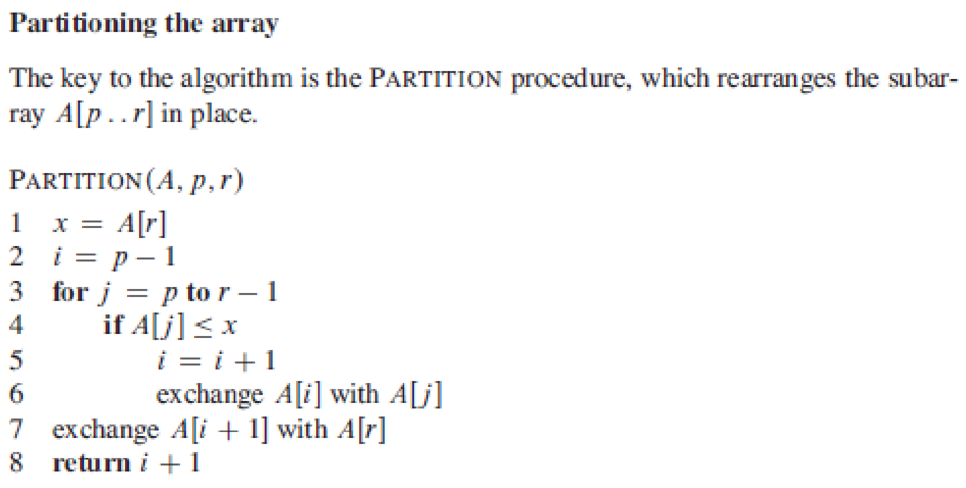
SORTING EXPLINATIONS!!

1. Design an algorithm to rearrange elements of a given array of *n* real numbers so that all its negative elements precede all its positive elements. (Hints: see the Hoare-Partition in CLRS 7-1. Perhaps modify the Hoare-partition to solve the problem.)

```
INPUT: A[1..n] - an array of integers (integers can be either negativ or positive)
OUTPUT: An array of the given numbers where the negative integers precedes the positive
ones.

KuntzSORT(A[])
result[] = A[].size
left = 0
right = n-1

for i = 0 to n
    if(A[i] < 0)
        result[left++] = A[i]
    else
        result[right--] = A[i]

return result
```

2. CRLS 7.2-2. Solve it using two versions of the quick sort: one uses the Partition algorithm on p171 and the other one uses the Hoare-Partition in CLRS 7-1. Do they always split the data into the same partitions?

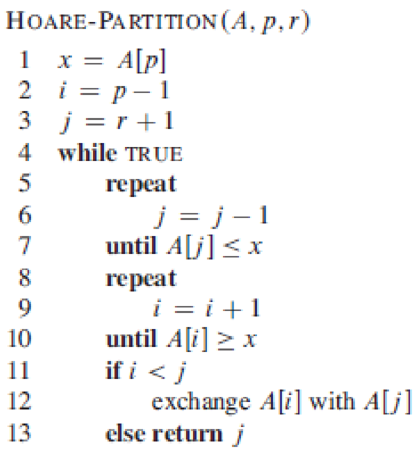
Array of all the same values.



Partitioning the array

The key to the algorithm is the PARTITION procedure, which rearranges the subarray $A[p..r]$ in place.

```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```



```
HOARE-PARTITION($A, p, r$)
$x = A[p]$
$i = p - 1$
$j = r + 1$
while TRUE
    repeat
        $j = j - 1$
    until $A[j] \leq x$
    repeat
        $i = i + 1$
    until $A[i] \geq x$
    if $i < j$
        exchange $A[i]$ with $A[j]$
    else return $j$
```

NOT SOLVED!!!!!

3. Write down a small array of integers. Try to run insertion sort, merge sort, bubble sort, selection sort, and quick sort on the array. Make sure that you understand all the sorting algorithms that we have learned so far.

Array = [2,8,1,6,9,4,2,6,8,1]

Insertion

Start from the left. Find first number not in order. Move that number back into correct position. Search again for the number that is not in order. Move that number back into correct position... REPEAT.

Start: [2,8,1,6,9,4,2,6,8,1] -> Find first number not in order.

1 is lesser then 8, select 1: [2,8,1,6,9,4,2,6,8,1] -> Find correct position for 1.

place 1 in the first spot: [1,2,8,6,9,4,2,6,8,1] -> search for first number not in order.

6 is lesser than 8, select 6: [1,2,8,6,9,4,2,6,8,1] -> Find correct position for 6.

place 6 in the right spot: [1,2,6,8,9,4,2,6,8,1] -> search for first number not in order.

.....and so on.

Merge sort

Uses divide-and-conquer. Divides array into two array.. and keeps doing that until we have arrays one digit. Then combine the two arrays and keep doing that until one big array is left. The combining is done by finding the smallest integer from the two arrays and put it into the combines array.. repeat until merged.

Start: [2,8,1,6,9,4,2,6,8,1] -> divide array into two arrays.

divide 1: $[2,8,1,6,9][4,2,6,8,1]$ -> arrays are not only containing one element, split!

divide 2: $[2,8][1,6,9][4,2][6,8,1]$ -> arrays are not only containing one element, split!

divide 3: $[2][8][1][6,9][4][2][6][8,1]$ -> arrays are not only containing one element, split!

divide 4: $[..][..][..][6][9][..][..][..][8][1]$ -> arrays are not only containing one element, split!

start merging: $[..][..][..][6][9][..][..][..][8][1]$ -> look at the first two integers from the arrays. Take the smallest of the two. Repeat until both arrays are empty.

merge: $[2][8][1][6,9][4][2][6][1,8]$ -> look at the first two integers from the arrays. Take the smallest of the two. Repeat until both arrays are empty.

merge: $[2,8][1,6,9][2,4][1,6,8]$ -> look at the first two integers from the arrays. Take the smallest of the two. Repeat until both arrays are empty.

merge: $[1,2,6,8,9][1,2,4,6,9]$ -> look at the first two integers from the arrays. Take the smallest of the two. Repeat until both arrays are empty.

merge: $[1,1,2,2,4,6,6,8,9,9]$ -> look at the first two integers from the arrays. Take the smallest of the two. Repeat until both arrays are empty.

Bubble sort

Bubble sort starts from one end of the array and check elements in pairs and swaps if need for the correct order. This is done repeatedly until sorted.

Start: [2,8,1,6,9,4,2,6,8,1] -> check the two first values, are they in order?

they are: [2,8,1,6,9,4,2,6,8,1] -> check the next two values, are they in order?

they are not, switch: [2,1,8,6,9,4,2,6,8,1] -> check the next two values, are they in order?

they are not, switch: [2,1,6,8,9,4,2,6,8,1] -> check the next two values, are they in order?

they are: [2,1,6,8,9,4,2,6,8,1] -> check the next two values, are they in order?

.... skip until first run-through is done: [2,1,6,8,4,2,6,8,1,9]

Repeat, second run-through is done: [1,2,6,4,2,6,8,1,8,9]

.. do this repeatedly until sorted.

Selection sort

Selection sort, finds the smallest element, then swap that element with the first spot. Then search again for the smallest value, and then swap that value with the second spot.. this continues until array is sorted.

Start: [2,8,1,6,9,4,2,6,8,1] -> find smallest value.

value 1 is found, swap it with the first spot: [1,8,2,6,9,4,2,6,8,1] -> find smallest value:

value 1 is found, swap it with the second spot: [1,1,2,6,9,4,2,6,8,8] -> find smallest value:

value 2 is found, swap it with the fourth spot: [1,1,2,2,9,4,6,6,8,8] -> find smallest value:

.....repeat until sorted.

Quick sort

Quick sort, finds a pivot point in the array (just choose the first value). Then it sorts the array so that all values on the left side of the pivot is smaller than the pivot value and the elements on the right side is greater than the pivot value. Two sub arrays is then created with the pivot value as a separator, and quick sort is then run on the two sub arrays. (The pivot value is not a part of the two subarrays).

Start: [2,8,1,6,9,4,2,6,8,1] -> choose first value as a pivot, quick sort values:

quick sorted: [1,1,2,2,8,6,9,4,6,8] -> split array at pivot(s) and call quick sort on the new subarrays:

quick sort the subarrays: [1,1,2] 2 [6,4,6,8,9,8] -> split array at pivot(s) and call quick sort on the new subarrays:

quick sort the subarrays: [1]1[2] 2 [6,4,6] 8 [9,8] -> split array at pivot(s) and call quick sort on the new subarrays:

Repeat.. until sorted.