COMP20003 Workshop Week 7

1	Sorting, Insertion Sort and Selection Sort revisited
2	have some fun …
3	Quicksort + Q 6.1
4	Review for the Test / sample test
5	Implementing Hash Table (P6.1)

Sorting: algorithms, stable sort, in-place sort…

	Selection	Insertion	Quick	Merge (top-down)
Basic Idea	Identify the smallest and swap it with the first…	From 2nd element, insert it to the left sub-array so that the extended left sub-array remains sorted.	Choose a pivot, partition array into a *lesser* and a *greater* (than pivot) halves…	Split to equal-size halves, sort them then merge them.
Best case				
Worst case				
Average				
In-place?				
Stable?				

Sorting by distribution counting – very special!

Condition: `m <= A[i] <= n` for all `i`, and `r=n-m+1` is not too large.

Special aspect: unlike all other sorting algorithms, we do not compare keys.

How? big-O = ? Example for n=100, m=0, n= 10.

Quiz 1

The best case of Selection sort is:

a. O(n log n).

b. O(n).

c. $O(n^2)$.

d. O(log n).

When?

Quiz 2

The best case of Insertion Sort is:

a. O(n log n).

b. O(n).

c. $O(n^2)$.

d. O(log n).

When?

Quiz 3

The average case of Insertion Sort is:

a. O(n log n).

b. O(n).

c. $O(n^2)$.

d. O(log n).

Quiz 4

The average case of Quick Sort is:

a. O(n log n).

b. O(n).

c. $O(n^2)$.

d. O(log n).

Quiz 5

The worst case of Quick Sort is:

a. O(n log n).

b. O(n).

c. $O(n^2)$.

d. O(log n).

When?

Quiz 6

Suppose that A[] is a sorted array of n elements. Which of the following are correct:

a. We can write a O(n) search algorithm.

b. We can write a O(log n) search algorithm.

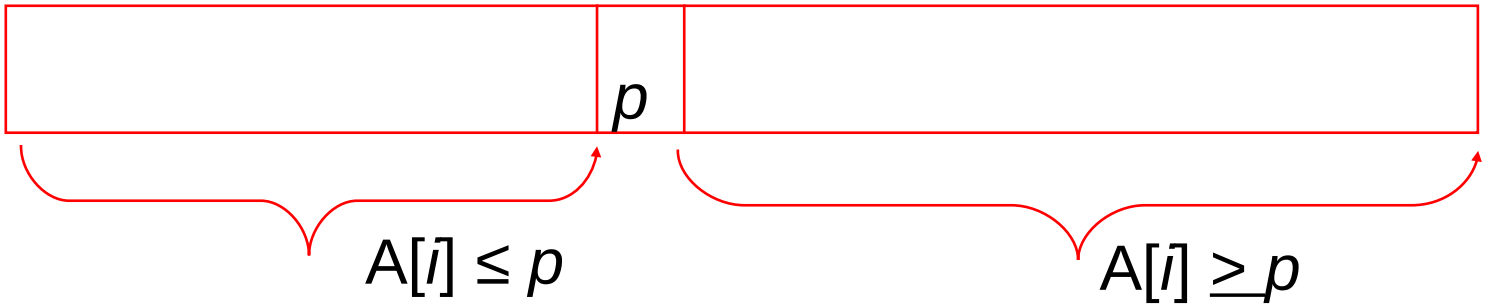
c. We can write a O(1) search algorithm.

d. All search algorithms on A[] can be used even if A[] is an unsorted array.

Partitioning (using the rightmost element as pivot)

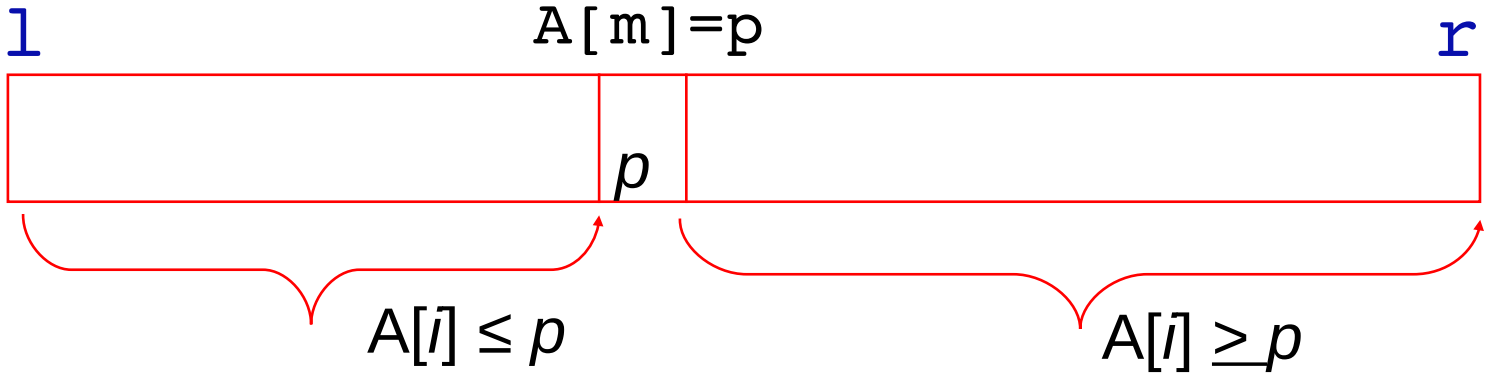
The task:

- Input: Given an unsorted slice `A[l..r]`,
  and supposing that `p= a[r]`.
- Output: A value `m` and re-arrangement of `A` so that:

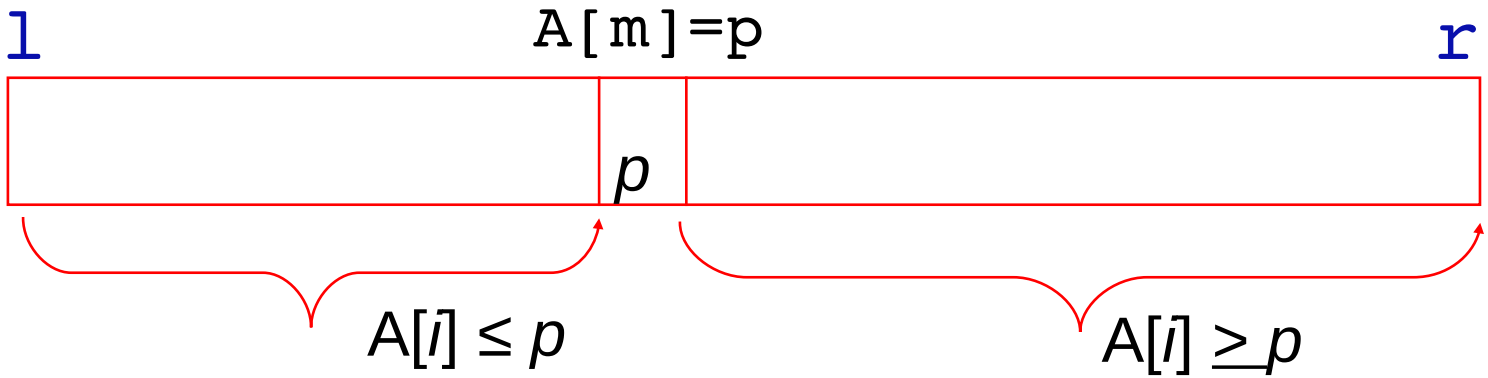


`A[m]=p`

Partitioning (using rightmost element as pivot)



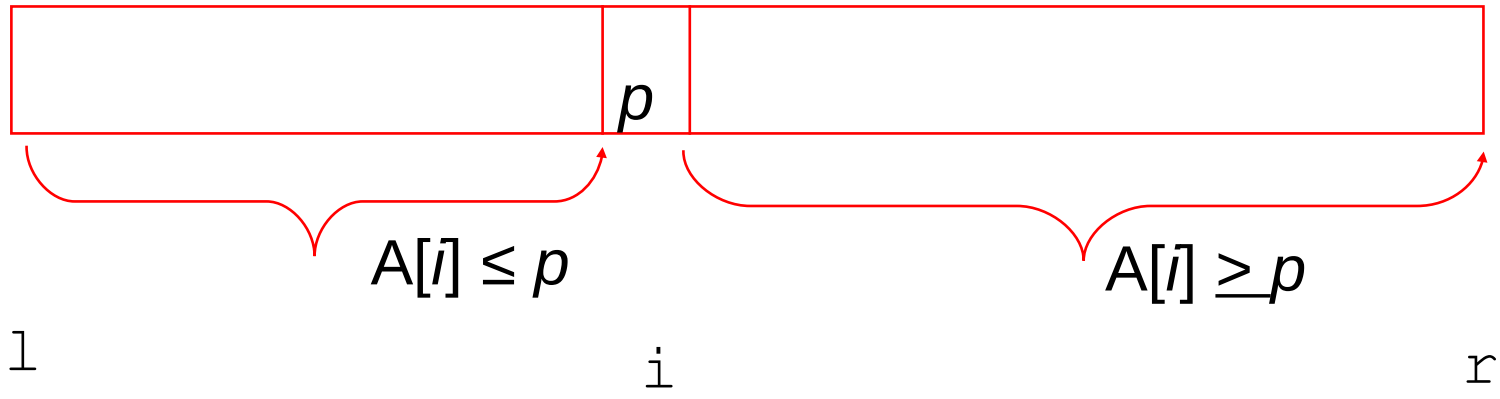
Partitioning variations



- Using element at a random position as pivot
- Using median-of-three as pivot
- Using the leftmost element as pivot
- …

Quick Sort (concept + algorithm)

```
void quicksort(item A[], int l, int r)
{
      int i;
      if (r <= l) return;
      i = partition(A,l,r);
      quicksort(A,l,i-1);
      quicksort(A,i+1,r);
}
```



Q 6.1

You are asked to show the operation of quicksort on the following keys. For simplicity, use the rightmost element as the partition element:

2 3 97 23 15 21 4 23 29 37 5 23

Comment on the stability of quicksort and its behavior on almost sorted inputs.

week 1-6 in a nutshell (*might be incomplete!*)

W	Lectures	Followed Workshops (Week W+1)
1	• Introduction & Efficiency • Fibonacci: recursive & iterative	• C review, prog environment, `Makefile`
2	• Big-O & Big-Θ : concepts & definition • Grow order of popular functions • Best-worst-average case analysis	• Complexity • Arrays: static and dynamic allocation • Multi-file C projects and `Makefile` • L: pointer parameters, dynamic arrays
3	• ADT • Arrays: selection sort, search • Linked lists: arrays vs linked lists	• C useful tools • Tree traversal, BST insert, `gdb`/`valgrind` • AVL and rotations • L: impl BST Insert
4	• Tree & BST, operations on BST • AVL & Rotations	• Stacks & Queue: linked lists & array impl • L: Assignment 1 and/or free mem used by a stack
5	• Distribution Counting (alg + complexity) • Hashing: collisions, chaining, linear probing, double hashing	• Hashing, dealing with collisions • L: Assigment 1 and/or impl simple hash table
6	• Sorting, insertion sort, qsort	• sorting algorithms, qsort, distribution counting • L: impl simple hash table

Sorting

Concepts: sorting, stable sort, in-place sort
Some sorting algorithm (on an array of n elements):

	Selection	Insertion	Quick	Merge
Basic Idea	Identify the smallest and swap it with the first…	From 2nd element, insert it to the left sub-array so that the extended left sub-array remains sorted.	Choose a pivot, partition array into a *lesser* and a *greater* (than pivot) halves…	Split to equal-size halves, sort them then merge them.
Best case	$O(n^2)$	$O(n)$	$O(n \log n)$	$O(n \log n)$
Worst case	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n \log n)$
Average	$O(n^2)$	$O(n^2)$	$O(n \log n)$	$O(n \log n)$
In-place?	✔	✔	✔	✖
Stable?	✖	✔	✖	✔

P6.1 Implementing a simple hash table, OR group work:

Choices:

- Implementing a simple hash table
- Group work with sample MST questions and/or programming tasks, for example:
  - Big-O questions
  - Hashing examples
  - AVL rotations
  - Programming: dynamic arrays and strings
  - Programming: linked lists