Another Divide-Conquer Sort

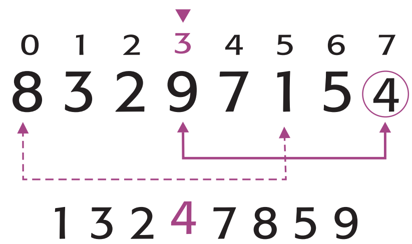
Unlike *mergesort* where we split list at mid-point, now pick split positions in a special way.

Change split strategy

- ✎ Elements a[k<s]<a[s]
- ✎ Elements a[k>s]>a[s]

Example

Quiz
Using 4 as split element, what should be the split position (index)? **Hint**: 3 elements smaller.



Therefore

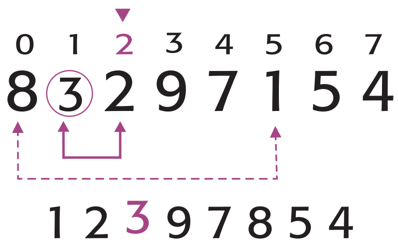
- ✎ Elements a[k<3]<4
- ✎ Elements a[k>3]>4
- ✎ a[3]=4 in sort position

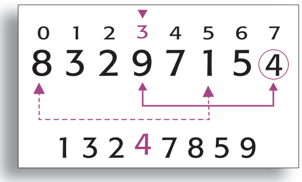
Divide-Conquer Sort
Partition Around Pivot

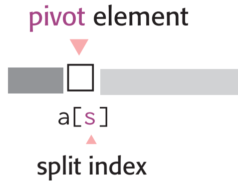
- ➪ **Split index, s**
- ➪ **Pivot element**

Split index (position) depends on choice of **pivot** element.

May pick <u>any element</u> to be pivot, choice determines where list is split.







A **partition** puts a key in sorted position, that's why it works well for the selection problem when the desired key happens to be picked as pivot.

Can it be used to sort lists?

Divide-Conquer Sort
Sort Strategy

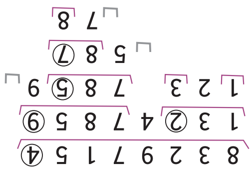
To sort the whole list

Repeatedly partition left and right sub-lists around some pivot until each element is in sorted position

Try by hand!

As convention, always choose pivot to be right-most element, stop if less than 2 elements, always sort left before right.

0	1	2	3	4	5	6	7
8	3	2	9	7	1	5	(4)



Simply

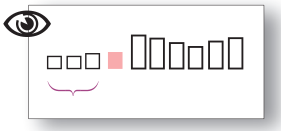
Quiz
What is the smallest list which will be partitioned?

Algorithm *quicksort*
Input Subarray $a[l \mathinner{..} r]$ of keys indexed $0 \mathinner{..} n-1$
Output Sorted subarray in nondecreasing order

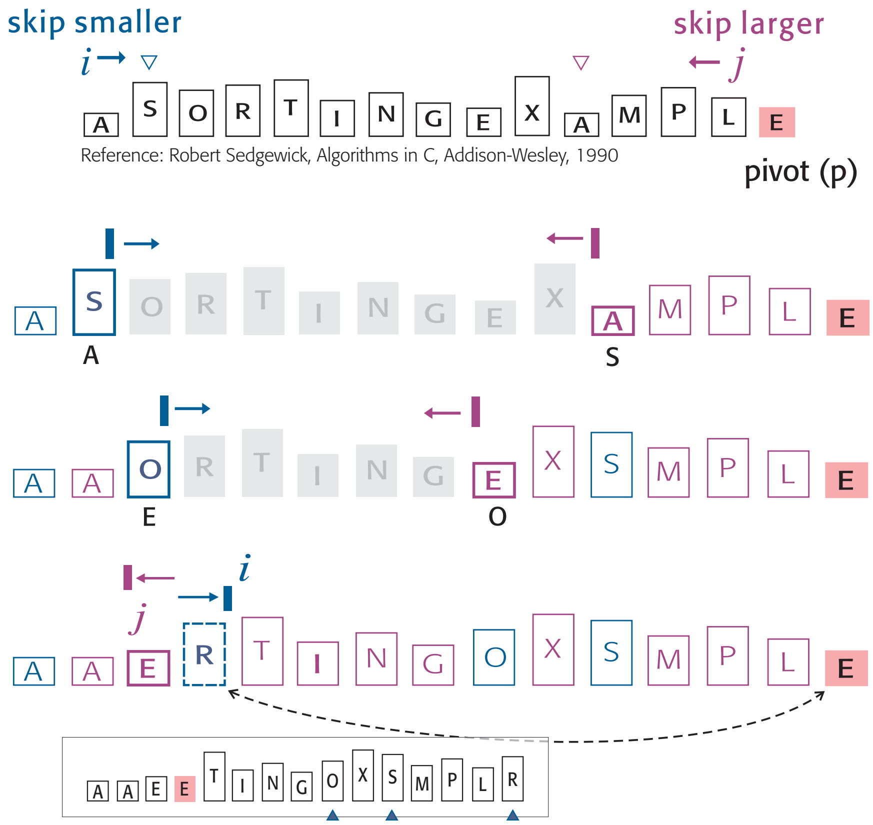
if $l < r$ **then**
    $s \leftarrow partition(a[l \mathinner{..} r])$ ▷ `around any element in subarray`
    $quicksort(a[l \mathinner{..} s-1])$
    $quicksort(a[s+1 \mathinner{..} r])$

Human insight not useful, <u>any</u> partition procedure will do

Divide-Conquer Sort
A Partition Procedure



Keep smaller elements this side of list where they belong.



Note *i* stops short of *j* ($i < j$), i.e., scan not done.

Note *i* skips ahead of *j* ($i > j$), no more items to check, partition ends (note extra runaway *j* comp).

Exchange pivot with most distant larger than *p* (in this case: R).

A Quicksort Algorithm

Both versions required ➪ **Hoare partition**
➪ **Sedgewick partition**

Quiz
How many comps by Hoare to split a list of *n* elements?

Quiz
Where would *i*, *j* start if pivot was chosen left-most instead of right?

Quiz
Can Lomuto's partition be used to *quicksort*? Try it.

Algorithm *quicksort2*

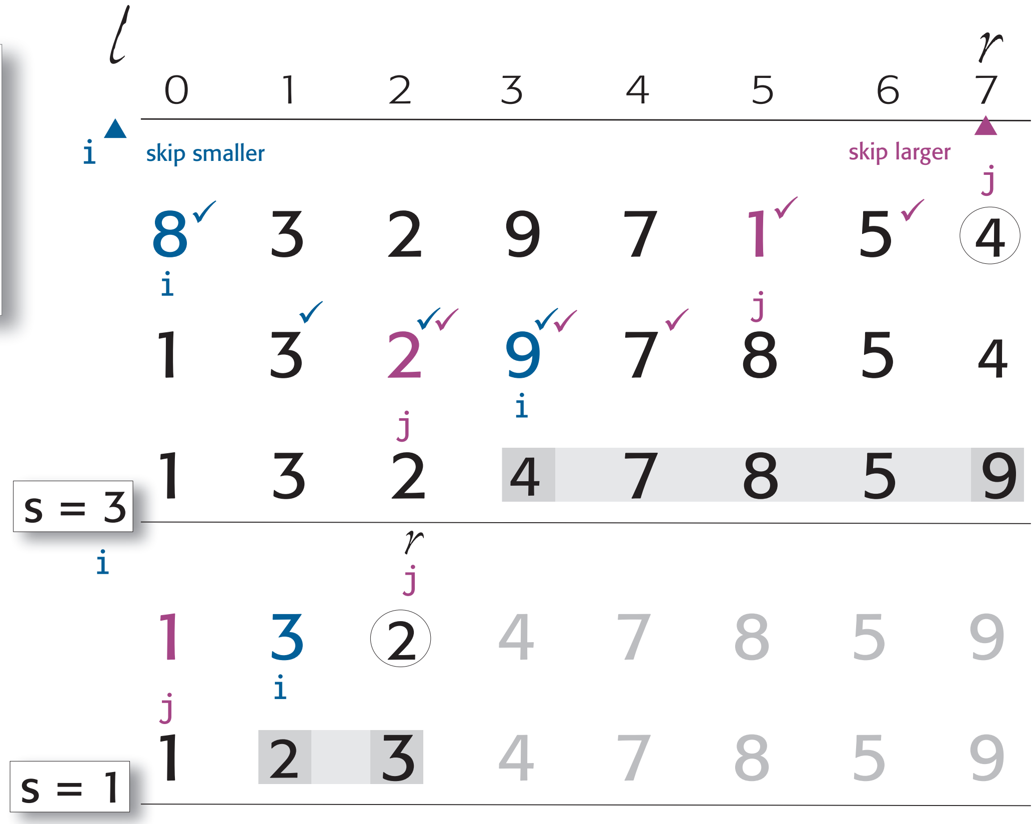
Reference: Robert Sedgewick, Algorithms in C, Addison-Wesley, 1990

1: **if** $l < r$ **then**
2: $p \leftarrow a[r],\ i \leftarrow l-1,\ j \leftarrow r$ ▷ select rightmost as pivot
3: **loop** ▷ break condition Line 6
4: **while** $a[{+}{+}i] < p$ **do** — skip smaller — find misplaced front ▷ low index scan
5: **while** $a[{-}{-}j] > p$ **do** — skip larger — find misplaced back ▷ high index scan
6: **if** $i \ge j$ **then break** ▷ quit scan if subarray done
7: swap $a[i], a[j]$ ▷ swap misplaced elements
8: swap $a[i], a[r]$ ▷ split point i, swap p in place
9: $quicksort2(a[l\ ..\ i-1])$
10: $quicksort2(a[i+1\ ..\ r])$

Algorithm *quicksort*
1: **if** $l < r$ **then**
2: $s \leftarrow partition(a[l\ ..\ r])$
3: $quicksort\ (a[l\ ..\ s-1])$
4: $quicksort\ (s+1\ ..\ r])$

A Quicksort Algorithm Example

$p \leftarrow a[r]$, $i \leftarrow l - 1$, $j \leftarrow r$
loop
  while $a[{++}\,i] < p$ **do**
  while $a[{-}{-}\,j] > p$ **do**
  if $i \geq j$ **then break**
  swap $a[i]$, $a[j]$
swap $a[i]$, $a[r]$



Quiz How many comparisons were performed after s=3? Hint: count them.

Sublist has to be at least of size 2 to partition (why?)

A Quicksort Algorithm Example (cont.)

Exercise What's the <u>total</u> number of comparisons performed by the *quicksort*? **Hint**: write list reduction sequence.

Compare to bubble sort (from formula.)

Quiz What are indices of the left and right sub-arrays after a split at i=7?

Exercise
Complete the sort, use the visualization link in the course website (Ans. last slide, next lecture).

Exercise Trace recursive calls of the *quicksort* (show parameters for each).