Developing algorithms on arrays

We may specify an algorithm that deals with arrays by giving its precondition and postcondition as pictures.

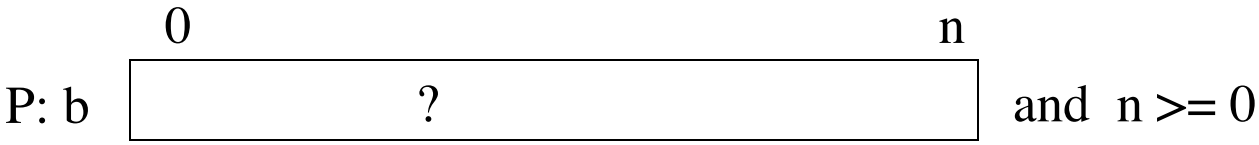
We then draw the invariant by drawing another picture that "generalizes" the precondition and postcondition, since the invariant is true at the beginning and at the end.

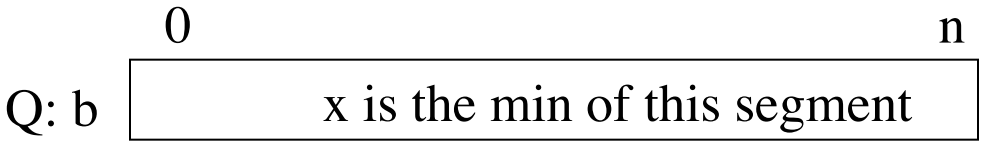
Four loopy questions —memorize them:

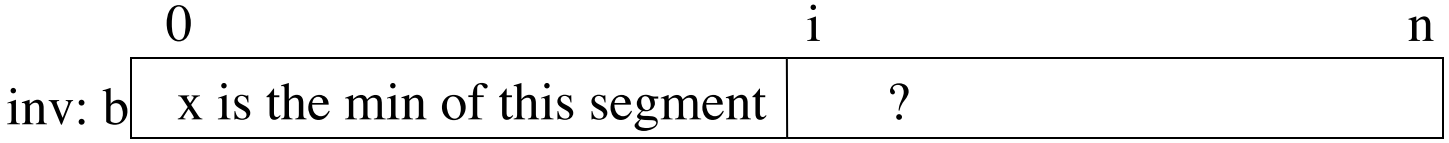
1. How does loop start (how to make the invariant true)?
2. When does it stop (when is the postcondition true)?
3. How does repetend make progress toward termination?
4. How does repetend keep the invariant true?

Invariant as picture: Combining pre- and post-condition

Finding the minimum of an array. Given array b satisfying precondition P, store a value in x to truthify postcondition Q:

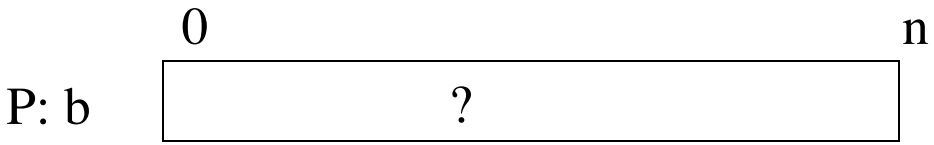




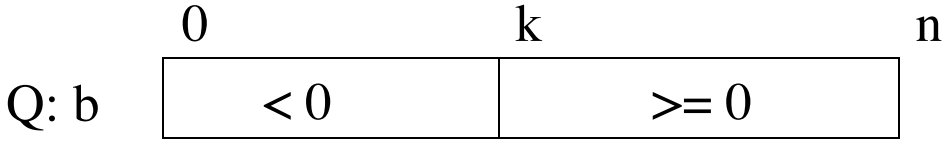


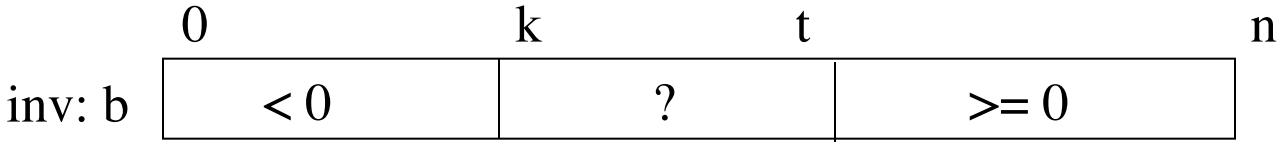
The invariant as picture: Combining pre- and post-condition

Put negative values before nonnegative ones. given precondition P:



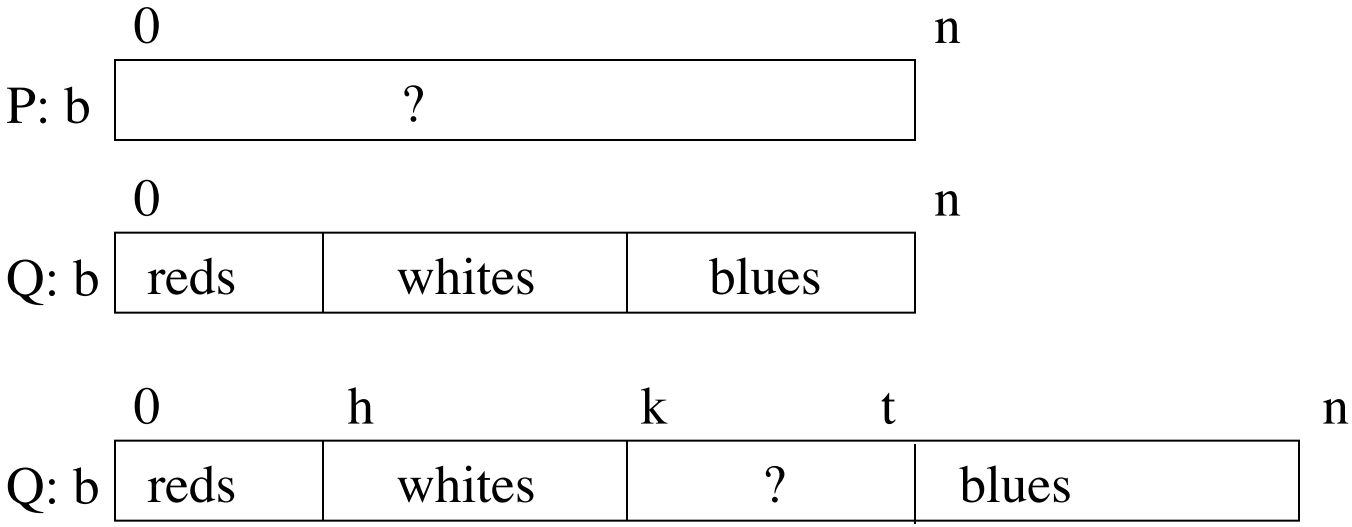
Swap the values of b[0..n-1] and store in k to truthify Q:



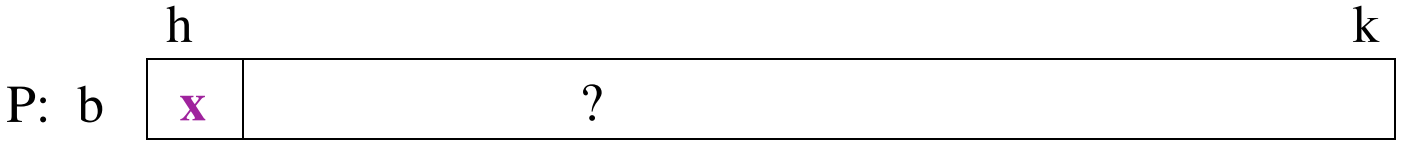


The invariant as picture: Combining pre- and post-condition

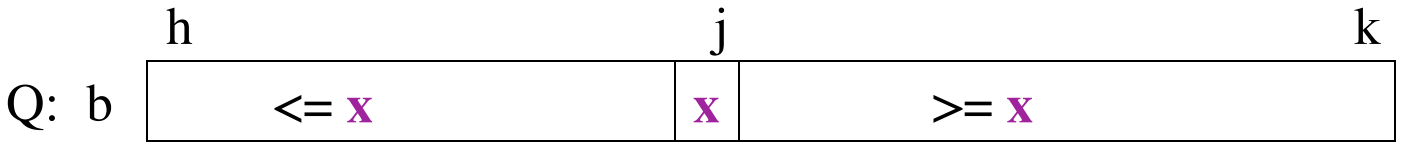
Dutch national flag. Swap values of 0..n-1 to put the reds first, then the whites, then the blues. That is, given precondition P, swap value of b[0.n] to truthify postcondition Q:

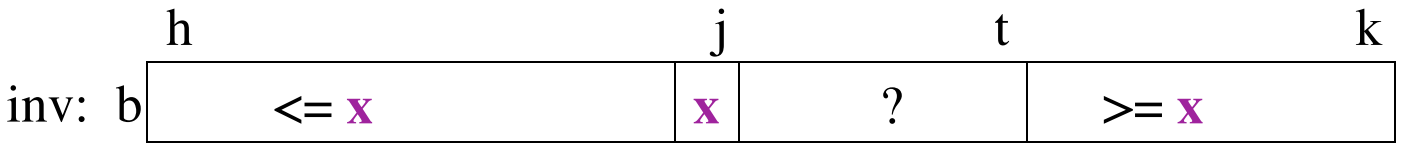


Partition algorithm: Given an array b[h..k] with some value x in b[h]:



Swap elements of b[h..k] and store in j to truthify P:





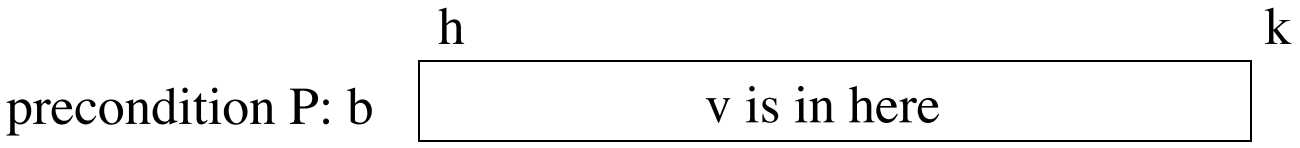
x is called the pivot value.

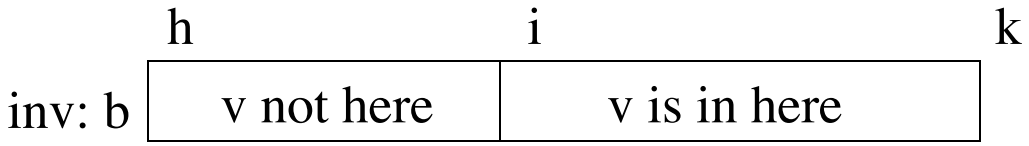
x is not a program variable; x just denotes the value initially in b[h].

Linear search (for value known to be in array)

Vague spec.: Find first occurrence of v in b[h..k-1], which is known to be there.
Better spec.: Store an integer in i to truthify postcondition Q:

Q: 1. v is not in b[h..i-1]
2. v = b[i]

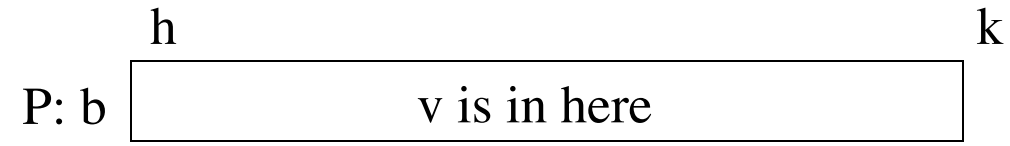


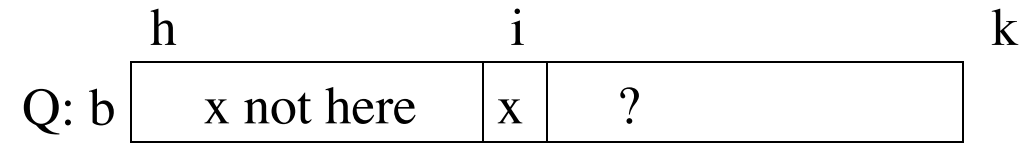


Linear search

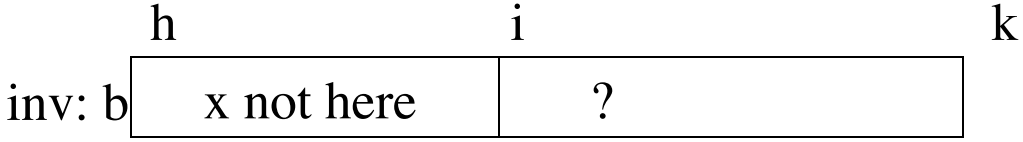
Vague spec.: Find first occurrence of v in b[h..k-1].
Better spec.: Store an integer in i to truthify postcondition Q:

Q: 1. v is not in b[h..i-1]
2. i = k OR v = b[k]





OR



Binary search: Vague spec: Look for v in **sorted** array segment b[h..k].

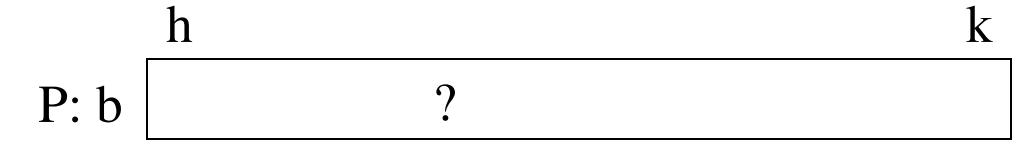
Better spec:

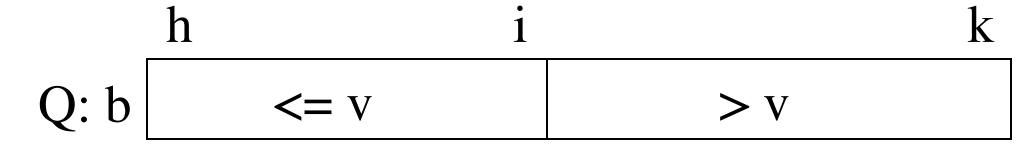
Precondition P: b[h..k] is sorted (in ascending order).

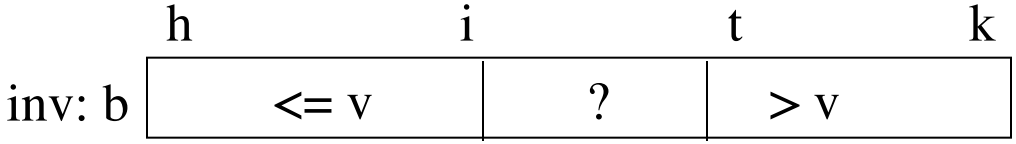
Store in i to truthify:

Postcondition Q: b[h..i] <= v and v < b[i+1..k]

Below, the array is in non-descending order:



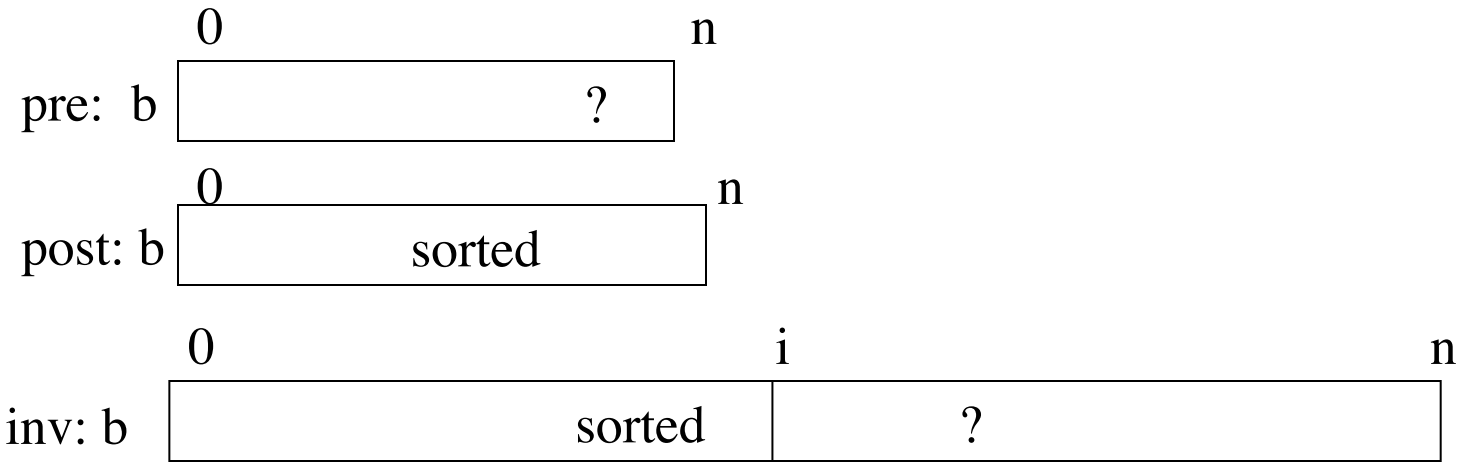




Called binary search because each iteration of the loop cuts the array segment still to be processed in half

Insertion sort

Swap the values in b[0..n-1] so that
the segment is sorted (in ascending order)



Selection sort

Swap the values in b[0..n-1] so that
the segment is sorted (in ascending order)