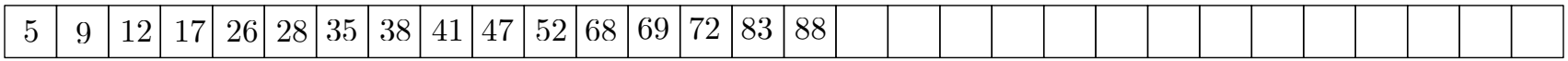
First step, sort the first half of the array by recursion.



Second step, sort the second half of the array by recursion:



Third step, merge the two halves.

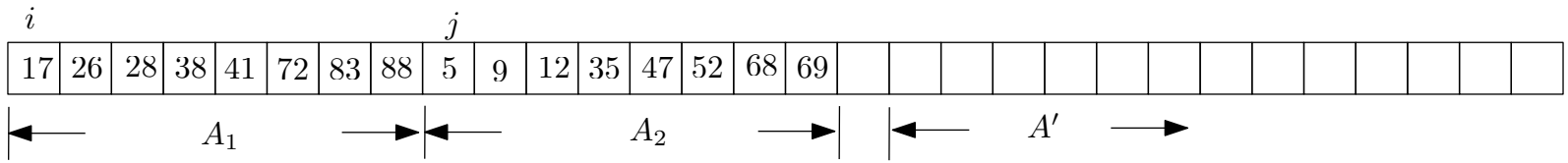


We can implement the merge step (i.e., Step 3 of Slide 3) as follows:

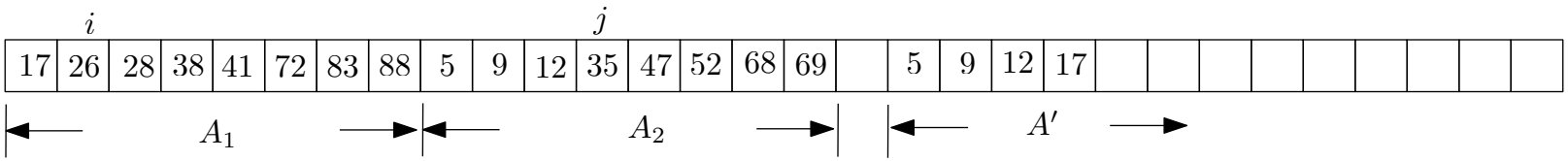
- Let A be the input array
- Let A_1 be the first half (already sorted) of A
- Let A_2 be the second half (already sorted) of A
- Create another array A' of length n
- Use A' to perform the merging of A_1 and A_2
- Copy A' to A
- Delete A' (i.e., freeing up memory)

Example

Create array A':

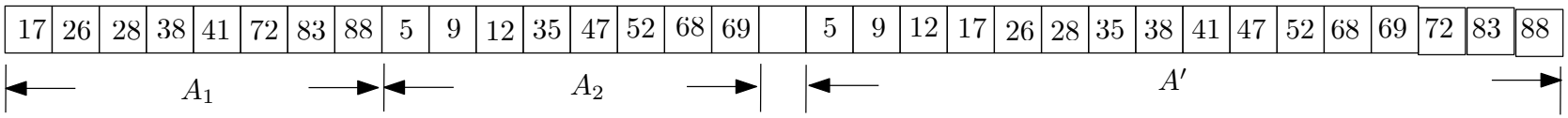


Appending 5, 9, 12, 17 to A, and so on:

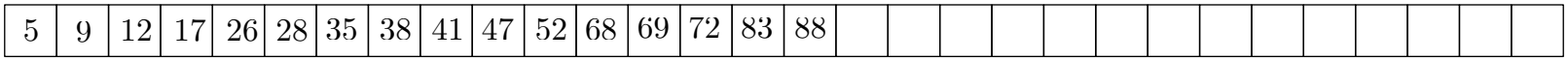


Example

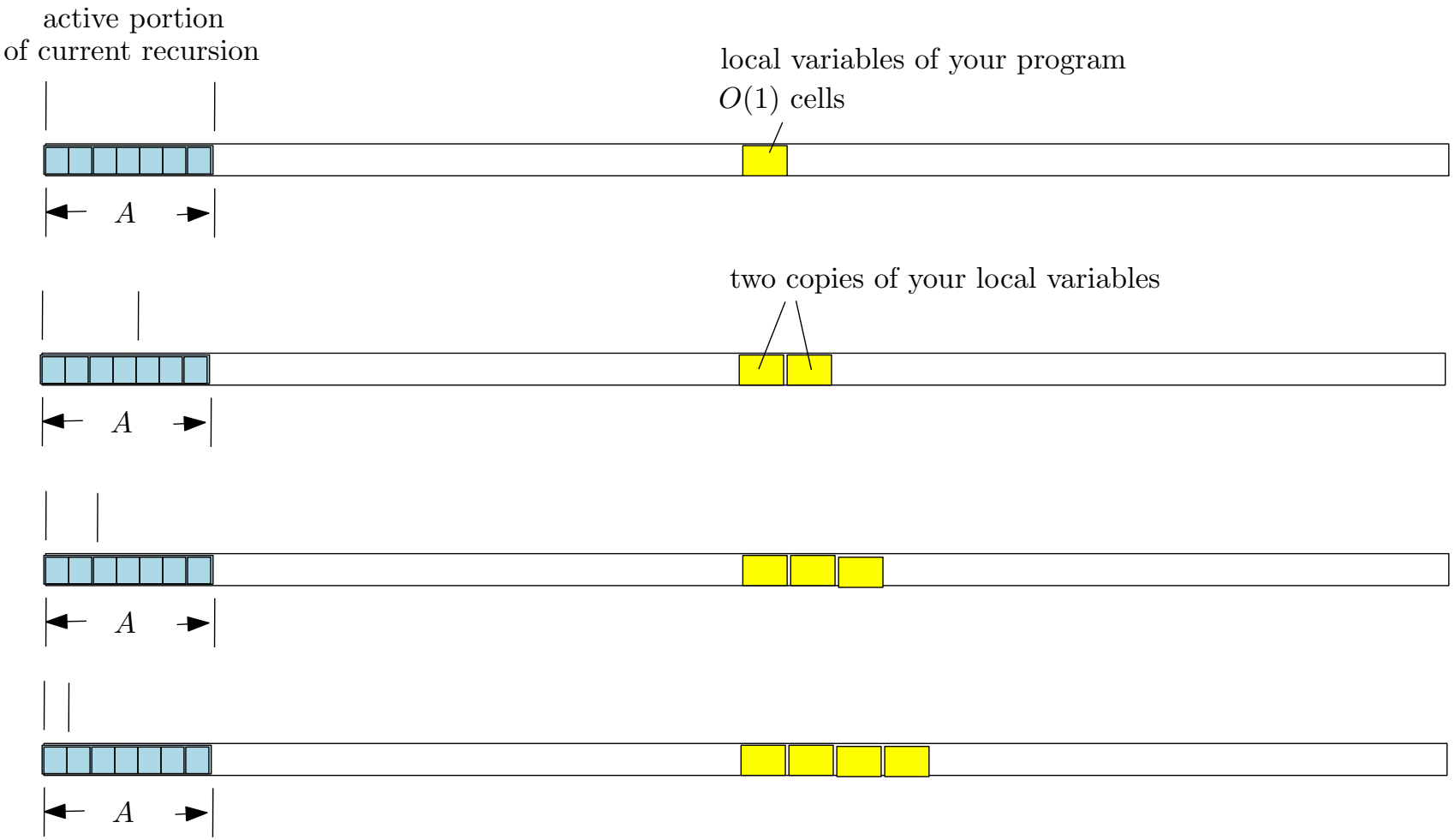
At the end of merging:

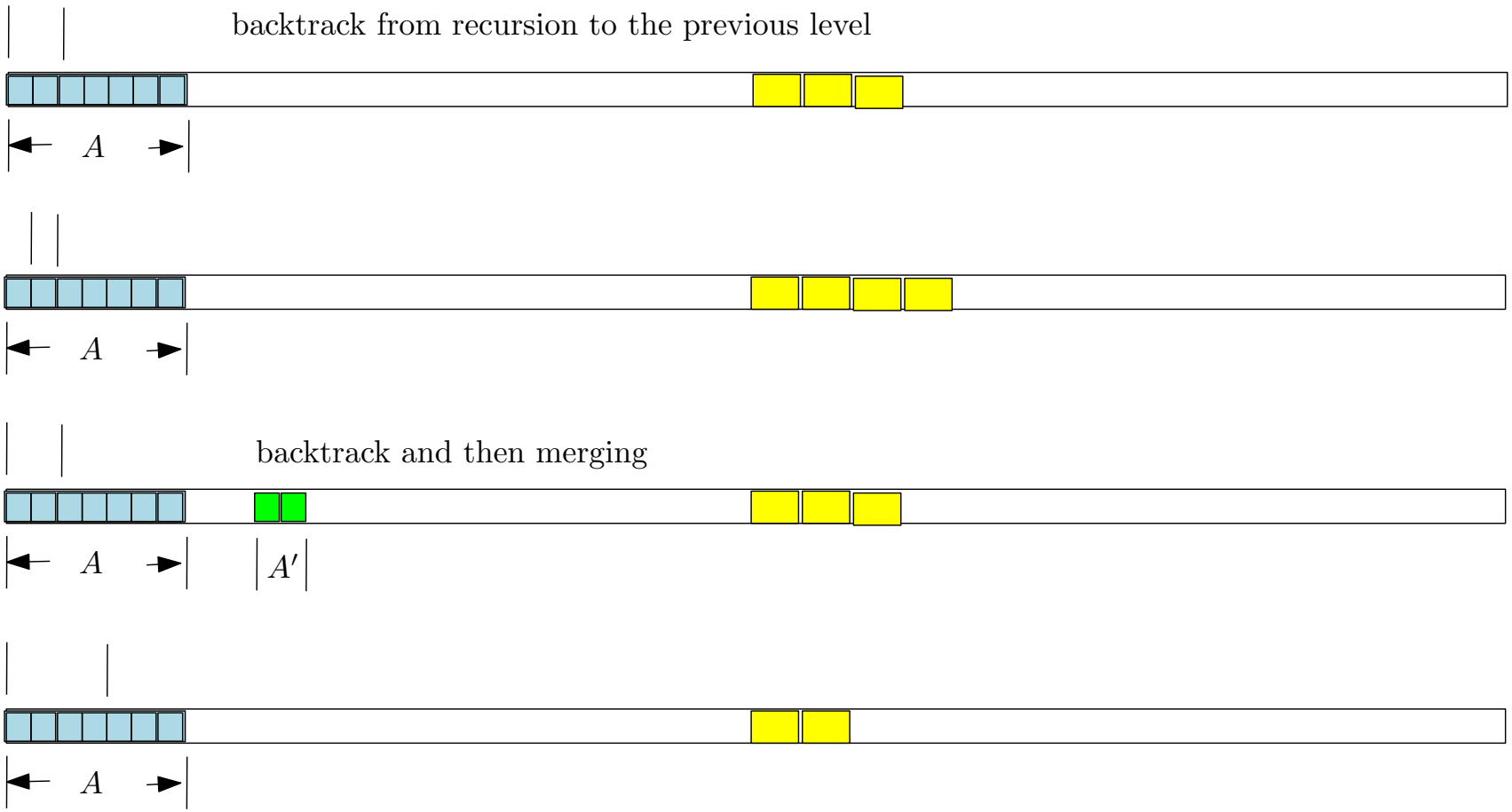


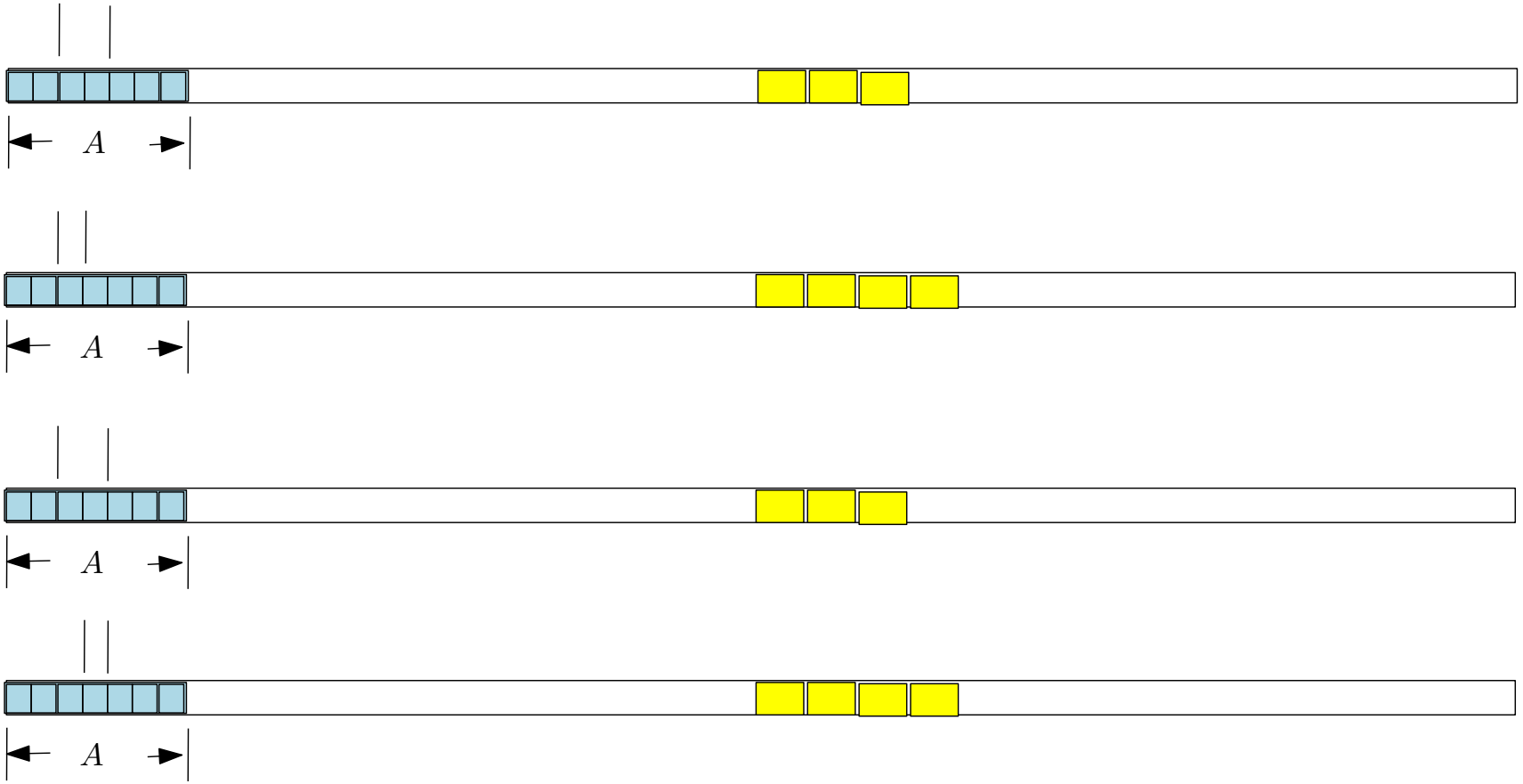
Copy A' to A, and destroy A':

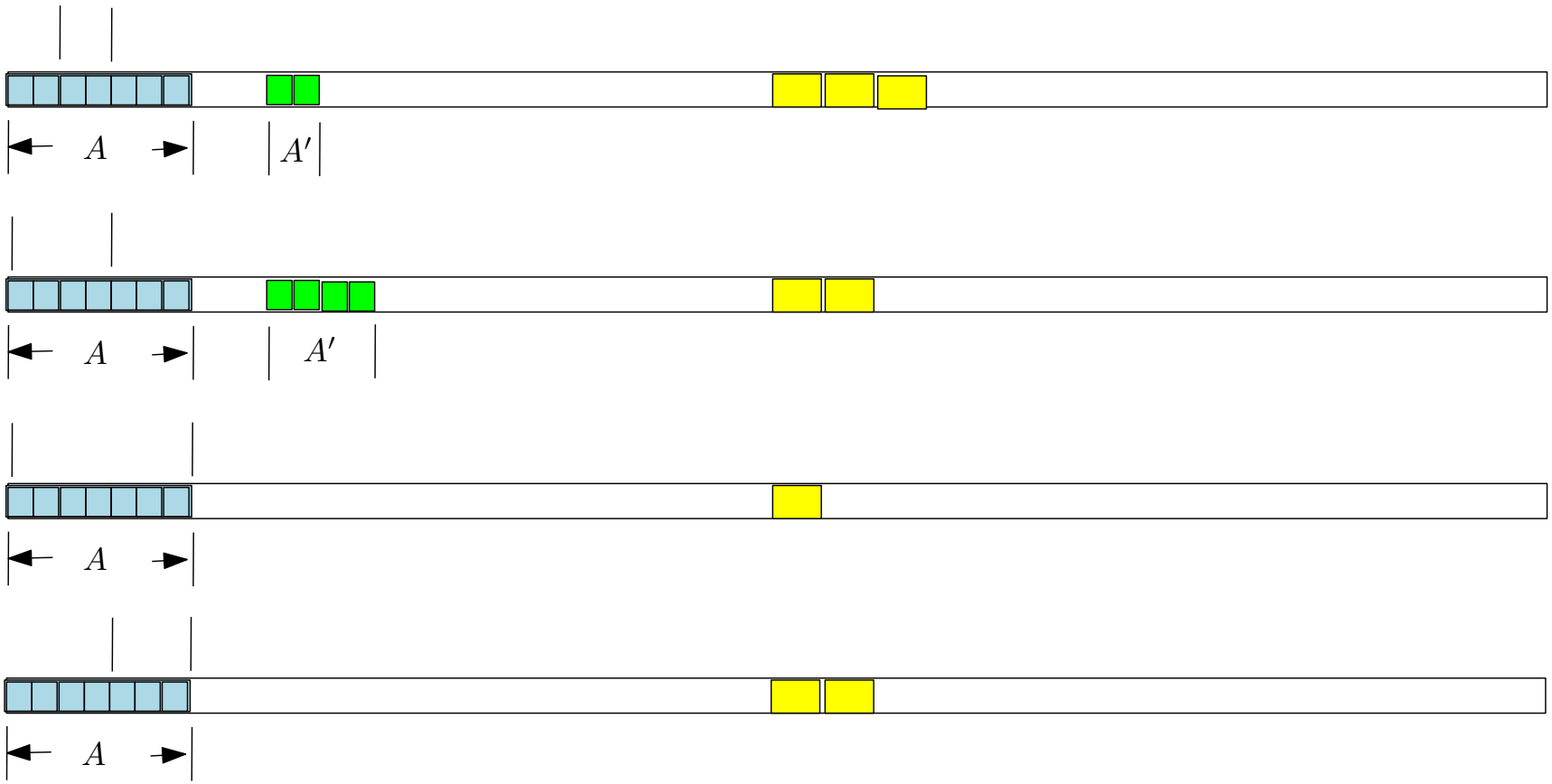


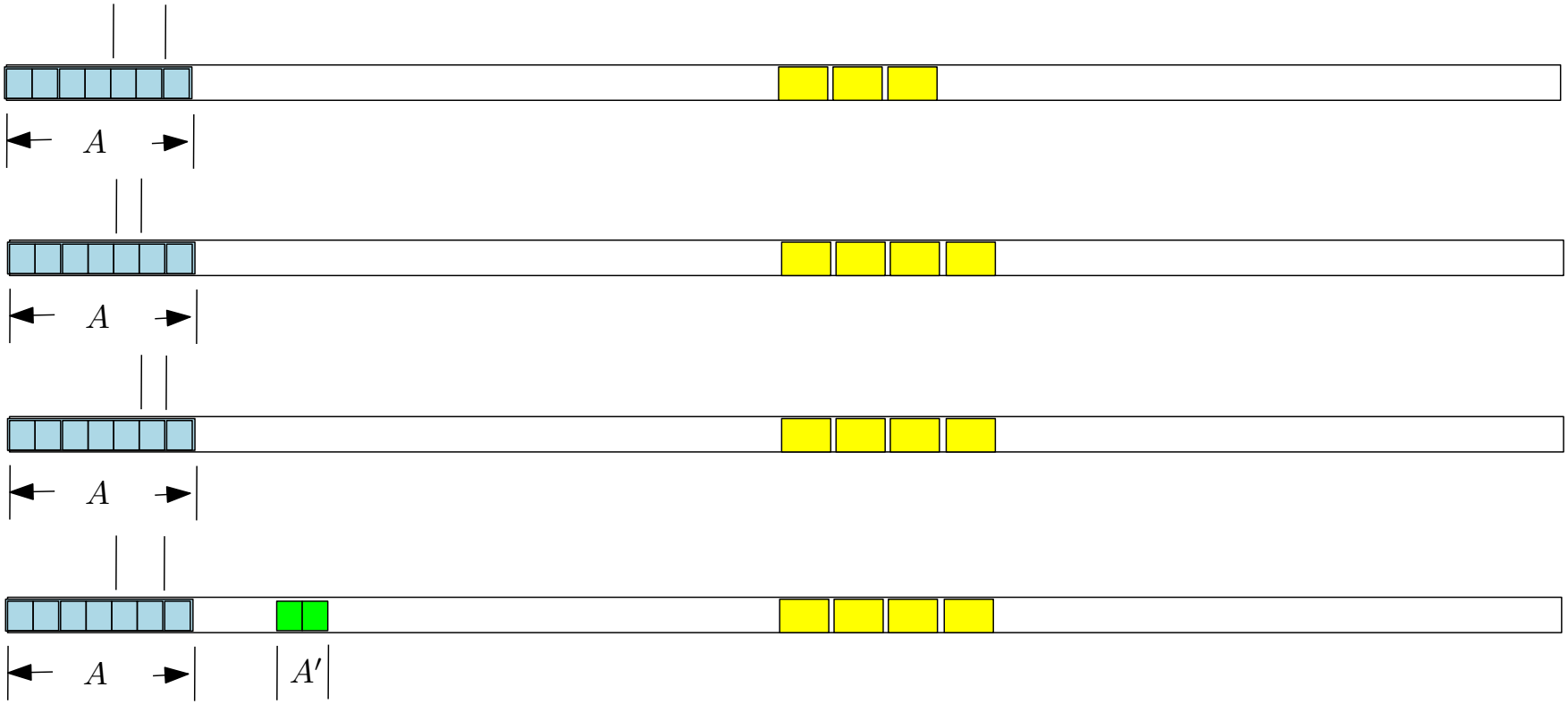
Next, we will demonstrate the entire history of memory allocation for using the algorithm to sort 7 elements.

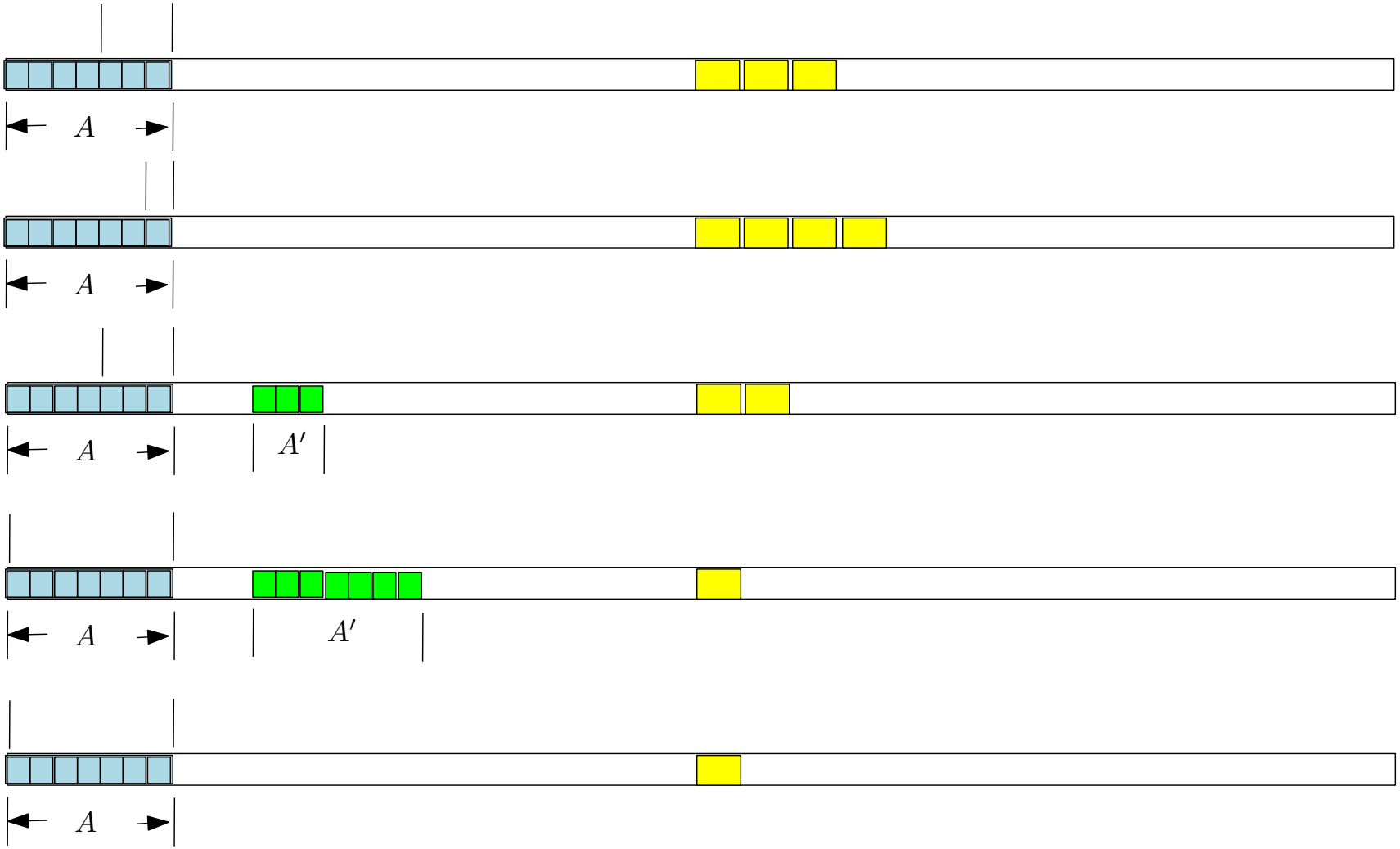

active portion
of current recursion
local variables of your program
$O(1)$ cells
A
two copies of your local variables
A
A
A


backtrack from recursion to the previous level
A
A
backtrack and then merging
A
A′
A


A
A
A
A


A
A′
A
A′
A
A


A
A
A
A
A′


A
A
A
A′
A
A′
A