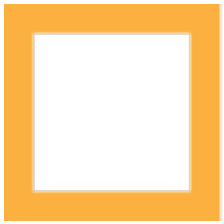
Algorithms & Data Structures
Advanced Sorting

Jacob Trier Frederiksen & Anders Kalhauge



cphbusiness

Spring 2019

Outline – Week 6

cphbusiness
COPENHAGEN BUSINESS ACADEMY

Tools & Housekeeping

- Quick Polls
- Infrastructure

Last Week: Exercise Follow-up

Sorting

- Divide and Conquor
- Merge Sort
- Quick Sort
- Stability – what is means
- Sorting – which one to choose?

Tools & Housekeeping
- Quick Polls
- Infrastructure

Last Week: Exercise Follow-up

Sorting
- Divide and Conquor
- Merge Sort
- Quick Sort
- Stability – what is means
- Sorting – which one to choose?

Quick Polls

Go to www.menti.com.

Comfort Zone: how do you feel about the course at this early point in time?

Workshop Hours: we need to find the most optimal place for face time.

To complete this course successfully, it is vital that communication and tooling is in place.

You should check that you're ready!

Are you up-to-speed on:

Java: do you have functioning Java delvelopment environment, either on your machine or in the cloud?

Moodle: can you access the course on Moodle? Receiving emails from Moodle?

Github: are you familiar with GitHub. Can you clone, pull, push, etc.?

Peergrade: can you achieve access to the grading site (after having received the site code from me? Go to www.peergrade.io/join.

Tools & Housekeeping
- Quick Polls
- Infrastructure

Last Week: Exercise Follow-up

Sorting
- Divide and Conquor
- Merge Sort
- Quick Sort
- Stability – what is means
- Sorting – which one to choose?

Create a class called `SortingAlgorithms`. This class should have an array of integer as a datafield and array size. The constructor should be used to create an array of given size.
`SortingAlgorithms` class should have three methods.

- One method for filling the array with random integers which you can call from the constructor.
- One method for implementing Insertion Sort,
- One method for Selection Sort.
- One method for an improved Three-Sum-Zero algorithm (book, `ThreeSumFast`, p.190), and time it, comparing with the `ThreeSum` for various sized arrays.

In the main method you create three objects with array sizes of 10^2, 10^3, 10^4, 10^5, and 10^6. Time your code using `Stopwatch` [1] class.

[1] see page 175 in Algorithms book

Tools & Housekeeping
- Quick Polls
- Infrastructure

Last Week: Exercise Follow-up

Sorting
- Divide and Conquor
- Merge Sort
- Quick Sort
- Stability – what is means
- Sorting – which one to choose?

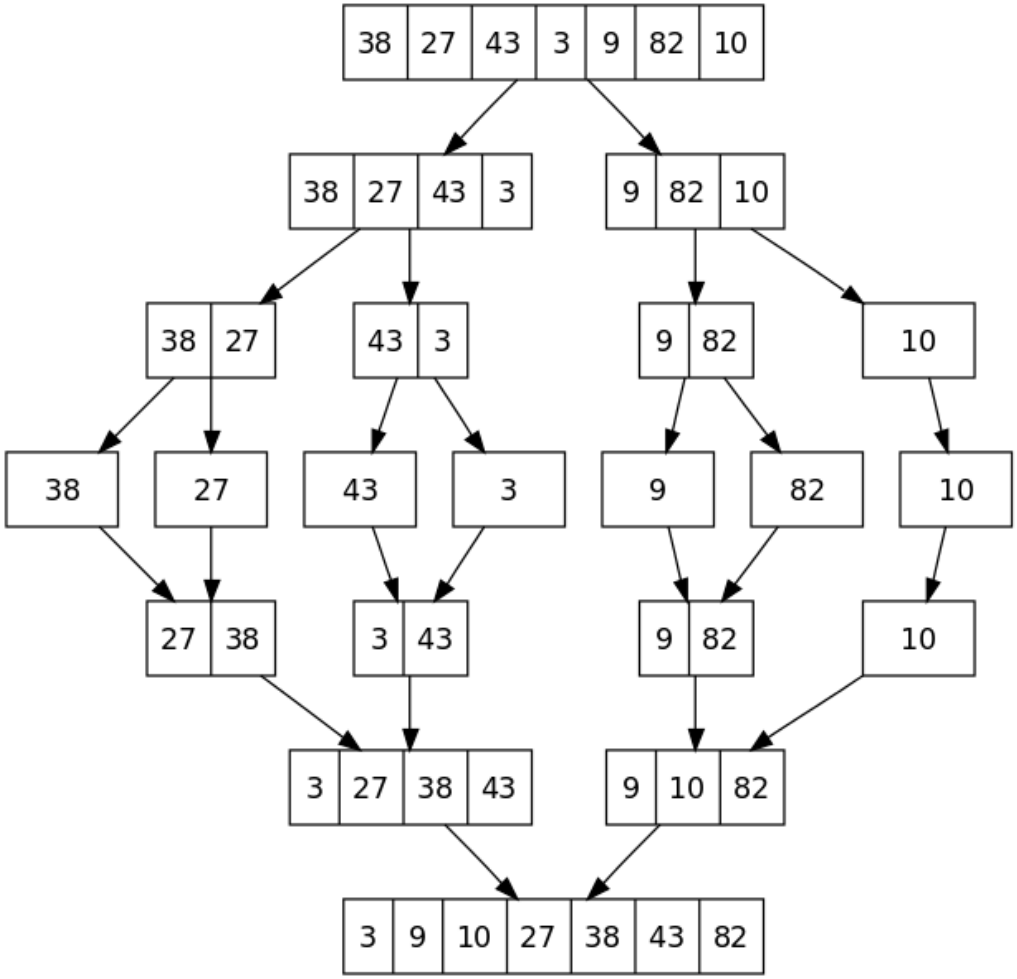
Divide and Conquor

- Divide and break
  - Break the problem in to smalle sub-problem recursively
  - Sub-problem should represent a part of the original problem
  - Keep on dividing until no more division is possible
- Conquer/Solve
  - Smalles sub-problem are solved
  - Solutions of all the sub-problems are merged
- Merge/Combine
  - Combines small solutions to the big solutions

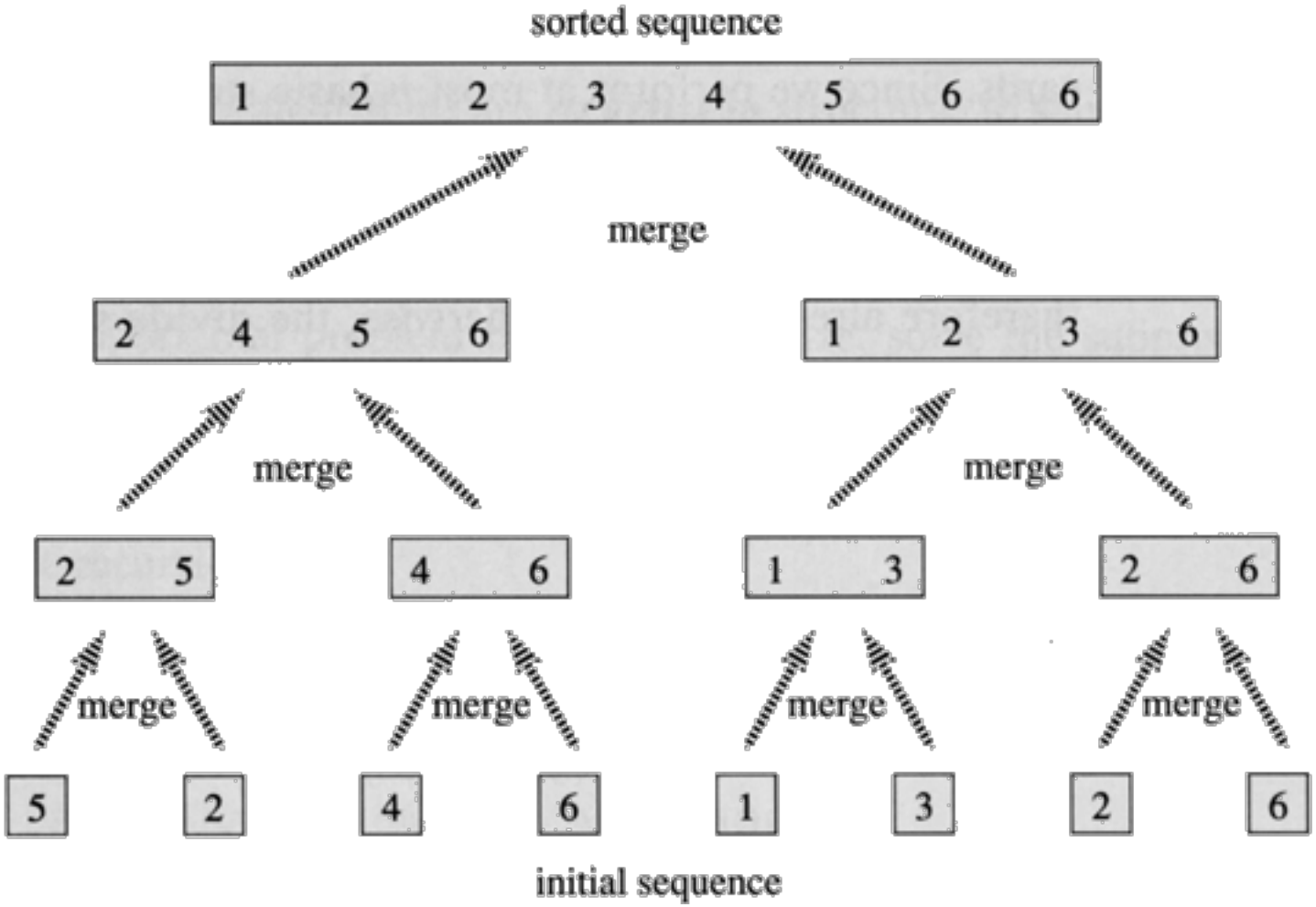
Merge Sort - top down

cphbusiness
COPENHAGEN BUSINESS ACADEMY

Divide and Conqour



Merge Sort - botton up



Merge Sort, pros and cons

Pros:
- Time: complexity is robustly $N \cdot \mathrm{Lg}(N)$.
- Stability: the algorithm is stable.

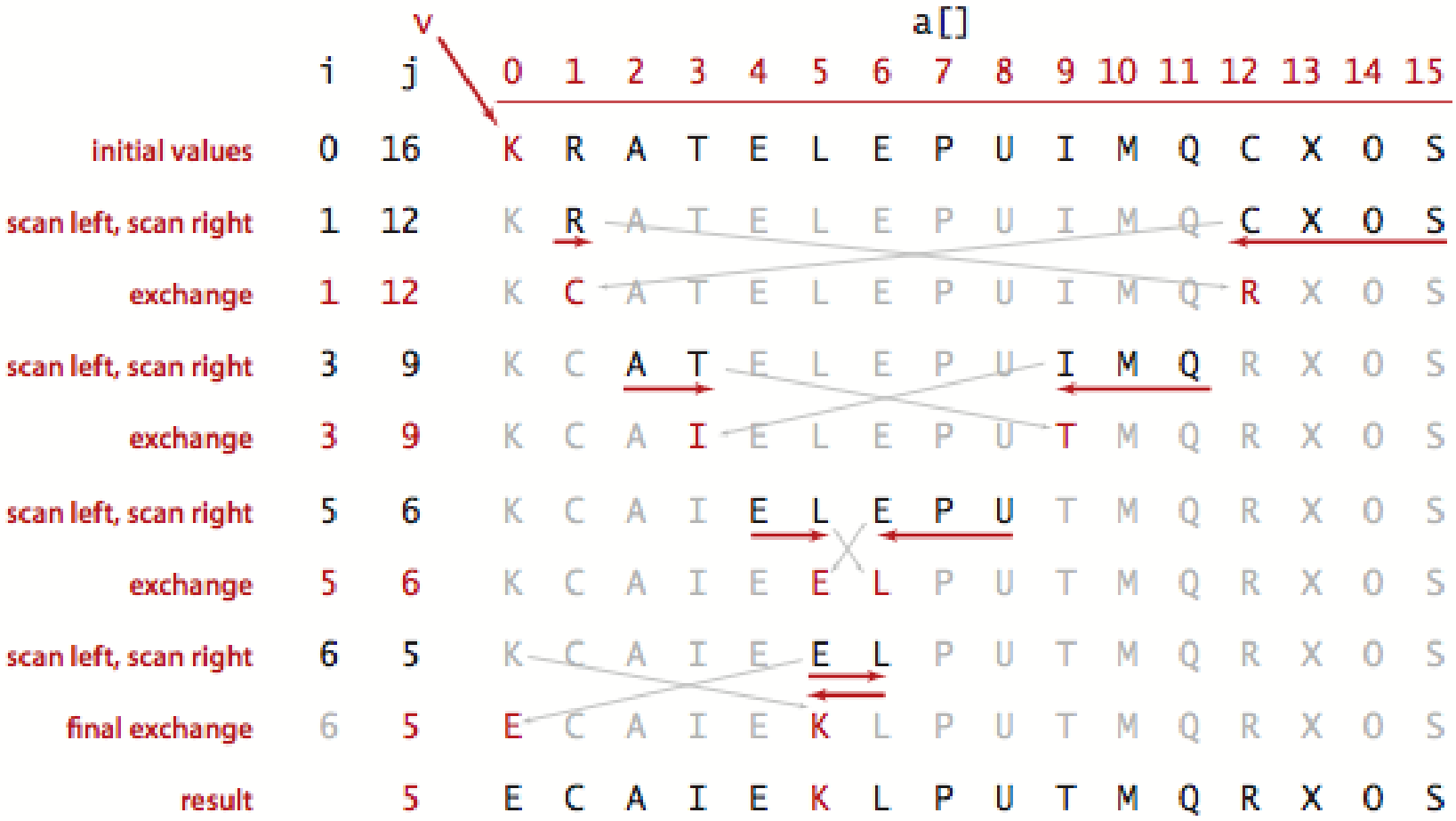
Cons:
- Space: memory overhead proportional to N.

Quick sort doesn't use auxiliary space

1. Choose a pivot p, e.g: 1^{st} element of array
2. Bring all elements less than the pivot to left end of the array and all elements greater to the other.
3. Place the pivot in between.
4. Sort items left of the pivot
5. Sort items right of the pivot

Classroom exercise (need two volunteers): on the whiteboard, sort the following arrays, using QS:

Team 1: [2, 4, 1, 3, 5],
Team 2: [1, 2, 3, 4, 5].



Partitioning trace (array contents before and after each exchange)

Quick Sort, pros and cons

cphbusiness
COPENHAGEN BUSINESS ACADEMY

Pros:
- Space: no (significant) overhead: $\sim lg(N)$
- Time: complexity is $\sim N \cdot lg(N)$ (best-to-avg case).

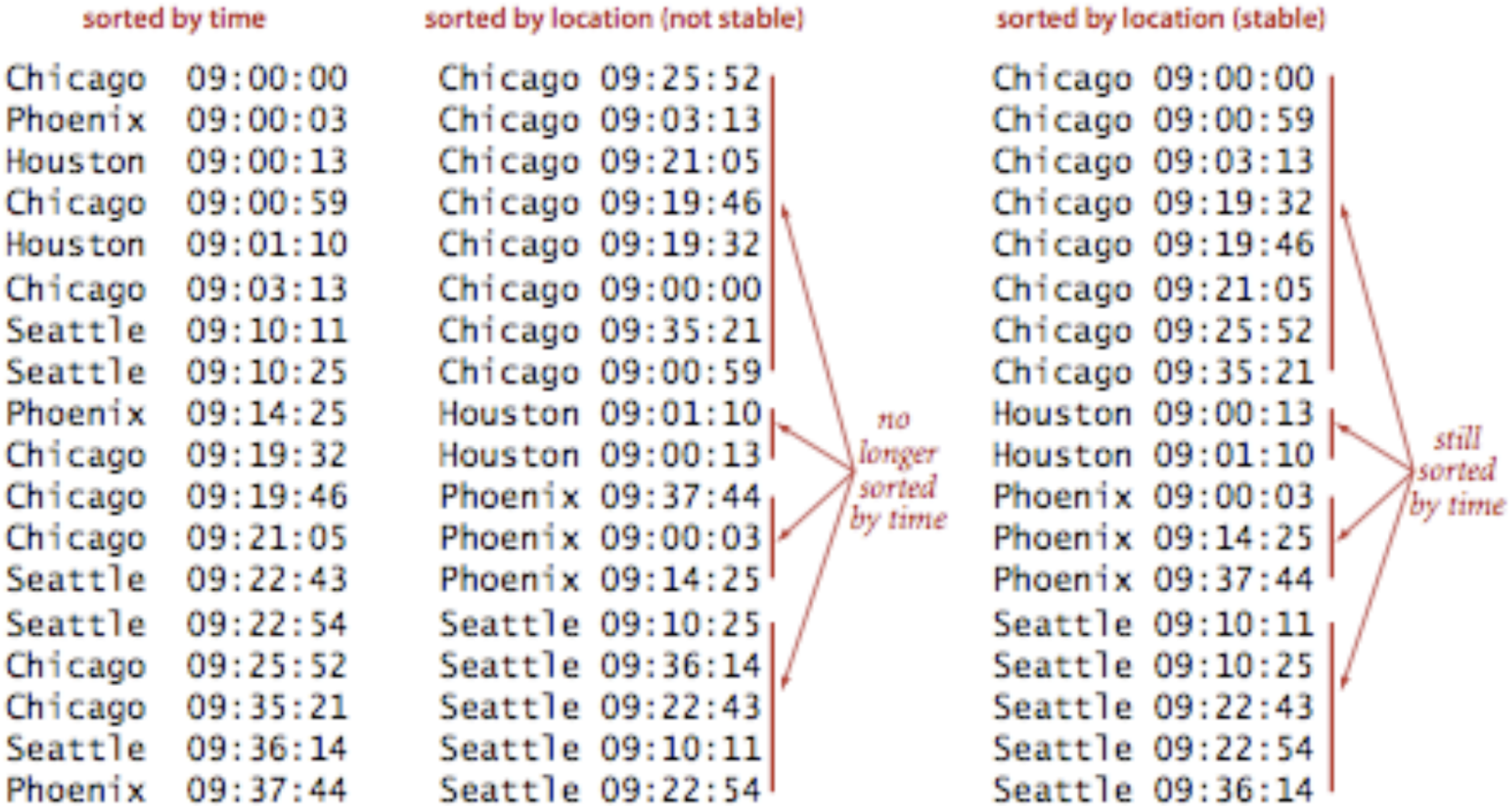
Cons:
- Time: complexity is $\sim N^2$ (worst case).
- Stability: the algorithm is not stable.

Q: how to circumvent the N^2 worst case of Quick Sort?

So, what does it mean, Stability?

- Team 1: Sort half a deck of cards of spades ♠ (ranks 2 through 7) and diamonds ♢ (ranks 2 through 7), first by rank, then by suit. NB: Random shuffle cards well to begin with. Use QUICK SORT.
- Team 2: Do the same on the half-deck of hearts ♡ and clubs ♣, instead. NB: Random shuffle cards well to begin with. Use MERGE SORT.

Stability, example



Stability when sorting on a second key

Choosing a Sort?

Which sorting algorithm should I pick?

algorithm	stable?	in place?	*order of growth to sort N items* running time	extra space	notes
selection sort	no	yes	N^2	1	
insertion sort	yes	yes	between N and N^2	1	depends on order of items
shellsort	no	yes	$N \log N$? $N^{6/5}$?	1	
quicksort	no	yes	$N \log N$	$\lg N$	probabilistic guarantee
3-way quicksort	no	yes	between N and $N \log N$	$\lg N$	probabilistic, also depends on distribution of input keys
mergesort	yes	no	$N \log N$	N	
heapsort	no	yes	$N \log N$	1	

Performance characteristics of sorting algorithms

Exercise 3

Re-design your `SortingAlgorithms` class. This time do the following:

1. Implement a Bottom-Up method for Merge Sort
2. Implement one method for Quick Sort

For these methods, check running time (optionally max memory consumption) for three cases:

1. Randomized integer arrays of sizes 10^1, 10^3, 10^5, 10^7
2. Forward sorted integer arrays of sizes 10^1, 10^3, 10^5, 10^7
3. Reverse sorted integer arrays of sizes 10^1, 10^3, 10^5, 10^7

For the sake of simplicity you may assume that the integer arrays have only unique values (no multiplets).