```
Merge(A: array of integer, l, m, h: integer)
 1  (∗ A is an array A[1, ..., n] and 1 ≤ l ≤ m < h ≤ n. ∗)
 2  (∗ Assume A[l, ..., m] is sorted and A[m + 1, ..., h] is sorted. ∗)
 3  n1 ← m − l + 1
 4  n2 ← h − m
 5  create arrays L[1, ..., n1 + 1] and R[1, ..., n2 + 1]
 6  L[1, ..., n1] ← A[l, ..., m]
 7  R[1, ..., n2] ← A[m + 1, ..., h]
 8  L[n1 + 1] ← ∞
 9  R[n2 + 1] ← ∞
10  i ← 1
11  j ← 1
12  for k ← l to h
13      if L[i] ≤ R[j]
14          A[k] ← L[i]
15          i ← i + 1
16      else
17          A[k] ← R[j]
18          j ← j + 1
```

Problem 1. *Prove that after* Merge *terminates the array* $A[l, \ldots, h]$ *is sorted.*

Solution:
We prove the following is a loop invariant with respect to the **for** cycle (lines 12–18) in Merge. The invariant consists of two separate claims.

Invariant 1. *Every time the execution of* Merge *reaches line 12,*

> **claim i:** $A[l, \ldots, k-1]$ *contains* $k - l$ *smallest elements of* L *and* R *in sorted order.*
>
> **claim ii:** $L[i]$ *and* $R[j]$ *are smallest elements in* L *and* R*, respectively, that have not been copied into* A*.*

Proof: By induction on k, the loop control variable.

Basis: $k = l$. In that case the range $[l, \ldots, k-1]$ is $[l, \ldots, l-1]$. But that is an empty range, therefore **claim i** is vacuously true. **claim ii** holds because on the one hand $i = 1$ and $j = 1$, and on the other hand by the premises of Merge, $L[1]$ is a smallest element in L and $R[1]$ is a smallest element in R. Clearly, $L[1]$ and $R[1]$ have not been copied into A.

Inductive hypothesis: Assume **claim i** and **claim ii** hold when the execution of Merge is at line 12 and the body of the **for** loop is to be executed at least once more, that is,

$$k \leq h \tag{1}$$

Inductive step: Next line 13 is executed. We prove it never compares ∞ with ∞. Assume the opposite. By the inductive hypothesis, precisely $k - l$ elements from L and R are already copied into A. Since $L[i] = \infty$ and $R[j] = \infty$, Merge has already copied at least $n_1 + n_2$

elements into A (namely, the elements that are not ∞). But then it has to be the case that $k - l \geq n_1 + n_2$. Clearly, $n_1 + n_2 = h - m + m - l + 1 = h - l + 1$. So we derived that

$$k - l \geq h - l + 1 \;\Leftrightarrow\; k \geq h + 1 \tag{2}$$

which contradicts (1).

So, the **if** at line 13 compares values, at least one of which is not ∞, therefore the comparison makes sense. First assume the comparison yields "YES". Note that in this case

$$i < n_1 + 1 \tag{3}$$

because $L[i] \neq \infty$. By the inductive hypothesis $L[i]$ is a smallest element in L and R that has not been copied into A. Line 14 is executed next. After the assignment at line 14, $A[l, \ldots, k]$ contains $k - l + 1 = (k + 1) - l$ smallest elements of L and R, therefore **claim i** holds for $k + 1$. At line 15 i gets incremented by one. Having in mind (3), clearly the new value of i is an valid index for L. By the assumptions of **Merge**, $L[i] \geq L[i - 1]$ (for the new value of i). It follows that $L[i]$ is a smallest element in L not copied into A. Therefore **claim ii** holds for $k + 1$.

In the alternative case, namely when the comparison in the **if** at line 13 yields "NO", the argument is completely analogous.

Termination: When the **for** loop terminates, it is the case that $k = h+1$, so the subarray $A[l, \ldots, k-1]$ from **claim ii** of the invariant is in fact $A[l, \ldots, h]$. According to the invariant, that subarray contains the $k-l = h+1-l$ smallest elements of L and R in sorted order. But those are precisely the elements of L and R that are not ∞. It follows $A[l, \ldots, h]$ contains precisely the elements it contained at the beginning of MERGE but in sorted order. $\square$

```
Partition(A: array of integer, l,h: integer)
1  (* A is an unsorted array A[1,...,n] and 1 ≤ l < h ≤ n. *)
2  pivot ← A[h]
3  pp ← l
4  for i ← l to h − 1
5      if A[i] < pivot
6          swap(A[i], A[pp])
7          pp ← pp + 1
8  swap(A[h], pp)
9  return pp
```

Problem 2. *Prove the value* pp *returned by* Partition *is such that* $l \leq pp \leq h$ *and* $\forall x \in A[l, \ldots, pp-1], \forall y \in A[pp+1, \ldots, h] : x < A[pp] \leq y$.

Solution:
We prove Invariant 2 is a loop invariant of the **for** cycle (lines 4–7) in Partition. Let Q^i for $l \leq i \leq h$ be the set of elements from the subarray $A[l, \ldots, i-1]$ that are strictly smaller than pivot every time the execution is at line 4.

Invariant 2. *Every time the execution of* Partition *is at line 4, the elements of* Q^i *form a contiguous subarray* $A[l, \ldots, l + |Q^i| - 1]$ *and* $pp = l + |Q^i|$.

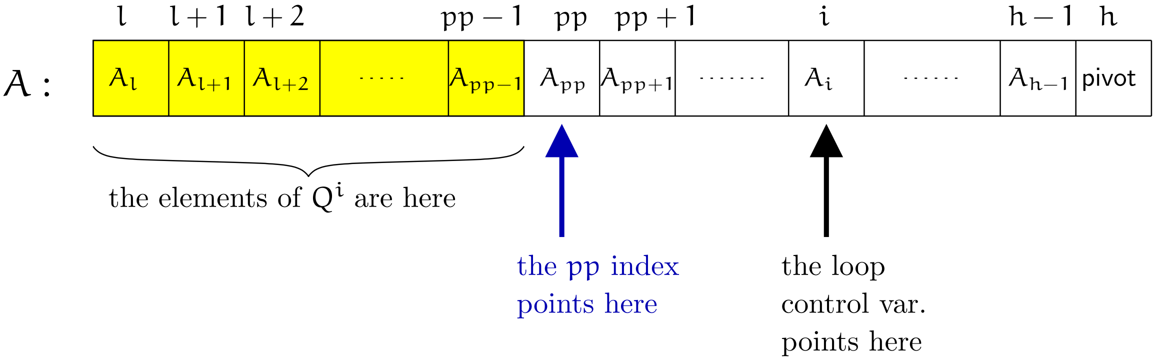
Proof: By induction on i, the loop control variable.

Basis: $i = l$. The subarray $A[l, \ldots, i-1]$ is $A[l, \ldots, l-1] = A[\emptyset]$, therefore $Q^l = \emptyset$, therefore the first part of the invariant is vacuously true[†]. The second part of the invariant holds since $pp = l = l + |\emptyset|$.

Inductive hypothesis: Assume the invariant holds for some i such that $i \leq h-1$, that is, the body of the **for** loop is to be executed at least once more.

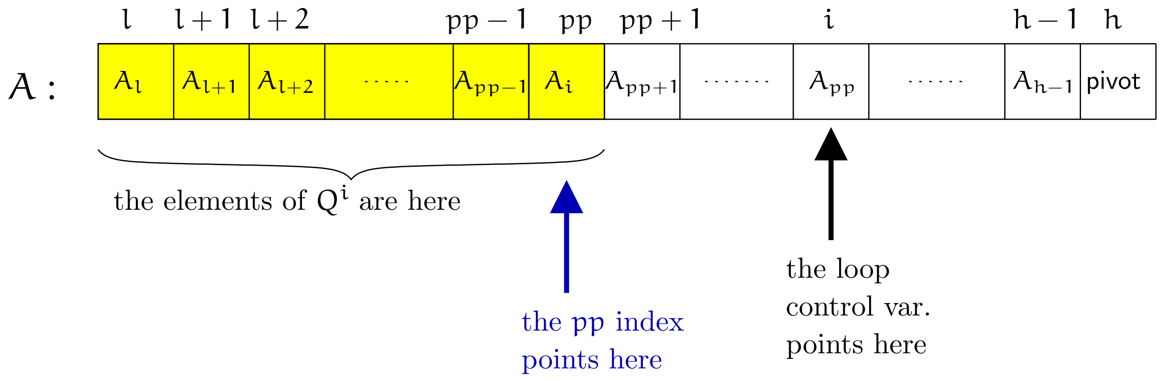
Inductive step: Consider the execution of the **for** loop afterwards. We consider several separate cases.

Case 1: $pp < i$ and $A[i] <$ pivot. By the inductive hypothesis, pp is the index of the element with the smallest index that is not smaller than pivot, and all the elements smaller than pivot—namely, the elements of Q^i—are left of $A[pp]$. The following figure illustrates the array when the execution is at line 4. On it, the elements of Q^i are shown in yellow. For brevity we write A_l rather than $A[l]$, *etc.*

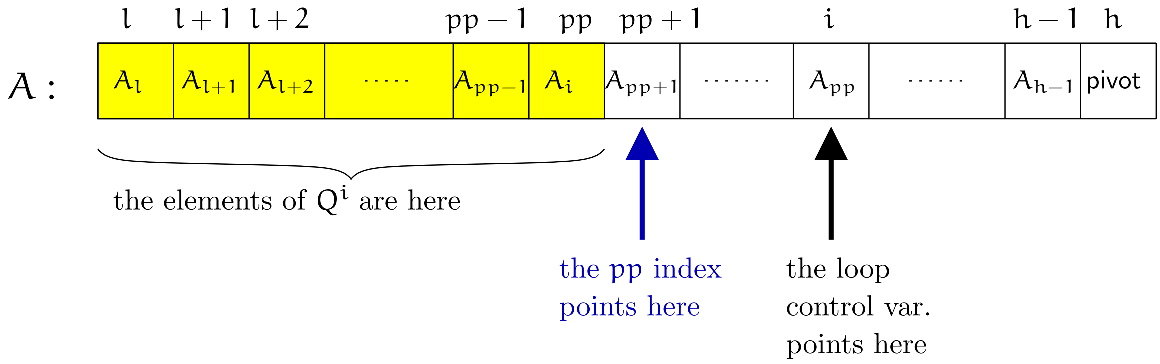


[†]The subarray $A^l[\emptyset]$ contains zero elements, hence the term "vacuously".

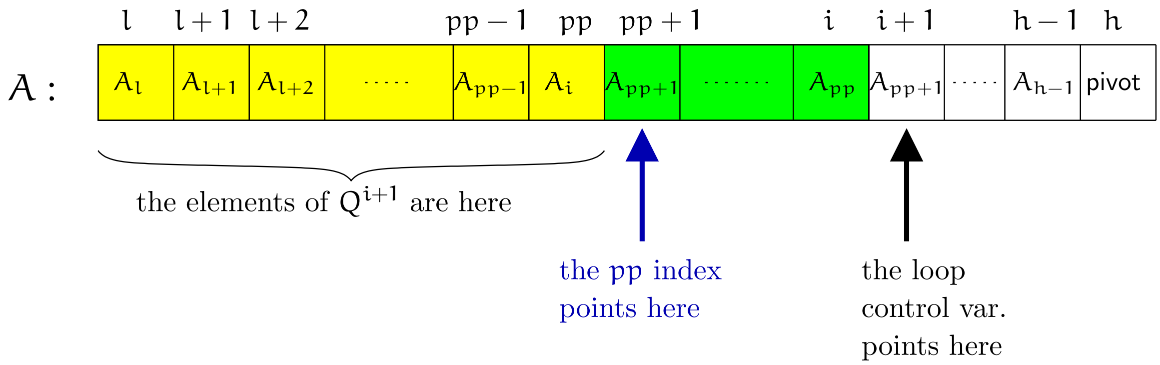
As $A[i] < \mathsf{pivot}$, the condition at line 5 is fulfilled and so the execution proceeds to line 6 where $A[pp]$ and $A[i]$ get swapped. Note that Q^i "grows" with one element, namely $A^i[i]$:



At line 7, pp is incremented to $pp + 1$:



Next the execution is at line 4 again, with i incremented to $i + 1$. Call the subarray $A[pp + 1, \ldots, i]$, *the green subarray*, as shown here:



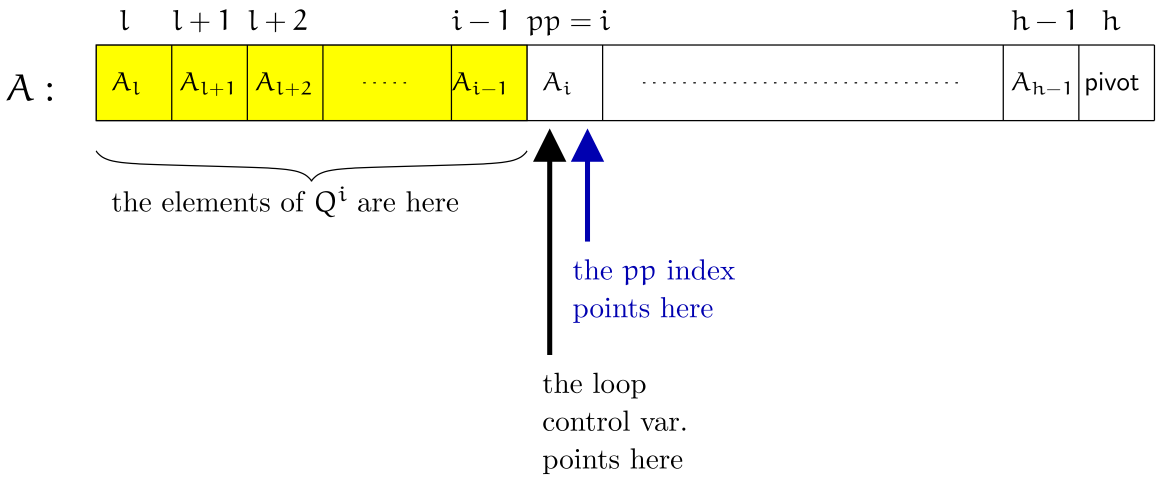
Note that all elements of the green subarray are greater than or equal to pivot for the following reasons:

- The elements in $A[pp + 1, \ldots, i - 1]$ are greater than or equal to pivot because of the inductive hypothesis.
- The current $A[i]$ is the former $A[pp]$. By the inductive hypothesis, it is greater than or equal to pivot.

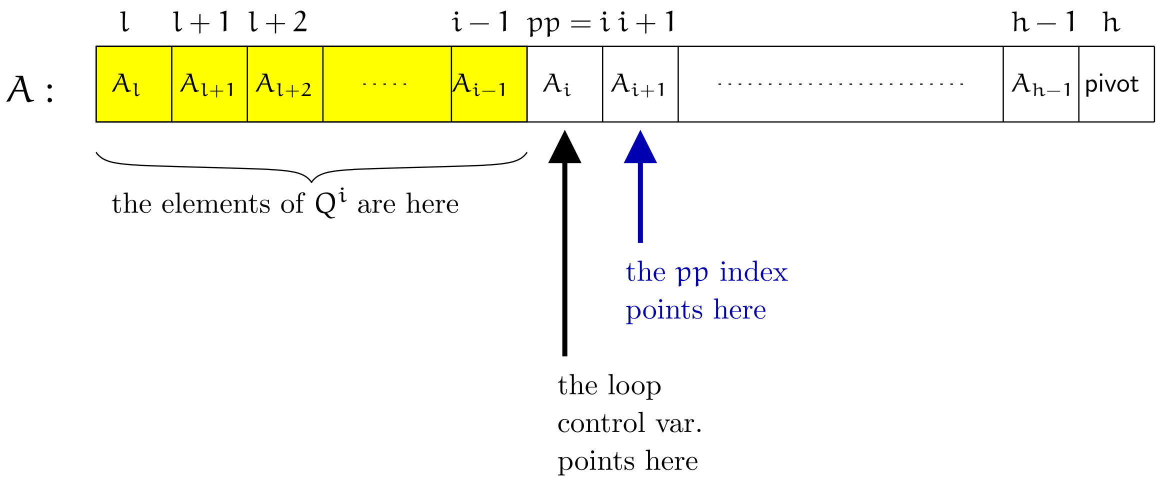
Therefore, the invariant holds when the execution is at line 4 and the loop control variable is $i+1$.

Case 2: $pp < i$ and $A[i] < \mathsf{pivot}$. In this case, the condition at line 5 is not fulfilled and so the execution proceeds directly to line 4 with the loop control variable being $i+1$. The inductive step, therefore, follows immediately from the inductive hypothesis since $Q^{i+1} = Q^i$, no element in the array is moved, and pp remains the same.

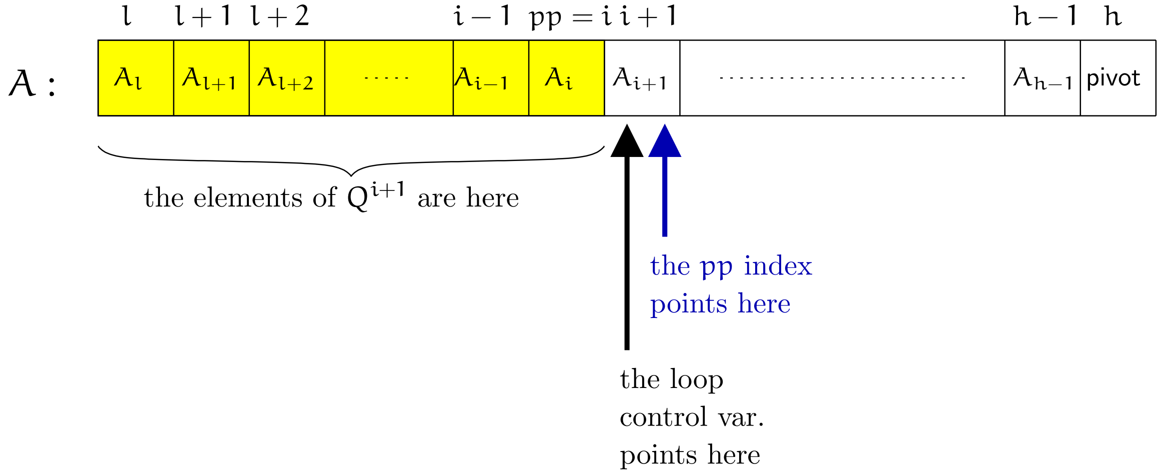
Case 3: $pp = i$. Observe that that is possible but only in case all the elements in $A[l, \ldots, i-1]$ are smaller than pivot. Then $|Q^i| = i - 1 - l + 1 = i - l$ and so the elements of Q^i are in the subarray $A[l, \ldots, l + |Q^i| - 1] = A[l, \ldots, l + i - l - 1] = A[l, \ldots, i-1]$ and $pp = l + |Q^i| = l + i - l = i$. See the following figure:



Case 3.a: If $A[i] < \mathsf{pivot}$ then the condition at line 5 is fulfilled and so the execution proceeds to line 6 where $A[pp]$ and $A[i]$ get swapped, that is, $A[i]$ is swapped with itself and the array does not change. Then at line 7, pp is incremented:



Then the execution proceeds to line 4 where i is incremented to $i+1$. Note that $Q^{i+1} = Q^i \cup \{A_i\}$:



Clearly, the invariant holds.

Case 3.b: If $A[i] \geq \mathsf{pivot}$, the condition at line 5 is not fulfilled and so the execution proceeds directly to line 4 with the loop control variable being $i+1$. The inductive step, therefore, follows immediately from the inductive hypothesis since $Q^{i+1} = Q^i$, no element in the array is moved, and pp remains the same.

Case 1, **Case 2**, **Case 3.a**, and **Case 3.b** are exhaustive and so the invariant is proved.

Termination: When the **for** loop terminates, it is the case that $i = h$. By definition, Q^h is the set of the elements in $A[l, \ldots, h-1]$ smaller than pivot. But since there is only one more element in $A[l, \ldots, h]$, namely pivot itself, Q^h consists precisely of the elements in $A[l, \ldots, h]$ smaller than pivot. By the loop invariant, the elements of Q^h form a contiguous subarray $A[l, \ldots, l + |Q^h| - 1]$ and pp is the index of the element immediately to the right of the rightmost of them. Clearly, all the elements with indices larger than pp are greater than or equal to pp. $\square$