Sorting

Chapter 7

Quick Sort

- One of the most popular fast sorting algorithms
- Quick sort overcomes the drawback of merge sort of creating an additional array
- Generally, quick sort is the most efficient algorithm for large arrays

Quick Sort

- Pick any element in the array (call it the pivot)
- Place the pivot in its correct position in the array
- Move all smaller elements to the left
- Move all bigger elements to the right
- Sort the left and right sides recursively

Quick Sort Example

8	10	5	1	3	20	13	7	2	12

Quick Sort Example

Select the pivot

8	10	5	1	3	20	13	7	2	12

pivot

Quick Sort Example

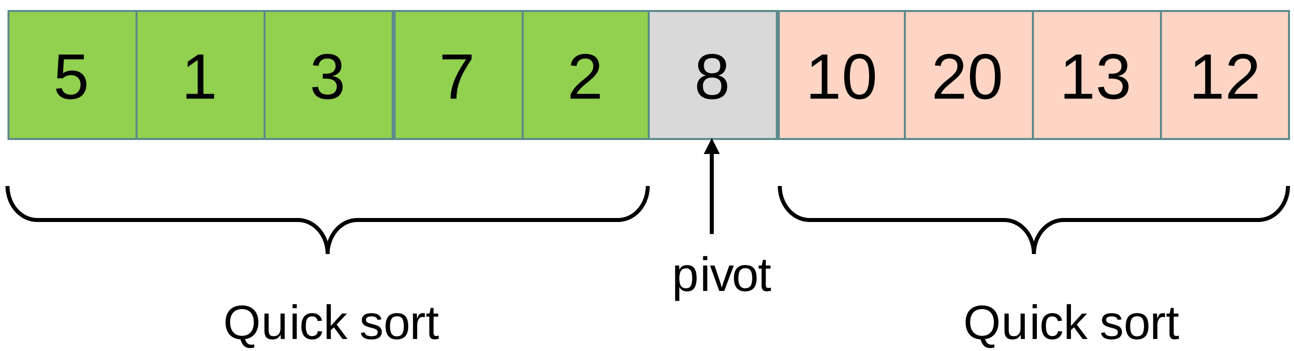
Partition the array

8	10	5	1	3	20	13	7	2	12

pivot

Quick Sort Example

Recursively sort both sides



Selecting the Pivot

- What would be a good pivot?
- What would be a bad pivot?
- What is the ideal pivot?
- Naïve selection: First element in the list
- A good selection: A random element
- A better selection: Median-of-three

Median-of-three

8	10	5	1	3	20	13	7	2	12

pivot

46	15	10	6	9	15	2	5	18	12

pivot

15	48	29	18	1	19	33	23	27	41

pivot

Partitioning

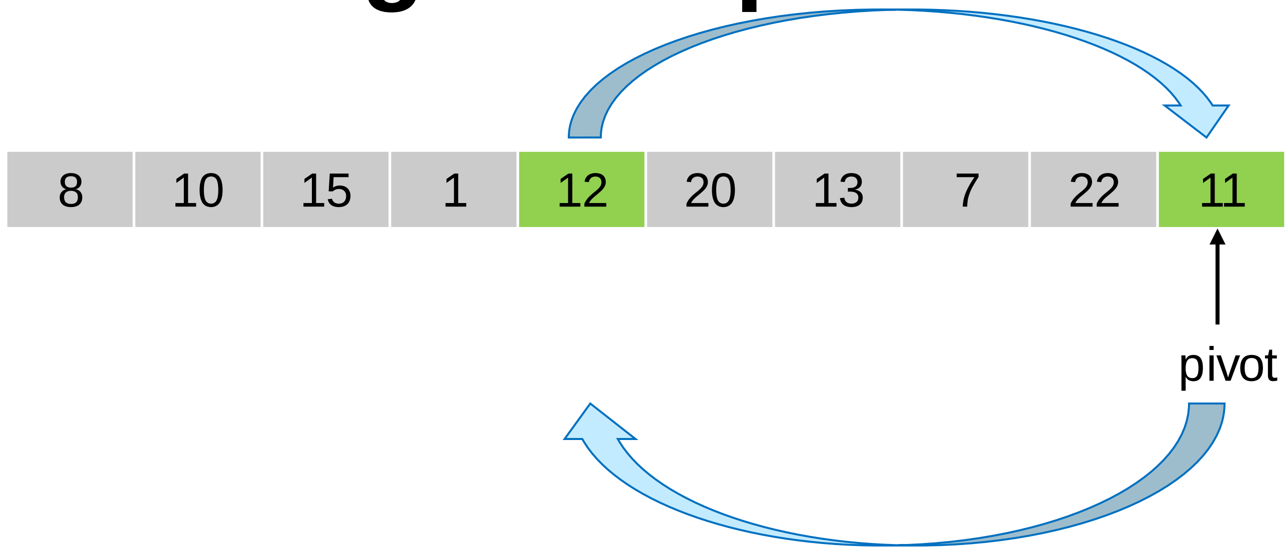
- The key idea about Quick Sort is to make an in-place partitioning
- Take the pivot out of the way
- Move bigger elements to the right
- Move smaller elements to the left
- Replace the pivot at its place

Partitioning Example

8	10	5	1	11	20	13	7	22	12

pivot

Partitioning Example



Partitioning Example

8	10	15	1	12	20	13	7	22	11
i								j	pivot

Partitioning Example

8	10	15	1	12	20	13	7	22	11

i

j

pivot

Partitioning Example

8	10	15	1	12	20	13	7	22	11
	i							j	pivot

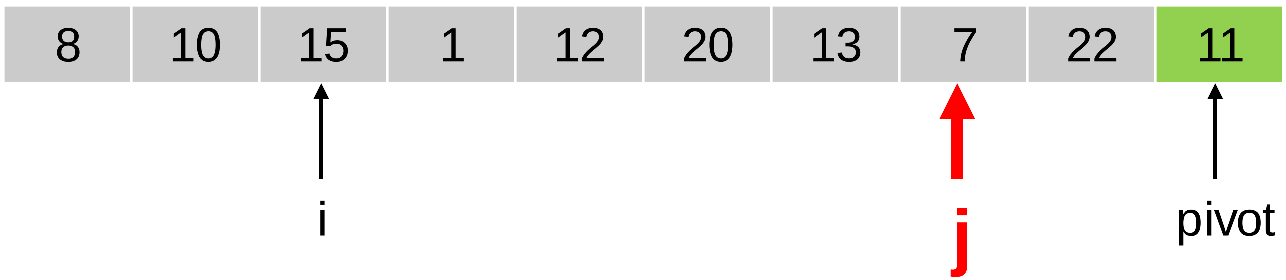
Partitioning Example

8	10	15	1	12	20	13	7	22	11
		i						j	pivot

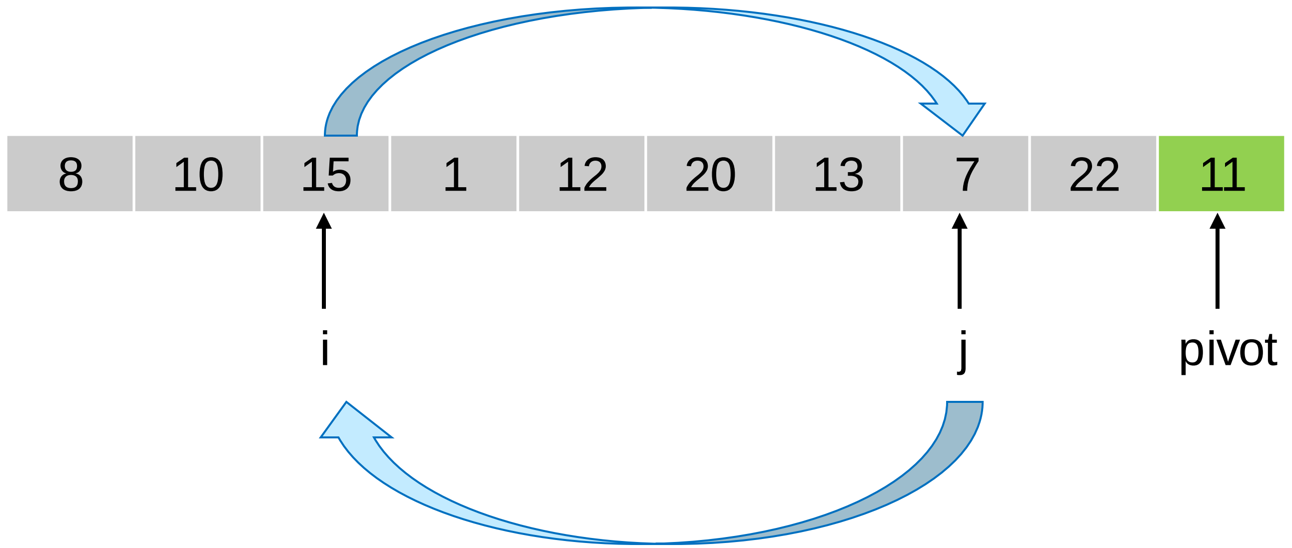
Partitioning Example

8	10	15	1	12	20	13	7	22	11
		i						j	pivot

Partitioning Example



Partitioning Example



Partitioning Example

8	10	7	1	12	20	13	15	22	11
		i					j		pivot

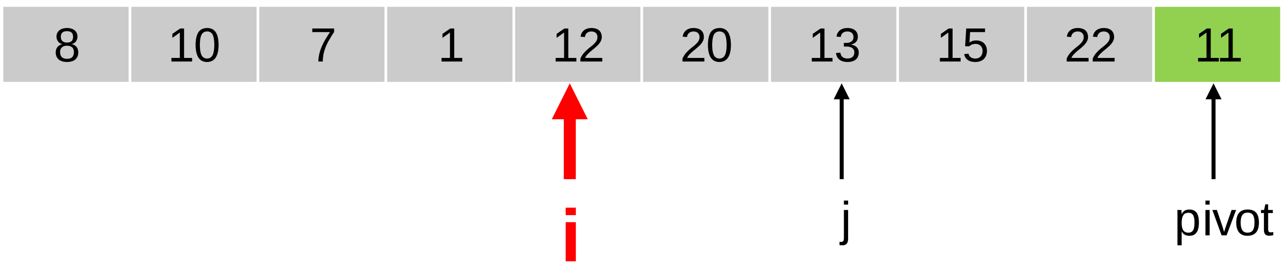
Partitioning Example

8	10	7	1	12	20	13	15	22	11
			i			j			pivot

Partitioning Example

8	10	7	1	12	20	13	15	22	11
			i			j			pivot

Partitioning Example



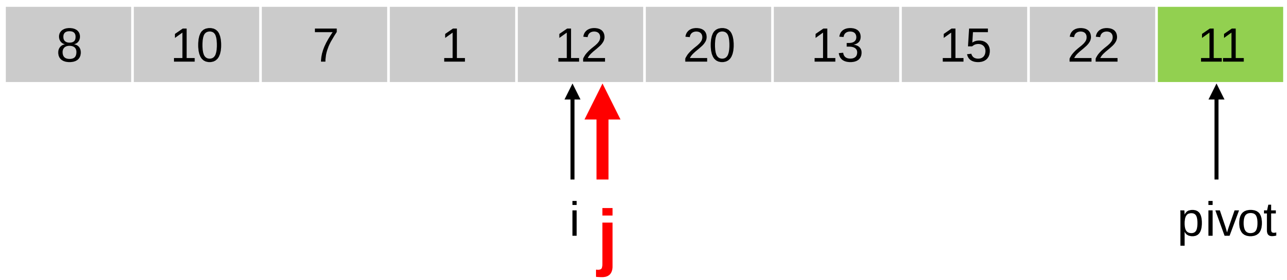
Partitioning Example

8	10	7	1	12	20	13	15	22	11
				i		**j**			pivot

Partitioning Example

8	10	7	1	12	20	13	15	22	11
				i	j				pivot

Partitioning Example

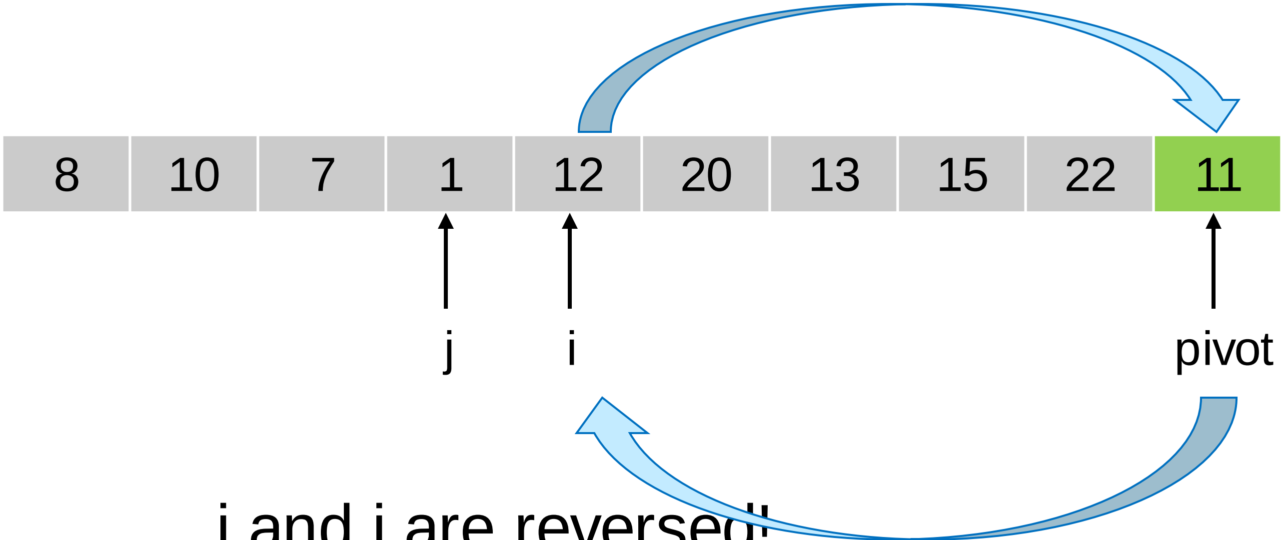


Partitioning Example

8	10	7	1	12	20	13	15	22	11

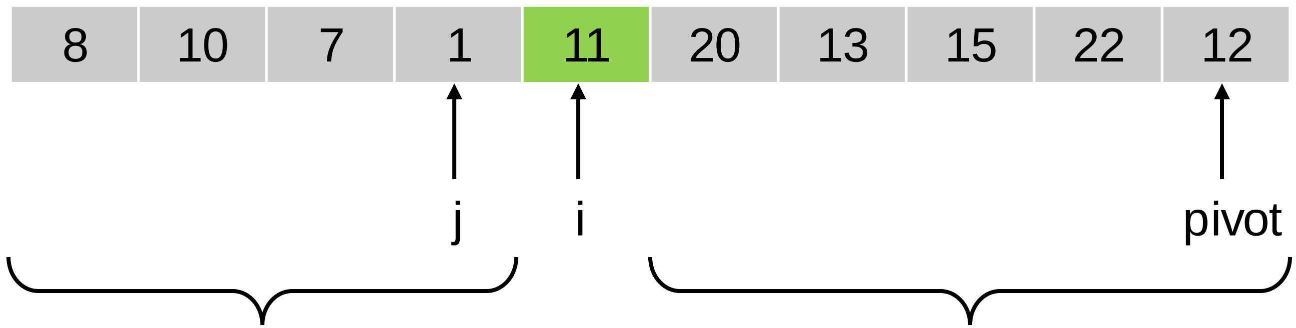
j i pivot

Partitioning Example



i and j are reversed!

Partitioning Example



Analysis of Quick Sort

- Cost of the partitioning step
  - O(n): One scan over the list
- Worst case
  - The sizes of the two sublists are 0 and n-1
  - $O(n^2)$
- Best case
  - The sizes of the two sublists are almost equal
  - O(n log n)
- Average case
  - None of the lists is excessively large or small
  - O(n log n)

Comparison of Sorting Algorithms

Algorithm	Worst-case	Best-case	Average	Stable*	In-place
Insertion					
Selection					
Bubble					
Shell					
Heap					
Merge					
Quick					

*A sorting algorithm is said to be stable if equal items remain in the same order after sorting

Comparison of Sorting Algorithms

Algorithm	Worst-case	Best-case	Average	Stable*	In-place
Insertion	$O(n^2)$	$O(n)$	$O(n^2)$	✓	✓
Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$		✓
Bubble	$O(n^2)$	$O(n)$	$O(n^2)$	✓	✓
Shell	$O(n^{3/2})$				✓
Heap	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$		✓
Merge	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	✓	
Quick	$O(n^2)$**	$O(n \log n)$	$O(n \log n)$		✓

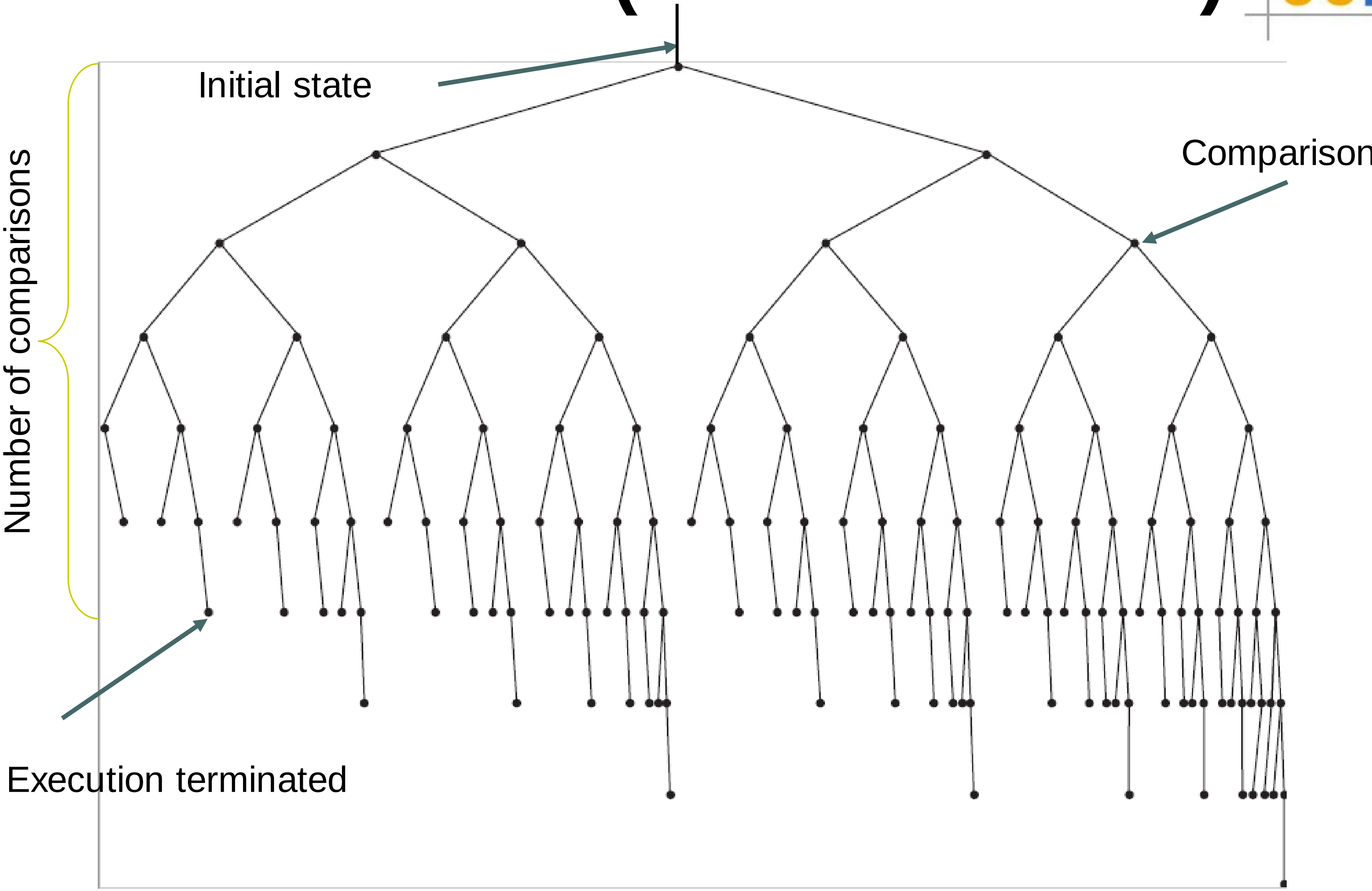
*A sorting algorithm is said to be stable if equal items remain in the same order after sorting

** Can be reduced to $O(n \log n)$ with a smart pivot selection algorithm

Lower Bound for Sorting

- Can we create a sorting algorithm with an asymptotic running time that is lower than O(n log n) in the worst case?
- Assumptions
  - Array elements can be in any order
  - Only comparisons are used to sort elements

Decision Tree (Execution Tree)



Worst-case

- Deepest part of the tree
- Number of leaf nodes
  - Equal to total number of permutations
  - n!
- Height of the tree
  - log(n!)
- $\log(n!) = \Omega(n \log n)$
- $n \log n$ is a lower bound for any comparison-based sorting algorithm