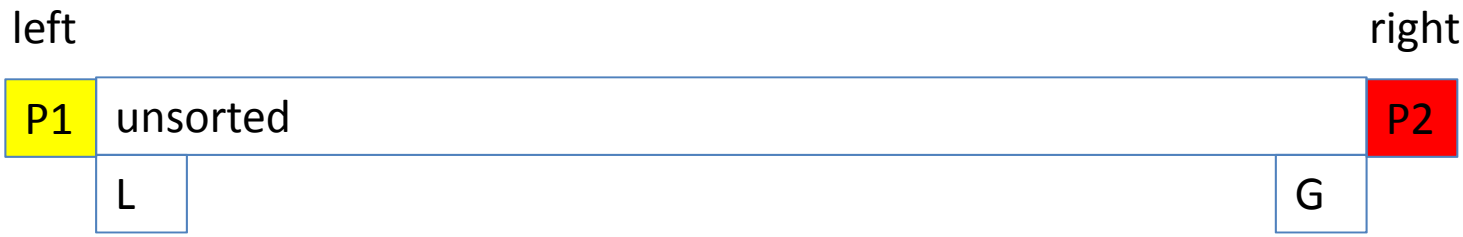
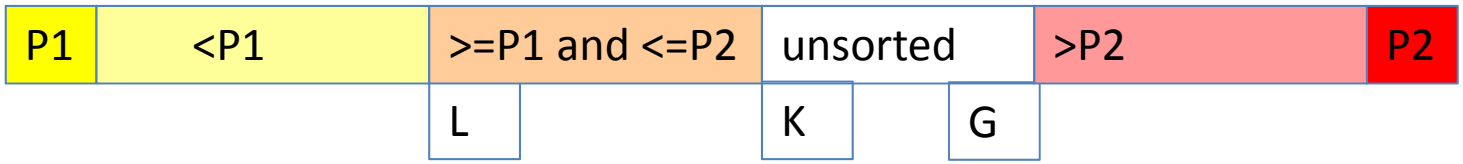


We start with everything unsorted
P1 and P2 are the first and last element
at indexes left and right, respectively.

P1 must be less than or equal to P2
if they are not, swap them.

L and G keep track of which elements are
Less than P1 and
Greater than P2

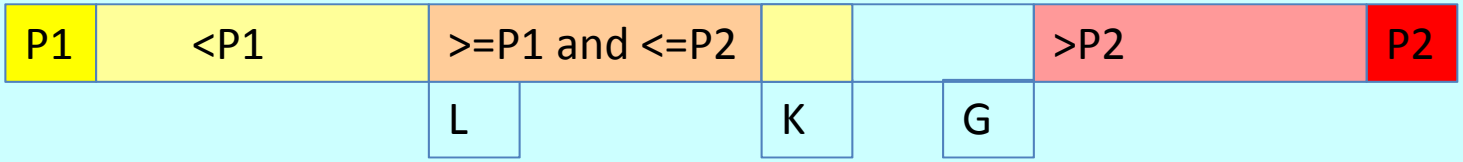


The algorithm is easier to understand if you look at it partway through.
K is the index of the element we are currently sorting

There are 3 possible cases.

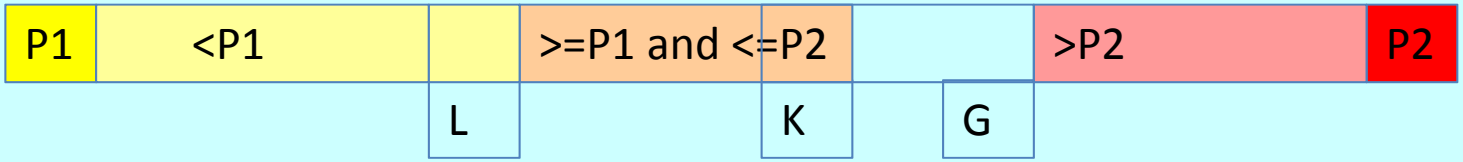
- array[K] < P1
- array[K] > P2
- P1 <= array[K] <= P2

case1

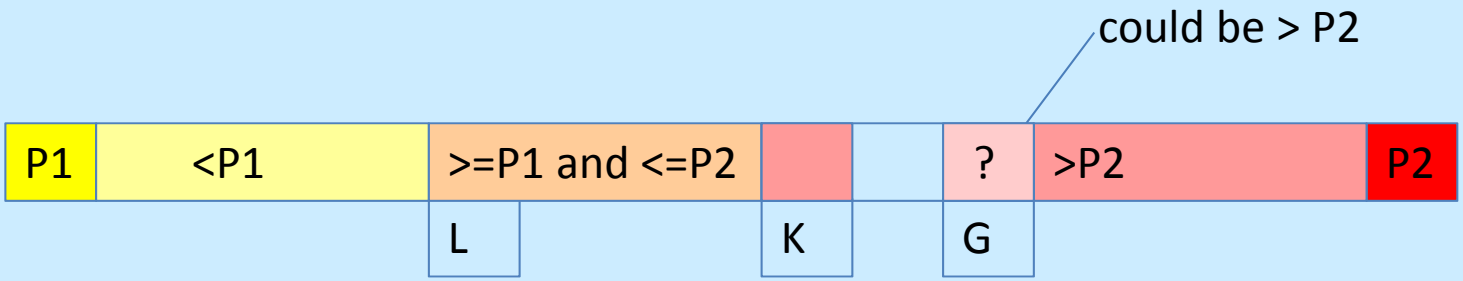


if array[K] < P1
just swap it into place

if array[K] < P1
swap array[K] with Array[L]
increment L

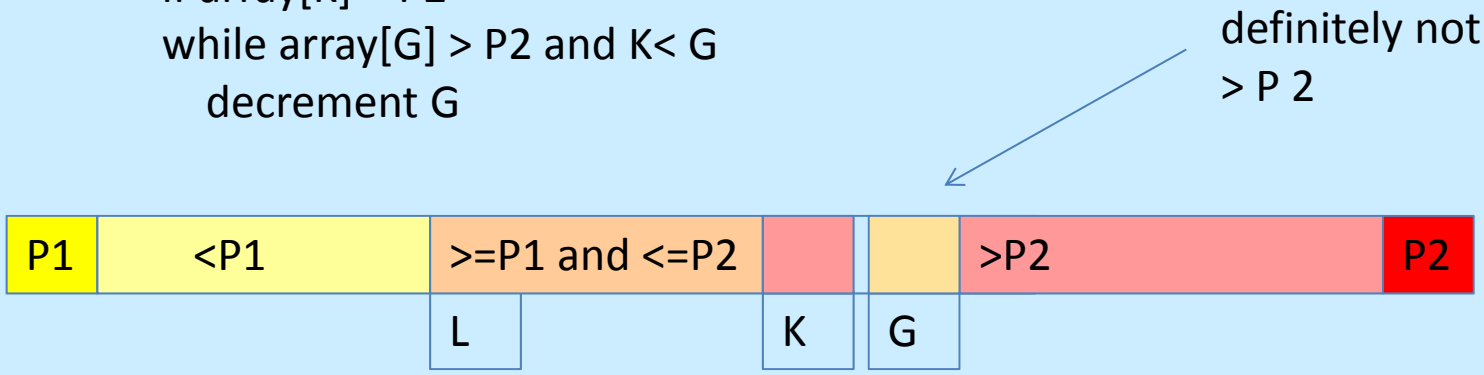


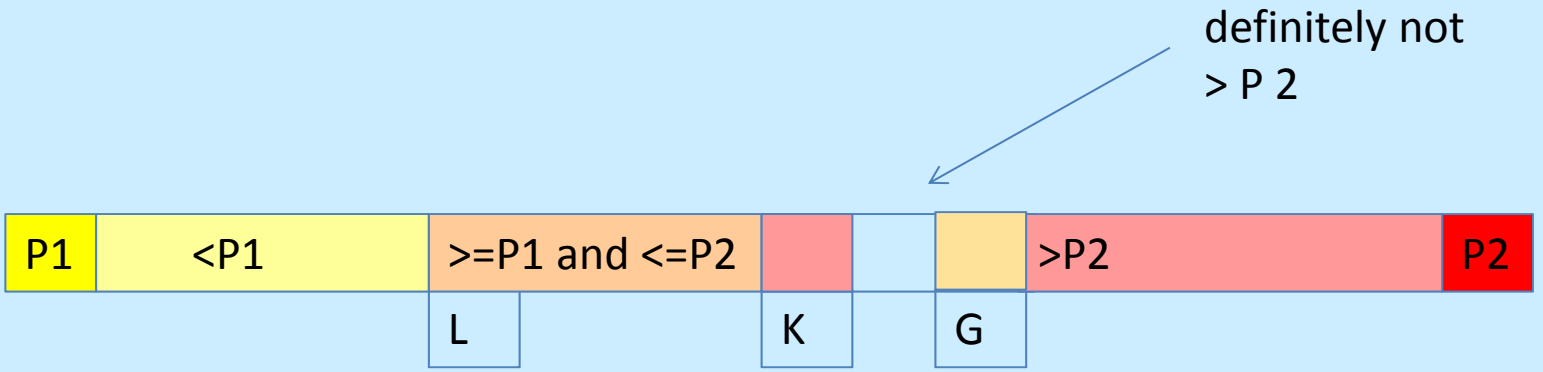
case2



otherwise if array[K] > P2
we can't just swap array[K] with array[L] because array[L] might be > P2 so we need to sort that out first

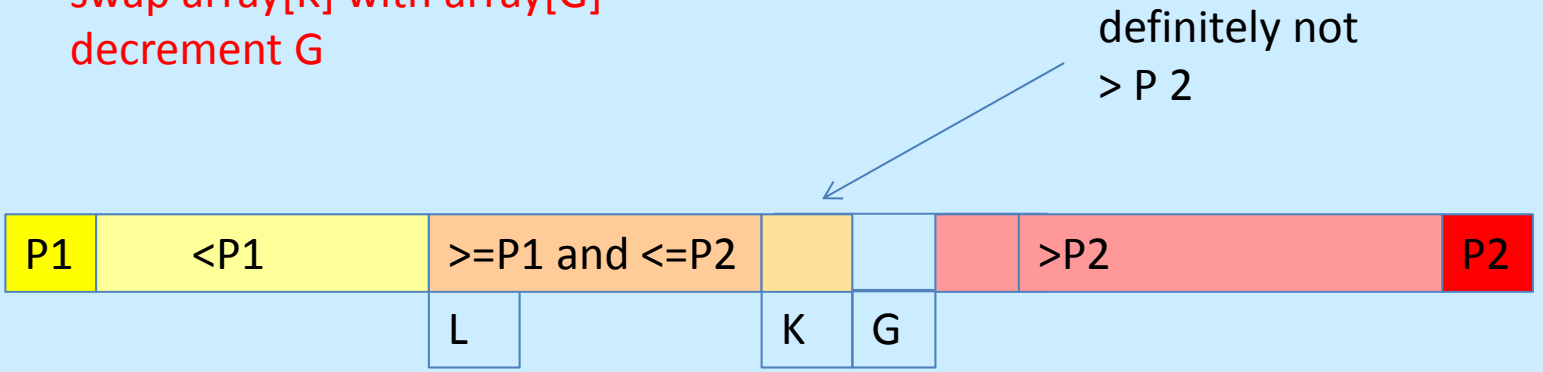
if array[K] > P2
while array[G] > P2 and K< G
decrement G

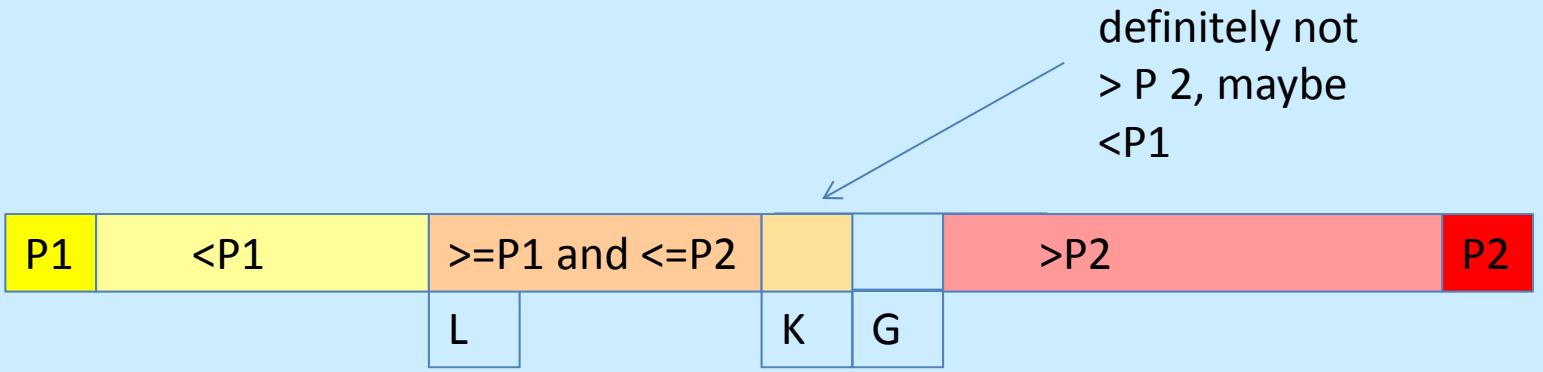




if array[K] > P2
we can't just swap array[K] with array[L] because array[L]
might be > P2 so we need to sort that out first
then we can swap array[K] with array[G]

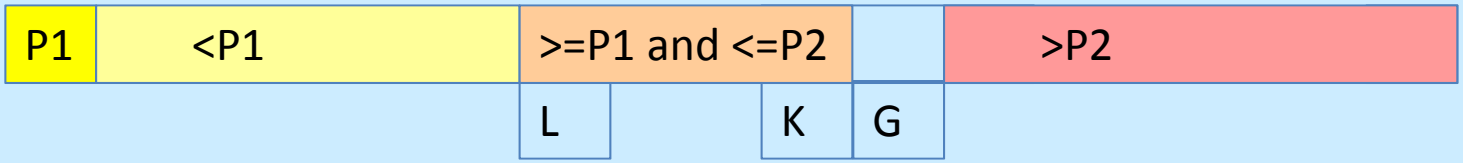
while array[G] > P2 and K< G
decrement G
swap array[K] with array[G]
decrement G



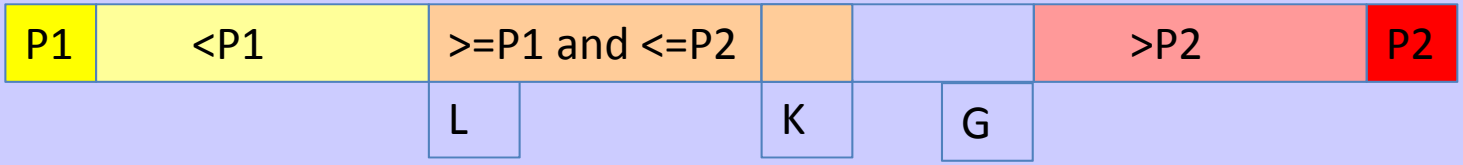


of course we now know A[K] is not >P2
but it might be < P1
in which case we need to swap it with array[L]

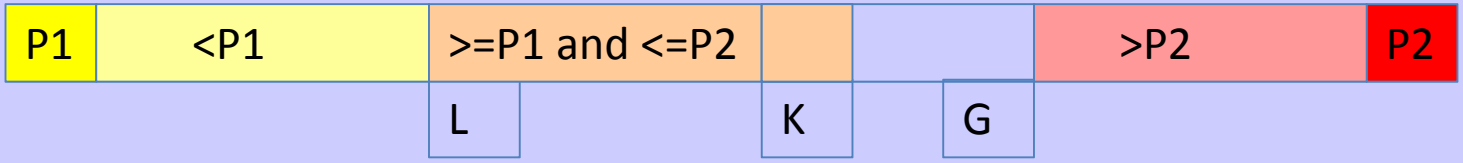
while array[G] > P2 and K< G
    decrement G
swap array[K] with array[G]
decrement G
if array[K] < P1
    swap array[K] and array [L]
    increment L

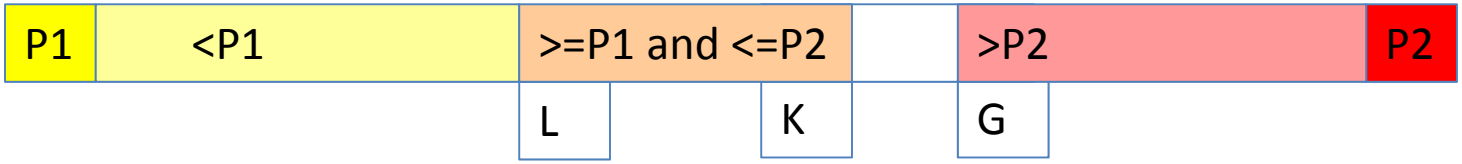


case3



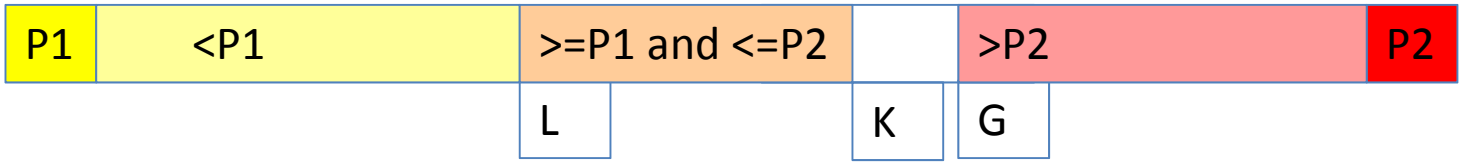
otherwise, array[K] is >=P1 and <=P2
and there is nothing to swap.

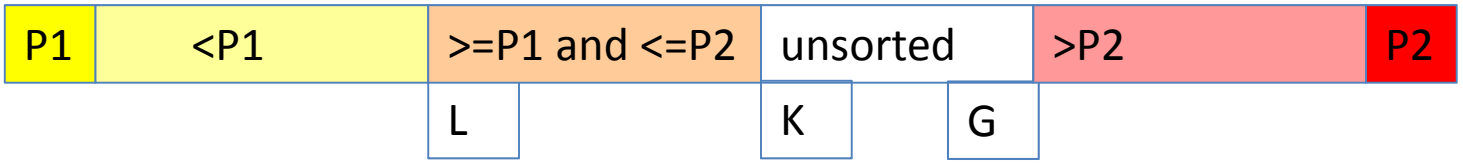




in all cases, array[K] has now been sorted, so

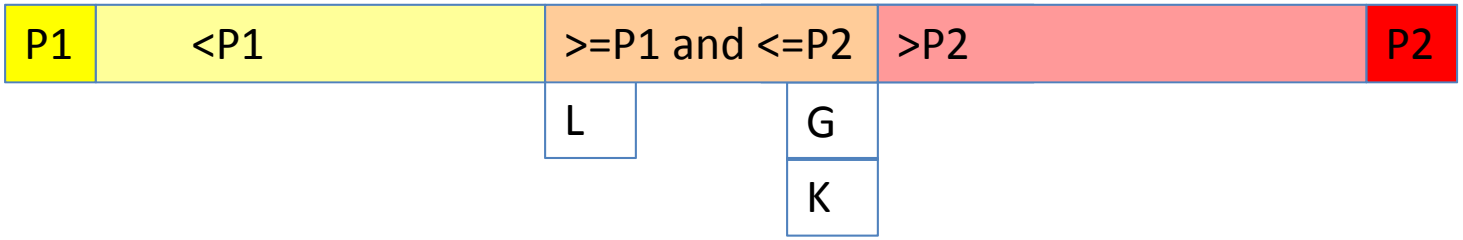
increment K

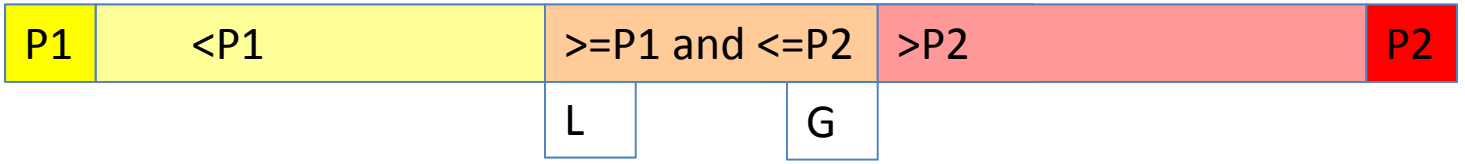




Starting from K=L we keep checking elements in this way as long as K is less than or equal to G.

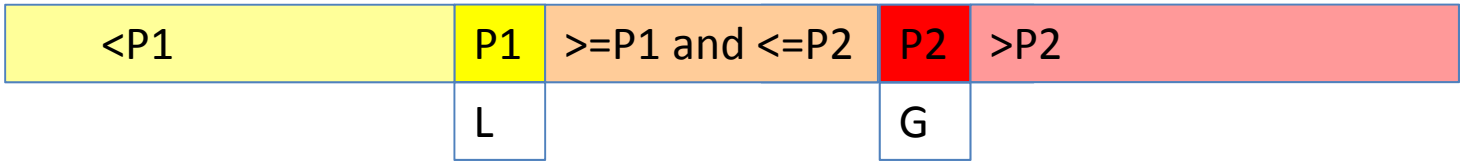
```
for K from L to G
        if array[K] < P1
                swap array[K] with Array[L]
                increment L
        else if array[K] > P2
                while array[G] > P2 and K< G
                        decrement G
                swap array[K] with array[G]
                decrement G
                if array[K] < P1
                        swap array[K] and array [L]
                        increment L
        increment K
```

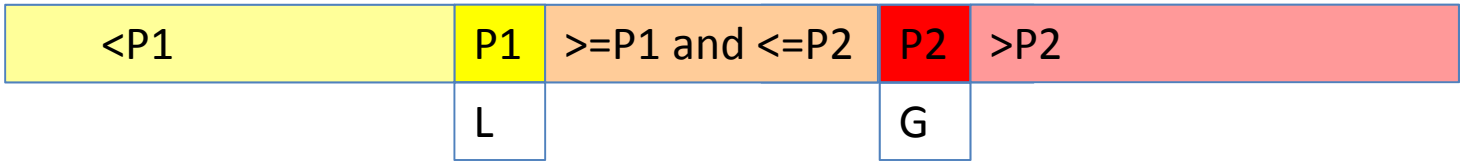




P1 and P2 must now be moved into the correct position

decrement L
decrement G
swap P1 with array[L]
swap P2 with array[G]





Then we recursively sort the partitions:

sort from left to L-1
sort from L+1 to G-1
sort from G+1 to right

recursive algorithms need a base case, so make sure we don't just keep sorting forever

```
if the partition contains 1 element or fewer
        return
```