Sorting
QuickSort

QuickSort Concept

- Divide and Conquer…
  - Split the problem based on a "pivot" – any arbitrary value in the list
  - Partition the list:
    - items < pivot are to the left
    - items >= pivot are to the right

  - If a partition has more than one value, apply QuickSort to the partition

QuickSort Algorithm

Chapter 14 Special Topic 3

- if range size >1, partition range
  - pick a pivot (e.g. first element in range)
  - li=0, ri=partition size (li goes right, ri goes left)
  - while (li<ri)
    - while (list[li] < pivot) li++
    - while (list[ri] >= pivot) ri--
    - if li<ri, swap list[li] and list[ri]
- When li==ri,
  - everything from start of range to ri-1 is < pivot,
  - everything from ri to end of range is >= pivot
- Sort (recursively) left sub-range (to ri-1) and right sub-range (from ri)

QuickSort Example

Pivot=57

57	54	22	18	37	97	44	74	44	83

li

ri

list[li]<pivot
false

QuickSort Example

Pivot=57

57	54	22	18	37	97	44	74	44	83

li

ri

list[li]>=pivot
true

QuickSort Example

Pivot=57

57	54	22	18	37	97	44	74	44	83

li

ri

list[li]>=pivot
false

QuickSort Example

Pivot=57

57	54	22	18	37	97	44	74	44	83

li

ri

SWAP

QuickSort Example

Pivot=57

44	54	22	18	37	97	44	74	57	83

li

ri

SWAP

QuickSort Example

Pivot=57

44	54	22	18	37	97	44	74	57	83

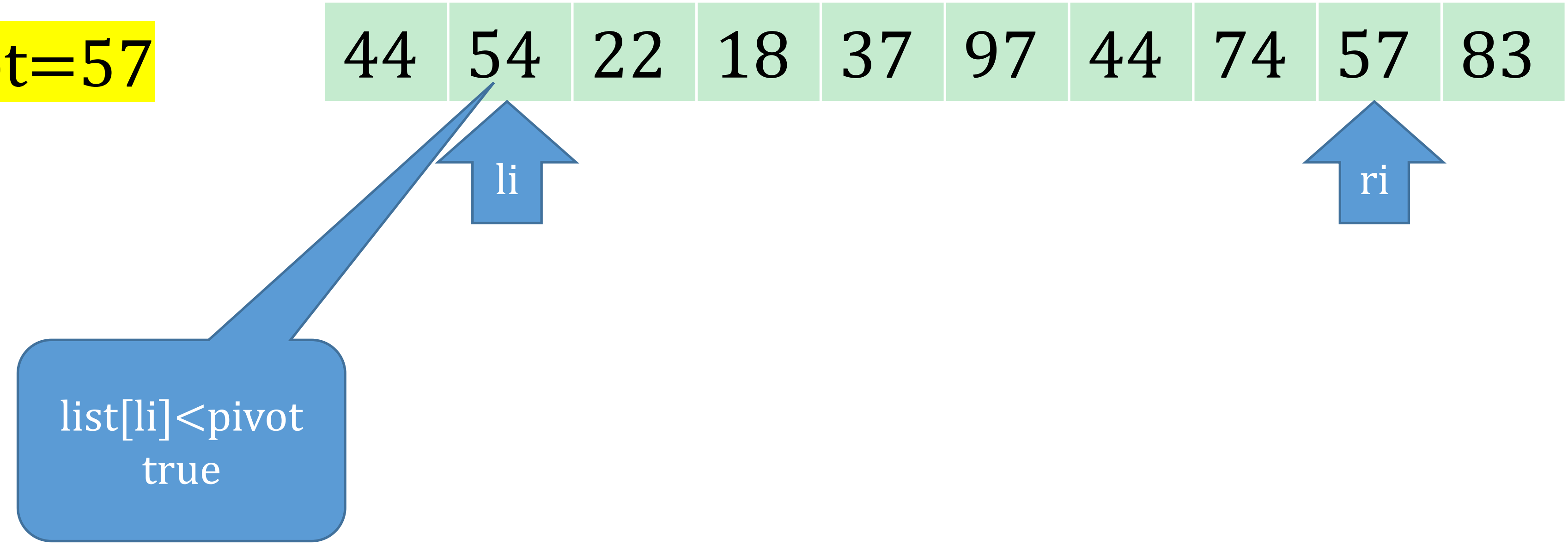
li

ri

list[li]<pivot
true

QuickSort Example

Pivot=57



QuickSort Example

Pivot=57

44	54	22	18	37	97	44	74	57	83

li

ri

list[li]<pivot
true

QuickSort Example

Pivot=57

44	54	22	18	37	97	44	74	57	83

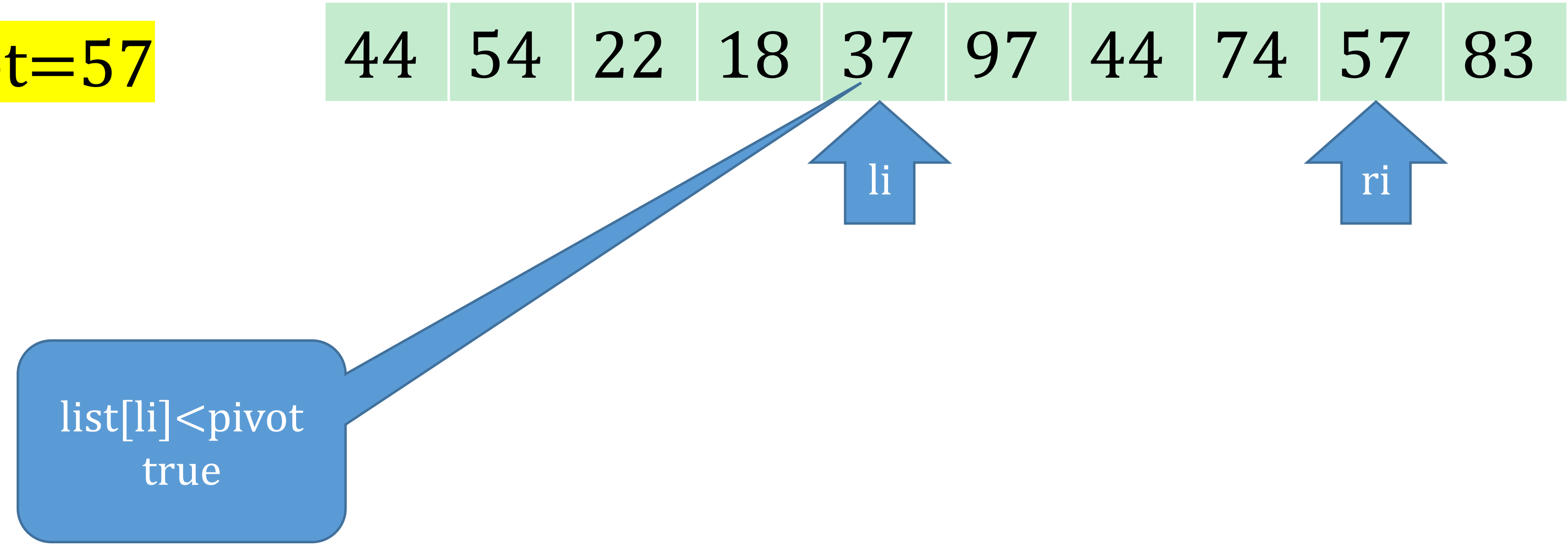
li

ri

list[li]<pivot
true

QuickSort Example

Pivot=57



QuickSort Example

Pivot=57

44	54	22	18	37	97	44	74	57	83

li

ri

list[li]<pivot
false

QuickSort Example

Pivot=57

44	54	22	18	37	97	44	74	57	83

li

ri

list[li]>=pivot
true

QuickSort Example

Pivot=57

44	54	22	18	37	97	44	74	57	83

li ri

list[li]>=pivot
true

QuickSort Example

Pivot=57

44	54	22	18	37	97	44	74	57	83

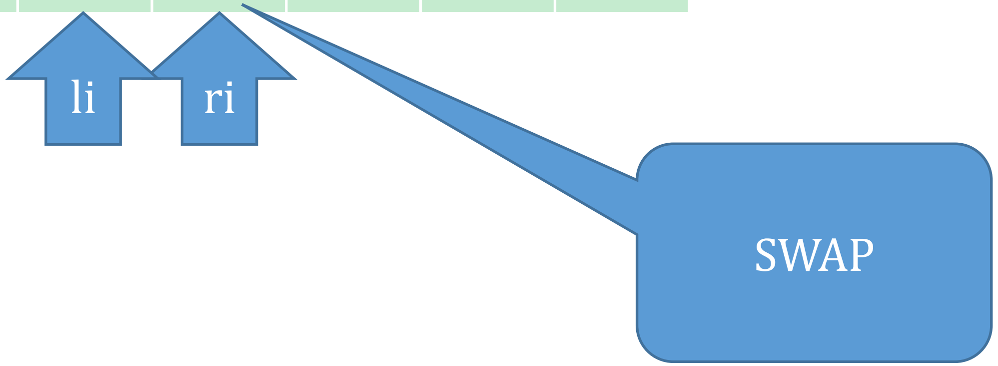
li ri

list[li]>=pivot
false

QuickSort Example

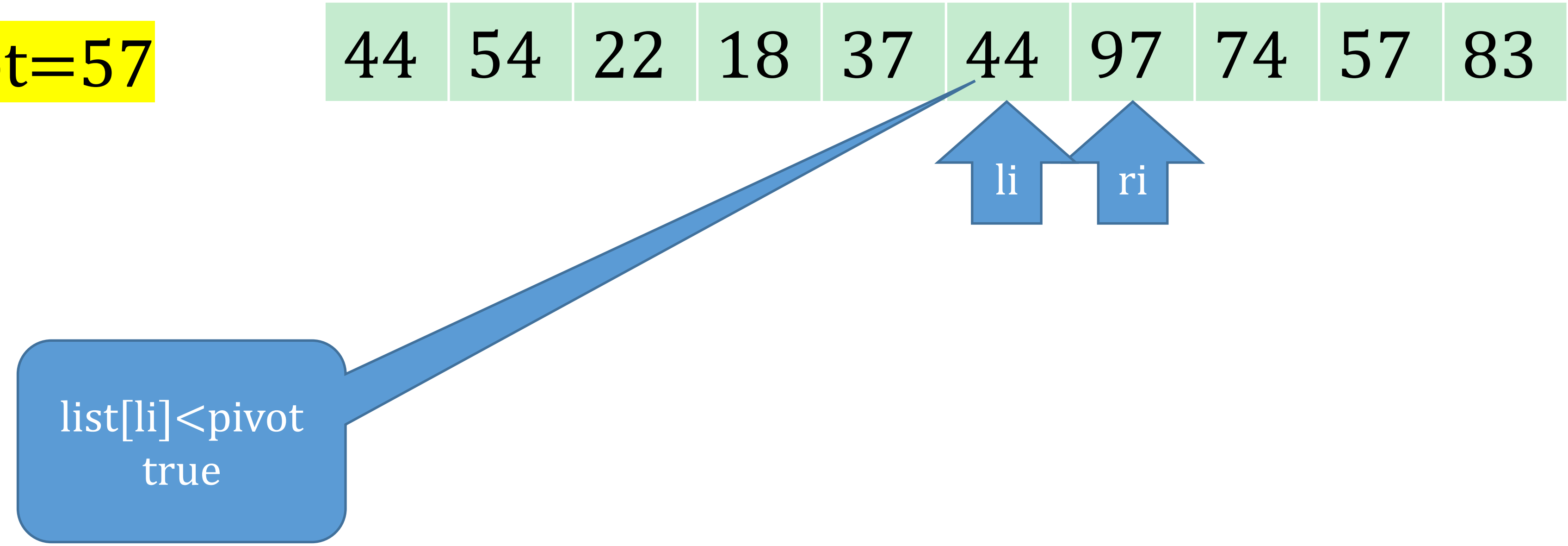
Pivot=57

44	54	22	18	37	97	44	74	57	83



QuickSort Example

Pivot=57



QuickSort Example

Pivot=57

44	54	22	18	37	44	97	74	57	83

li ri

li==ri
Partition

QuickSort Example

Pivot=57

44	54	22	18	37	44	97	74	57	83

ri

Partition

QuickSort Example

Pivot=44

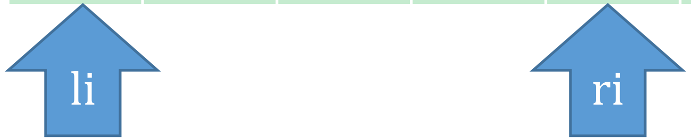
44	54	22	18	37	44	97	74	57	83

li ri

QuickSort Example

Pivot=44

44	54	22	18	37	44	97	74	57	83



QuickSort Example

Pivot=44

37	54	22	18	44	44	97	74	57	83

li ri

QuickSort Example

Pivot=44

37	54	22	18	44	44	97	74	57	83
	li			ri					

QuickSort Example

Pivot=44

37	54	22	18	44	44	97	74	57	83
	li		ri						

QuickSort Example

Pivot=44

37	18	22	54	44	44	97	74	57	83
	li		ri						

QuickSort Example

Pivot=44

37	18	22	54	44	44	97	74	57	83

li ri

QuickSort Example

Pivot=44

37	18	22	54	44	44	97	74	57	83

li

QuickSort Example

Pivot=44

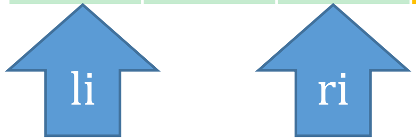
37	18	22	54	44	44	97	74	57	83

ri

QuickSort Example

Pivot=37

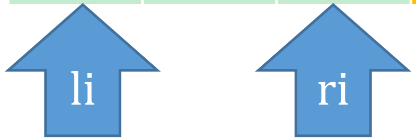
37	18	22	54	44	44	97	74	57	83



QuickSort Example

Pivot=37

22	18	37	54	44	44	97	74	57	83



QuickSort Example

Pivot=37

22	18	37	54	44	44	97	74	57	83

li ri

QuickSort Example

Pivot=37

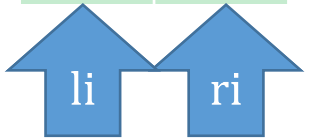
22	18	37	54	44	44	97	74	57	83

li i

QuickSort Example

Pivot=22

22	18	37	54	44	44	97	74	57	83



QuickSort Example

Pivot=22

QuickSort Example

Pivot=22

18	22	37	54	44	44	97	74	57	83

li i

QuickSort Example

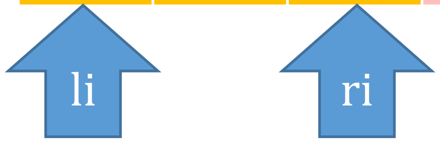
Pivot=54

18	22	37	54	44	44	97	74	57	83
			li		ri				

QuickSort Example

Pivot=54

18	22	37	44	44	54	97	74	57	83



QuickSort Example

Pivot=54

18	22	37	44	44	54	97	74	57	83
				li	ri				

QuickSort Example

Pivot=54

18	22	37	44	44	54	97	74	57	83

li i

QuickSort Example

Pivot=44

18	22	37	44	44	54	97	74	57	83
			li	ri					

QuickSort Example

Pivot=97

18	22	37	44	44	54	97	74	57	83
						li			ri

QuickSort Example

Pivot=97

18 22 37 44 44 54 83 74 57 97

li ri

QuickSort Example

Pivot=97

18 22 37 44 44 54 83 74 57 97

li i

QuickSort Example

Pivot=83

18 22 37 44 44 54 83 74 57 97

li ri

QuickSort Example

Pivot=83

18 22 37 44 44 54 57 74 83 97

li ri

QuickSort Example

Pivot=57

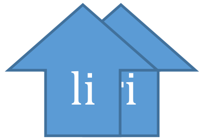
18 22 37 44 44 54 57 74 83 97

li ri

QuickSort Example

Pivot=57

18 22 37 44 44 54 57 74 83 97



QuickSort Analysis Example

57	54	22	18	37	97	44	74	44	83	10
44					44	97		57		6+4=10
37	18		54	44		83			97	3+3+3=9
22		37	44	44	54	57		83		3+2+2=7
18	22									2

18	22	37	44	44	54	57	74	83	97	Total: 38

$$10 \log_2 10 = 30.103 < 38 < 10^2 = 100$$

QuickSort Example

66	39	53	67	59	64	98	90	78	18	10
18			64		67				66	5+5=10
	39				66				67	4+4=8
		53				67			98	3+3=6
			64				90			2+2=4
			59	64			78	90		-----------
18	39	53	59	64	66	67	78	90	98	Total: 38

QuickSort Analysis

- Worst case:
  - Partition is always 1 / n-1 – Requires n partitions
  - Within partition, we must compare each element (partition size)

$$\sum_{s=2}^{n} s = \frac{n \times (n-1)}{2} = O(n^2)$$

- Best case:
  - Partition is in the middle of the range – Requires $\log_2(n)$ partitions
  - Partition size is $n/2^d$ where d is the depth
  - At a given depth, all n data items will be compared
  - Complexity is $O(n \times \log_2 n)$

Why don’t we count swaps?

- In any given partition, number of swaps <= partition size
- Therefore, if we are counting instructions

$$
\begin{aligned}
I(n) &= n * I_{comp} + SwapPercent * n * I_{swap} \\
&= n * \left(I_{comp} + SwapPercent * I_{swap}\right) \\
&= c * n
\end{aligned}
$$

$$O\left(I(n)\right) = n$$

What if array is pre-sorted?

- Picking first (or last) element in range as pivot is simple, but causes worst case if array is pre-sorted
- Therefore, we often pick the middle of the range as the pivot
  - Doesn’t change performance if elements start randomly
  - Improves performance if elements start sorted
  - There are still “worst case” scenarios, but much less likely

QuickSort Advantages

- Performance close to O(n log(n)) complexity
  - as good as we can get on sorts
  - Larger sized arrays come closer to best case performance
  - Better choice of pivot reduces chance of worst case
- No extra memory for array copies required!
  - All sorting is done on the original array
- This is the algorithm used in Java
  - Also used in C – hence C library function is “qsort”