Quicksort

Koffman & Wolfgang

- kapitel 8, avsnitt 9

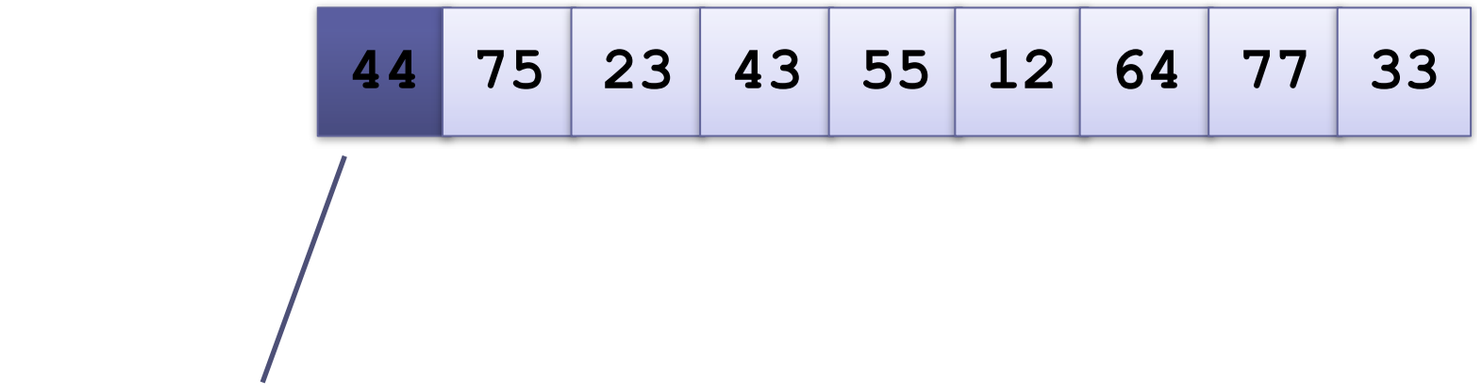
Quicksort

Quicksort väljer ett spcifikt värde (kallat *pivot*),
och delar upp resten av fältet i två delar:

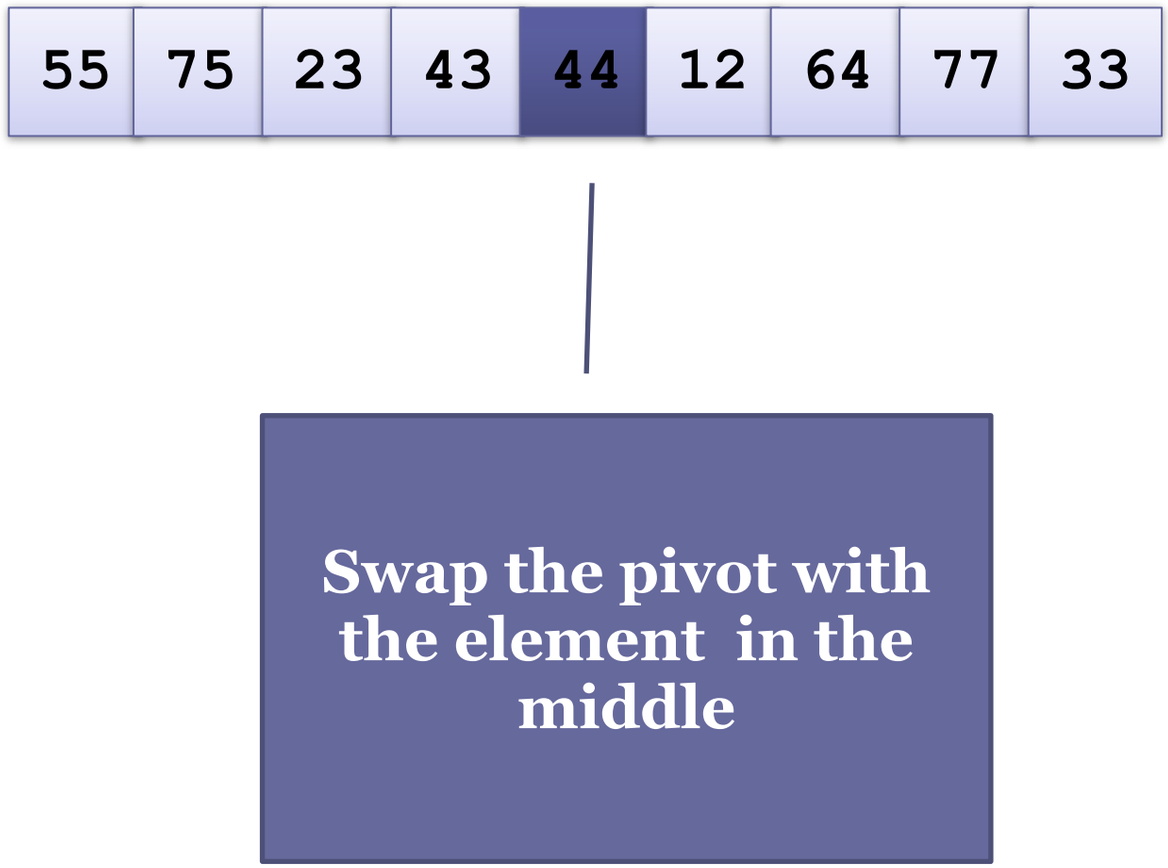
- alla element som är ≤ pivot läggs i vänstra delen
- alla element som är > pivot läggs i högra delen
- pivoten placeras mittemellan
- sortera vänstra och högra delen var för sig, rekursivt

Trace of Quicksort

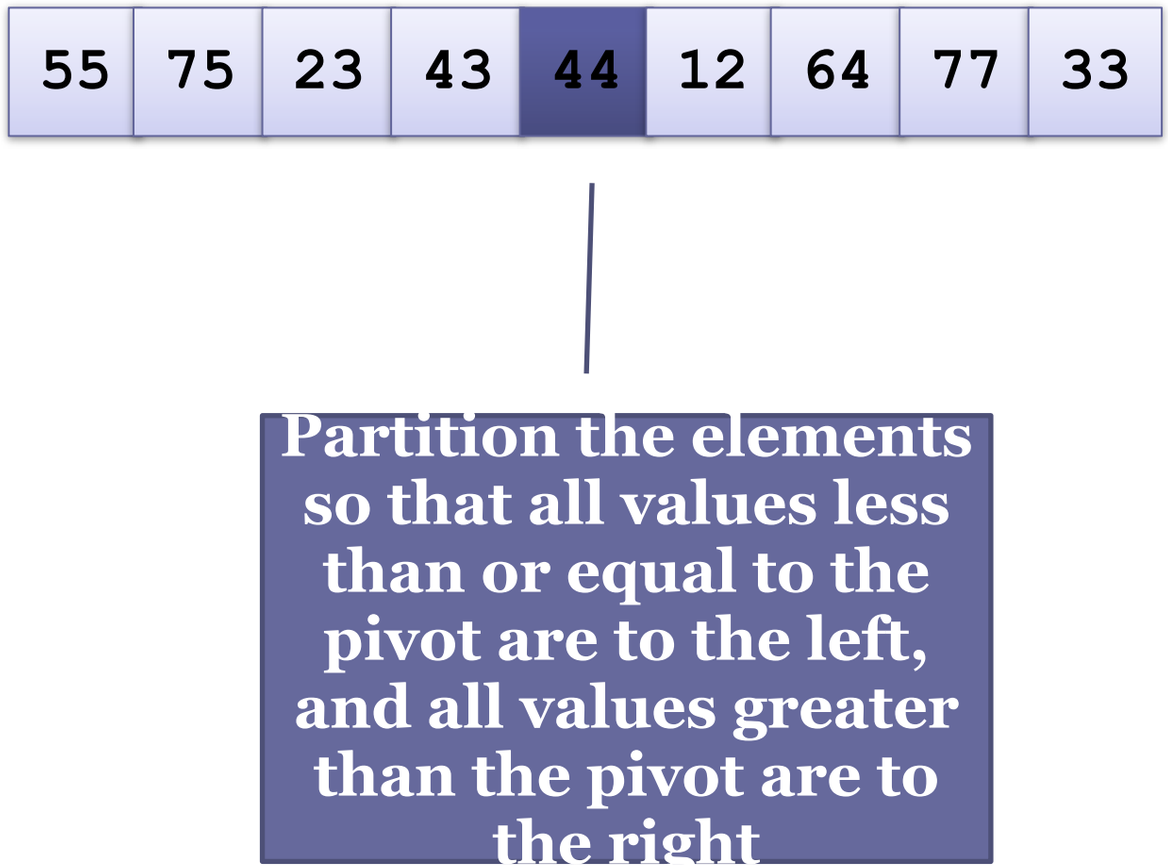
Trace of Quicksort (cont.)



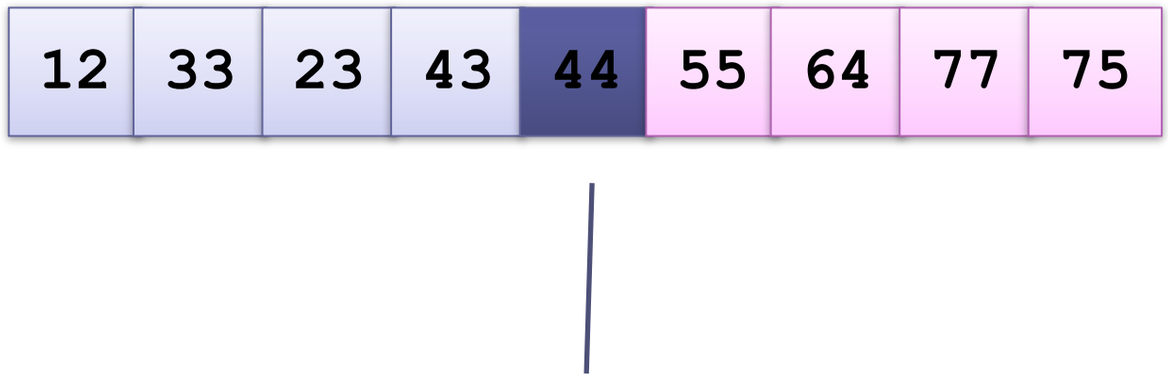
Trace of Quicksort (cont.)



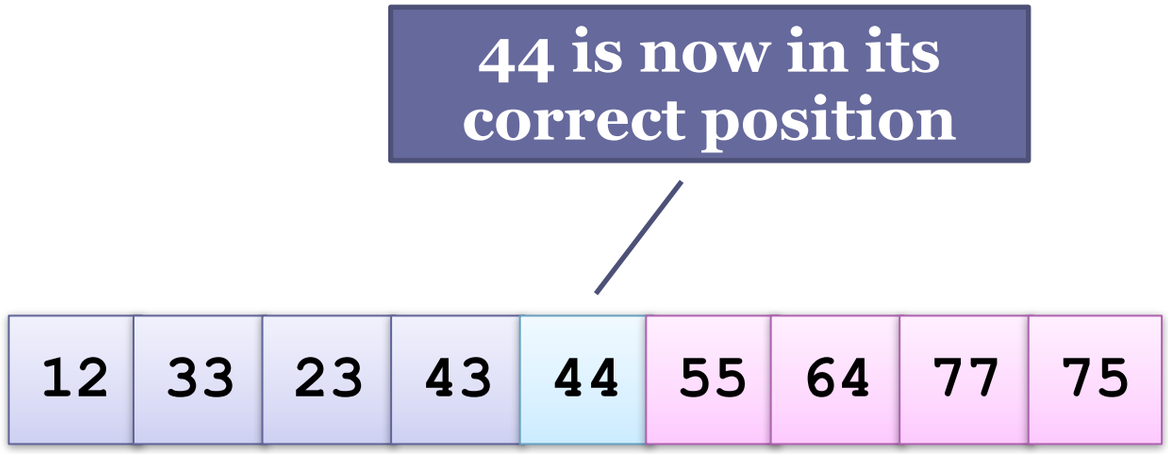
Trace of Quicksort (cont.)



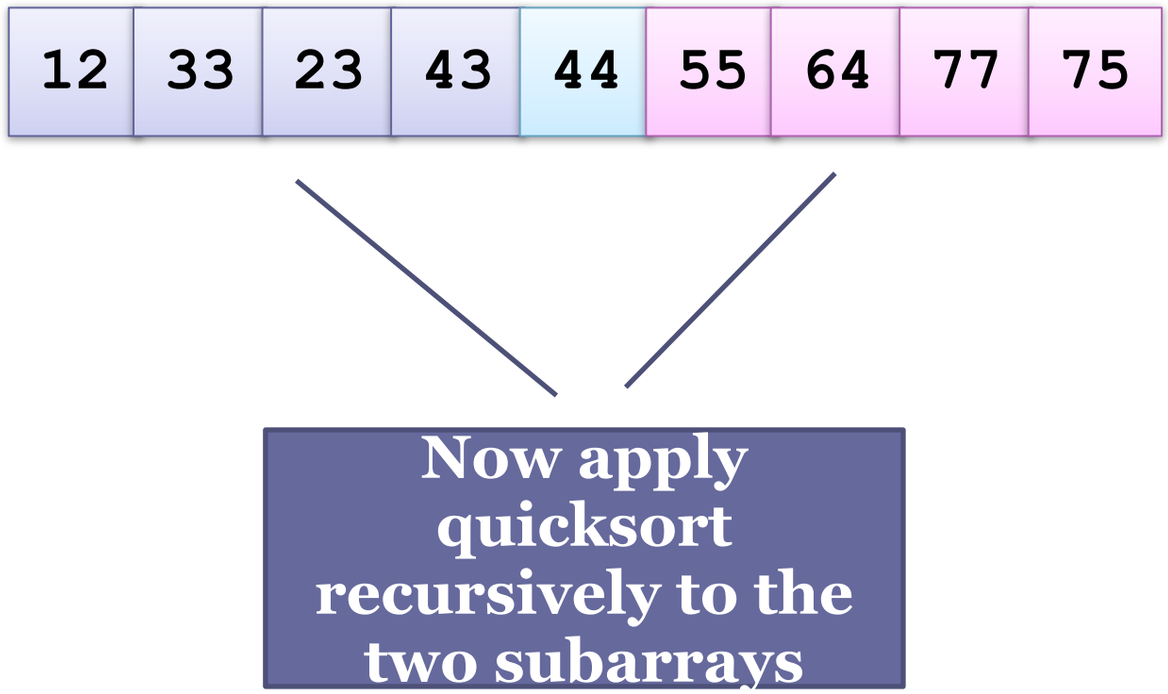
Trace of Quicksort (cont.)



Quicksort Example(cont.)



Trace of Quicksort (cont.)



Trace of Quicksort (cont.)

Pivot value = 12

Trace of Quicksort (cont.)

Pivot value = 12

Trace of Quicksort (cont.)

Pivot value = 33

Trace of Quicksort (cont.)

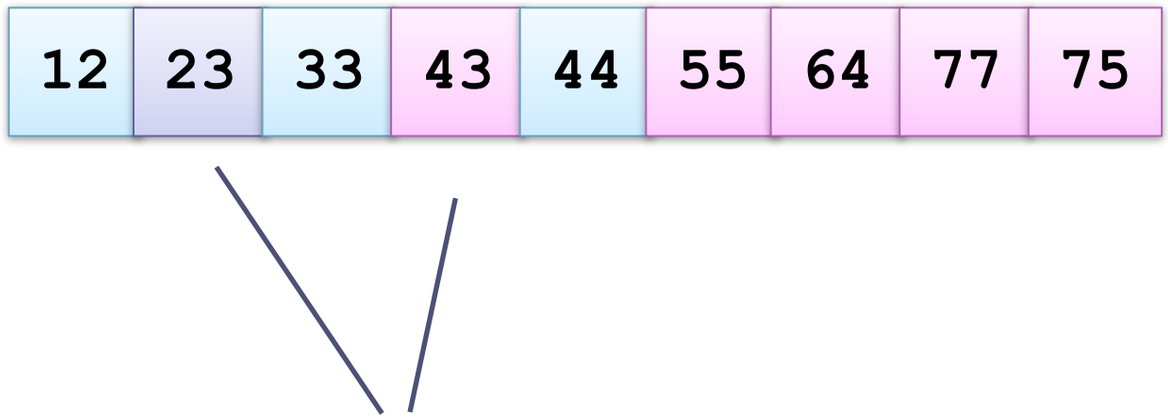
Pivot value = 33

Trace of Quicksort (cont.)

Pivot value = 33

Trace of Quicksort (cont.)

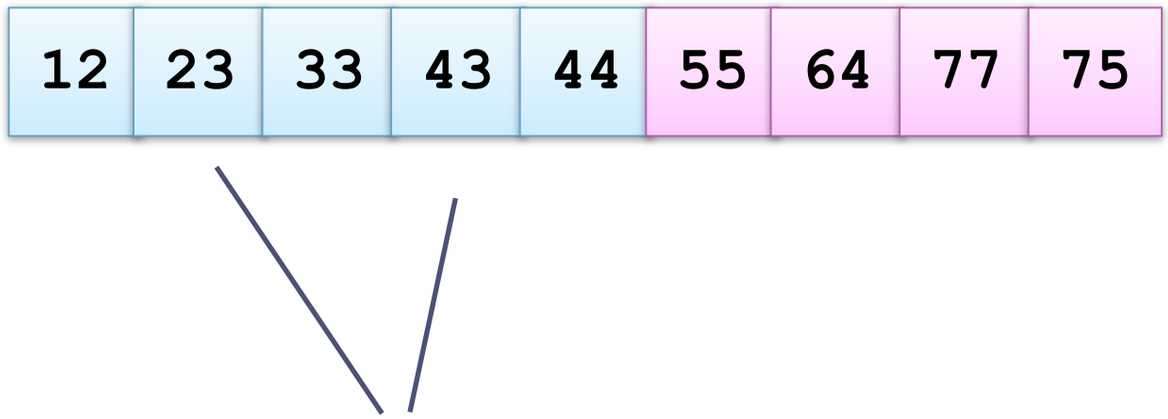
Pivot value = 33



Left and right subarrays have single values; they are sorted

Trace of Quicksort (cont.)

Pivot value = 33



Left and right subarrays have single values; they are sorted

Trace of Quicksort (cont.)

Pivot value = 55

Trace of Quicksort (cont.)

Pivot value = 64

Trace of Quicksort (cont.)

Pivot value = 77

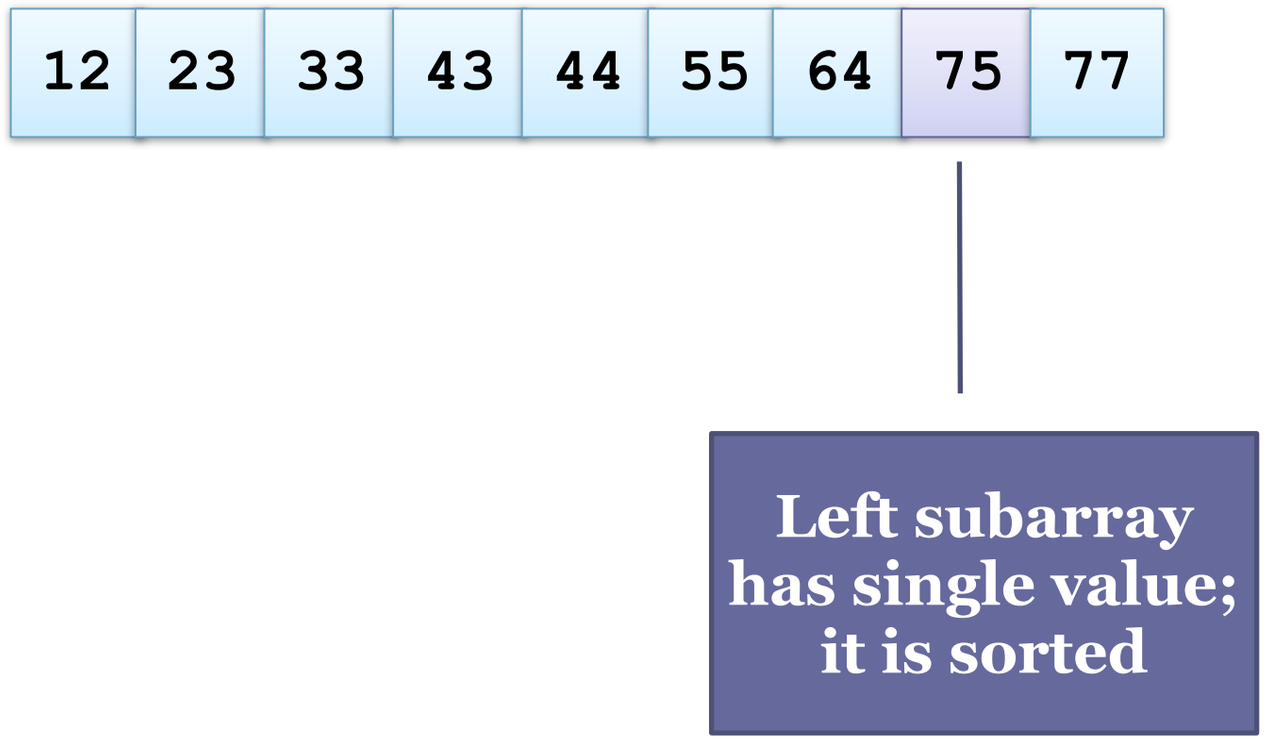
Trace of Quicksort (cont.)

Pivot value = 77

Trace of Quicksort (cont.)

Pivot value = 77

Trace of Quicksort (cont.)



Trace of Quicksort (cont.)

Analys av Quicksort

Om pivoten är ett slumpmässigt värde från det aktuella fältet,

- så kommer statistiskt sett pivoten att ligga i mitten av fältet
- vänster och höger halva kommer (statistiskt sett) att bli lika stora
- dvs, varje rekursivt anrop kommer att sortera ett hälften så stort fält (statistiskt sett)
- dvs, samma resonemang som för Mergesort
- fast Quicksort är in-place

Algorithm for Partitioning

If the array is randomly ordered, it does not matter which element is the pivot.

For simplicity we pick the element with subscript `first`

Trace of Partitioning (cont.)

If the array is randomly ordered, it does not matter which element is the pivot.

For simplicity we pick the element with subscript `first`

Trace of Partitioning (cont.)

For visualization purposes, items less than or equal to the pivot will be colored blue; items greater than the pivot will be colored light purple

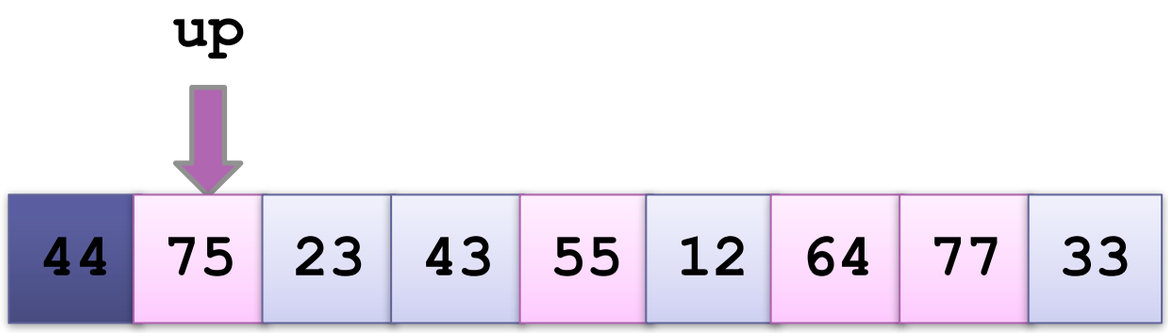
Trace of Partitioning (cont.)

For visualization purposes, items less than or equal to the pivot will be colored blue; items greater than the pivot will be colored light purple

Trace of Partitioning (cont.)

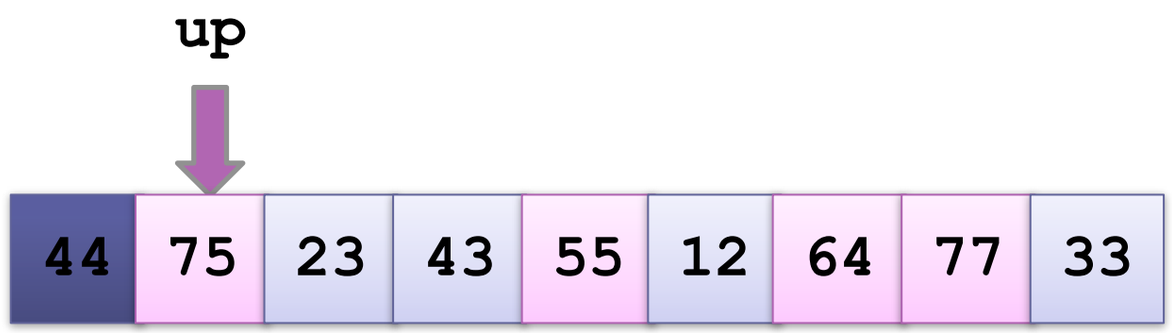
Search for the first value at the left end of the array that is greater than the pivot value

Trace of Partitioning (cont.)



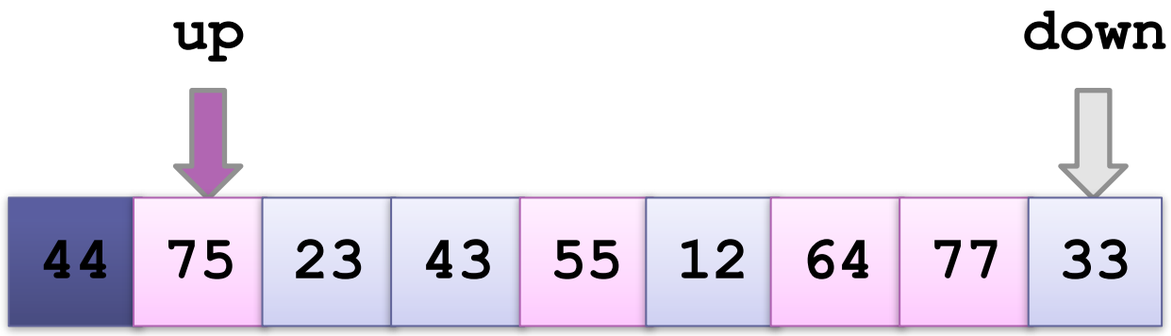
Search for the first value at the left end of the array that is greater than the pivot value

Trace of Partitioning (cont.)



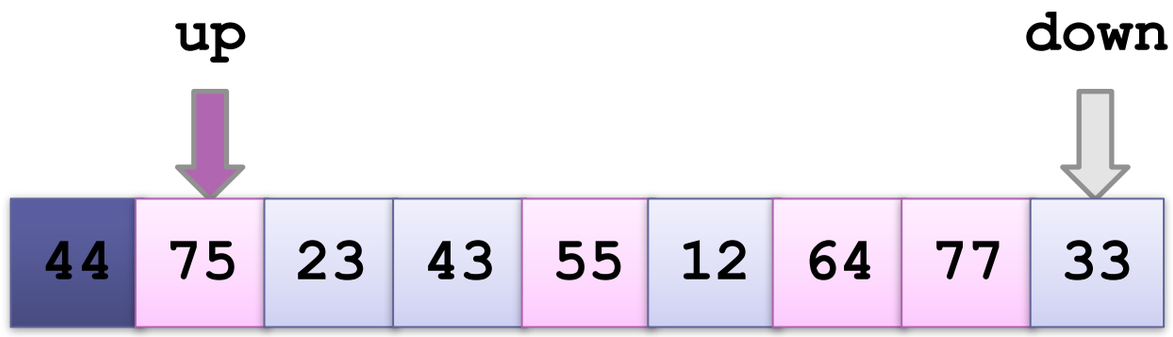
Then search for the first value at the right end of the array that is less than or equal to the pivot value

Trace of Partitioning (cont.)



Then search for the first value at the right end of the array that is less than or equal to the pivot value

Trace of Partitioning (cont.)



Exchange these values

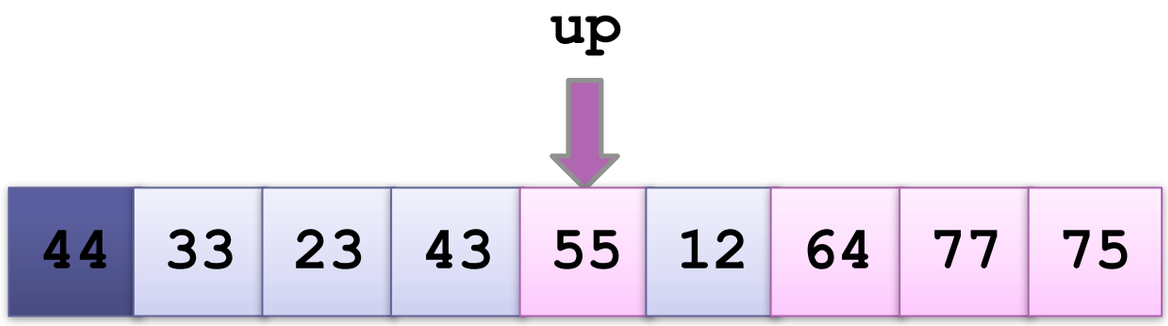
Trace of Partitioning (cont.)

Exchange these values

Trace of Partitioning (cont.)

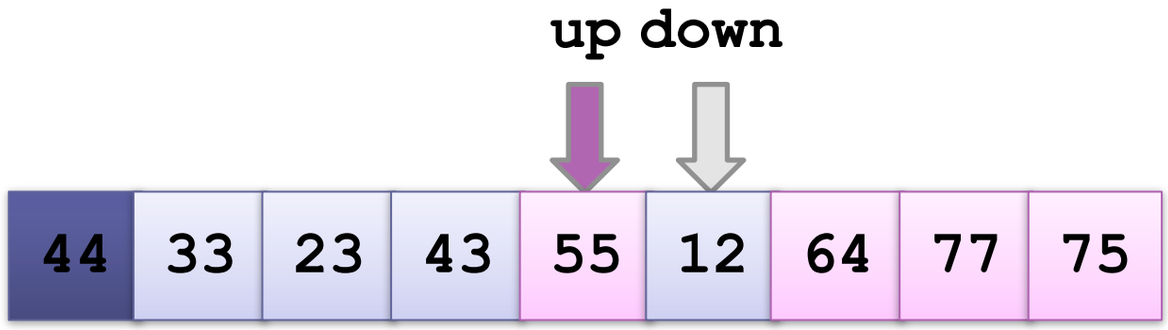
Repeat

Trace of Partitioning (cont.)



Find first value at left end greater than pivot

Trace of Partitioning (cont.)



Find first value at right end less than or equal to pivot

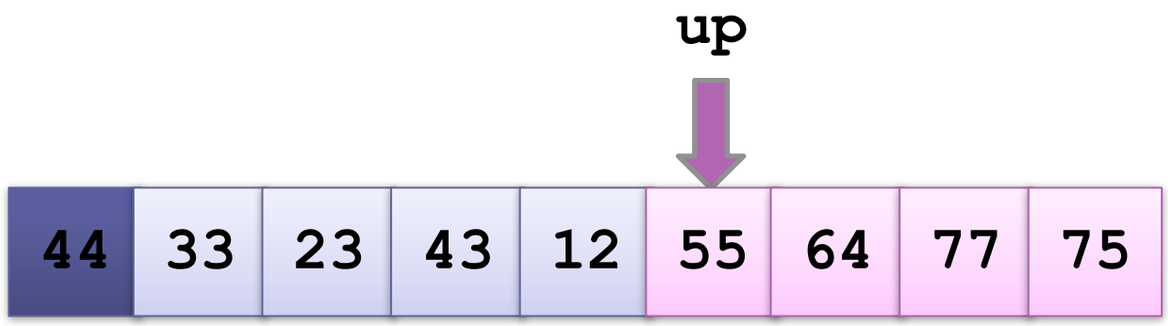
Trace of Partitioning (cont.)

Exchange

Trace of Partitioning (cont.)

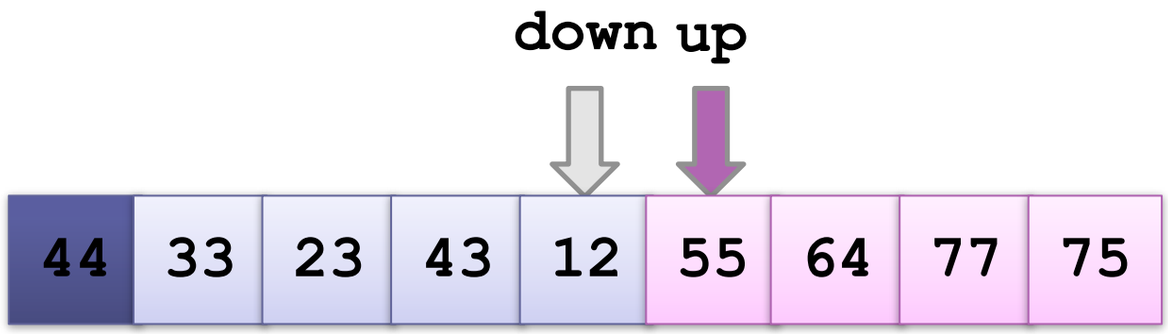
Repeat

Trace of Partitioning (cont.)



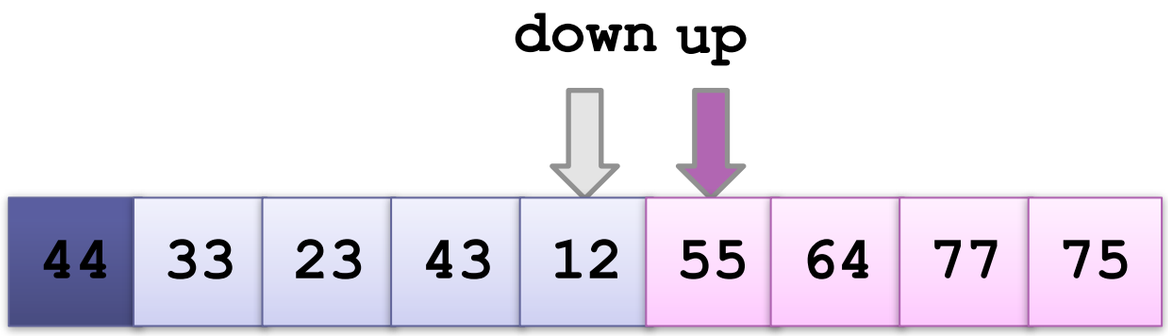
Find first element at left end greater than pivot

Trace of Partitioning (cont.)



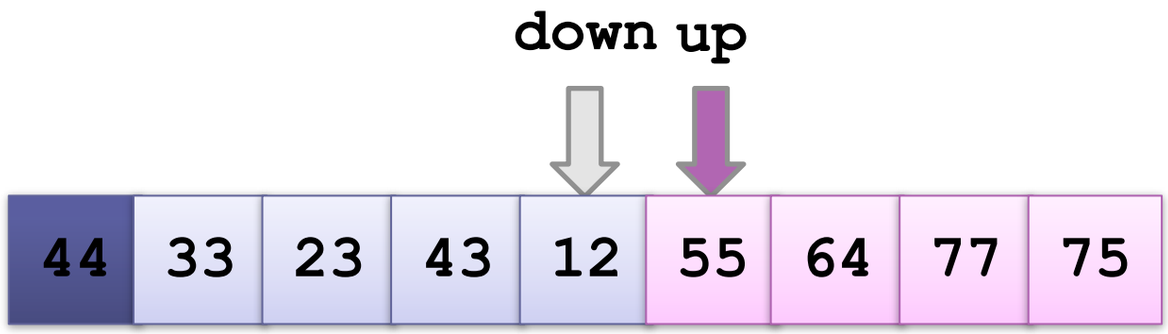
Find first element at right end less than or equal to pivot

Trace of Partitioning (cont.)



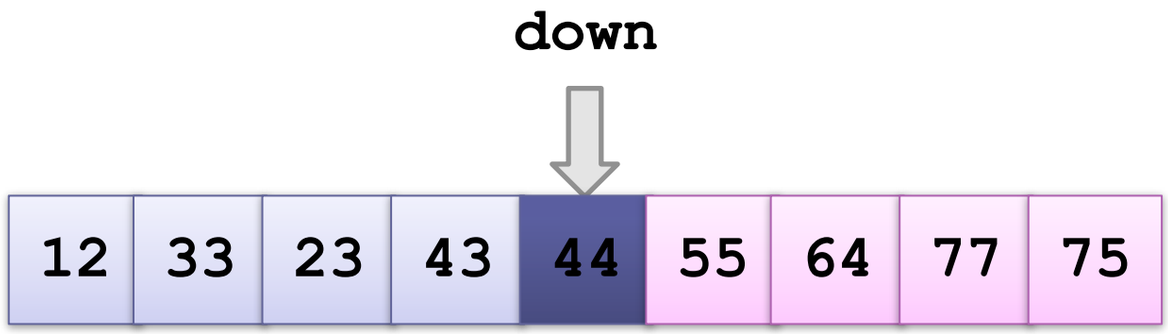
Since down has "passed" up, do not exchange

Trace of Partitioning (cont.)



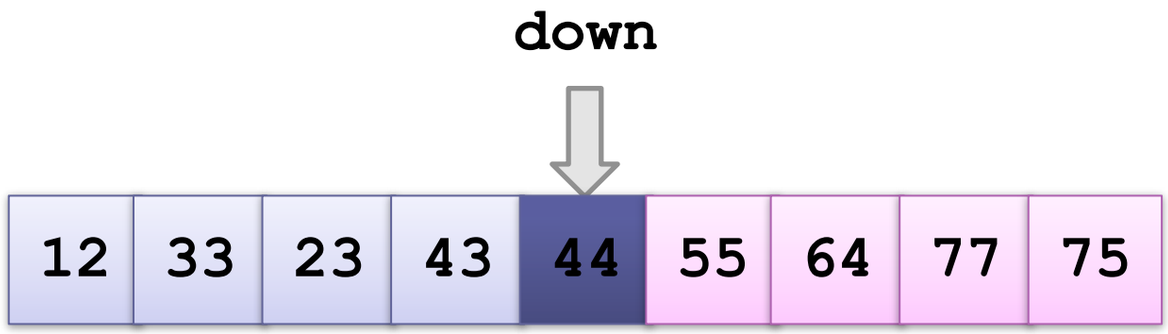
Exchange the pivot value with the value at down

Trace of Partitioning (cont.)



Exchange the pivot value with the value at `down`

Trace of Partitioning (cont.)



The pivot value is in the correct position; return the value of `down` and assign it to the pivot index `pivIndex`

Partitionsalgoritm

Algorithm for `partition` Method

1. Define the pivot value as the contents of `table[first]`.
2. Initialize `up` to `first` and `down` to `last`.
3. **do**
4. Increment `up` until `up` selects the first element greater than the pivot value or `up` has reached `last`.
5. Decrement `down` until `down` selects the first element less than or equal to the pivot value or `down` has reached `first`.
6. **if** `up < down` then
7. Exchange `table[up]` and `table[down]`.
8. **while** `up` is to the left of `down`
9. Exchange `table[first]` and `table[down]`.
10. Return the value of `down` to `pivIndex`.

Analys av Quicksort

Om pivoten väljs dumt så kan Quicksort bli ineffektiv:

- om det ena delfältet alltid råkar bli tomt
- t.ex., om man alltid tar första värdet som pivot, och listan redan är sorterad
- i det fallet blir komplexiteten $O(n^2)$, precis som insättningssortering

Bättre är att välja pivoten smartare:

- t.ex., utgå från tre värden — första, mittersta och sista elementet
- välj medianen av dessa som pivot
- detta minskar risken för dålig delning

Sorteringsalgoritmer

	Number of Comparisons		
	Best	Average	Worst
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$
Merge sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Heapsort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Quicksort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$