Algorithms and Data Structures

- An algorithm is a step-by-step procedure (a "recipe") for performing a task.
- A data structure is a systematic way of organising data and making it accessible in certain ways

This thread of the course is concerned with the design and analysis of "good" algorithms and data structures.

Algorithms and Data Structures in CS1

Data Structures

Arrays, linked lists, stacks, trees

Algorithm design principles

Recursive algorithms, dynamic programming

Sorting Algorithms

Insertion sort, selection sort, bucket sort

Textbooks

[CLRS] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. McGraw-Hill, 2002.

[GT] Michael T. Goodrich and Roberto Tamassia. *Algorithm Design – Foundations, Analysis, and Internet Examples*. Wiley, 2002.

[W] Mark A. Weiss. *Data Structures & Algorithm Analysis in JAVA*. Addison-Wesley, 1999.

Criteria for evaluating algorithms

- Correctness
- Efficiency w.r.t.
  - running time
  - space (= amount of memory used)
  - network traffic
  - number of times secondary storage is accessed
- Simplicity

Measuring the Running Time

The running time of a program depends on a number of factors such as:

1. The running time of the algorithm.

2. The input of the program.

3. The quality of the implementation and the quality of the code generated by the compiler.

4. The machine used to execute the program.

Example 1: Sequential Search in JAVA

```
public static int seqSearch(int[] A,int k) {
    for(int i = 0; i < A.length; i++)
        if ( A[i] == k )
            return i;
    return -1;
}
```

Sequential Search in pseudo-code

Algorithm seqSearch(A, k)

Input: An integer array A, an integer k

Output: The smallest index i with $A[i] = k$, if such an i exists, or -1 otherwise.

```
1.  for i ← 0 to A.length − 1 do
2.      if A[i] = k then
3.          return i
4.  return −1
```

Worst Case Running Time

Assign a size to each possible input.

Definition: The (worst-case) running time of an algorithm A is the function $T_{\mathsf{A}} : \mathbb{N} \to \mathbb{N}$ where $T_{\mathsf{A}}(n)$ is the maximum number of computation steps performed by A on an input of size n.

Average Running Time

Worst-case running time seems overly pessimistic!

Definition (?)

The average running time of an algorithm A is the function $AVT_{\mathsf{A}} : \mathbb{N} \to \mathbb{N}$ where $AVT_{\mathsf{A}}(n)$ is the average number of computation steps performed by A on an input of size n.

Problems with average time

- What precisely does average mean? What an "average" input is depends on the application.
- Average time analysis is mathematically very difficult and often infeasible.

A reasonable approach

Worst-Case Analysis + Experiments

Example 2: Binary Search

Algorithm binarySearch(A, k, i_1, i_2)

Input: An integer array A in increasing order, integers i_1, i_2 and k

Output: An index $i, i_1 \leq i \leq i_2$ with $A[i] = k$, if such an i exists, or -1 otherwise.

```
1.  if i_2 < i_1 then
2.      return -1
3.  else
4.      j ← ⌊(i_1 + i_2)/2⌋
5.      if k = A[j] then
6.          return j
7.      else if k < A[j] then
8.          return binarySearch(A, k, i_1, j − 1)
9.      else
10.         return binarySearch(A, k, j + 1, i_2)
```

JAVA code for binary search

```
public static int binarySearch(int[] A,int k,int i1,int i2) {
    if ( i1 > i2 )
        return -1;
    int j = i1+i2/2;
    if ( A[j] == k )
        return j;
    else if ( A[j] < k )
        return binarySearch(A,k,i1,j-1);
    else
        return binarySearch(A,k,j+1,i2);
}
```

A code fragment for measuring the running time

```
Random rand = new Random(System.currentTimeMillis());

int[] A = new int[size];
for( int j =0; j < size; j++)
    A[j] = rand.nextInt(size);

int k = rand.nextInt(size);

long start = System.currentTimeMillis();
seqSearch(A,k);
long end = System.currentTimeMillis();
int t = (int) (end - start);
```

The actual running time

input size	seqSearch		binarySearch	
	wc	avc	wc	avc
100	4 ms	<= 1 ms	<= 1 ms	<= 1 ms
1000	<= 1 ms	<= 1 ms	<= 1 ms	<= 1 ms
10000	<= 1 ms	<= 1 ms	<= 1 ms	<= 1 ms
100000	3 ms	1.5 ms	<= 1 ms	<= 1 ms
200000	5 ms	3.2 ms	<= 1 ms	<= 1 ms
300000	7 ms	4.6 ms	<= 1 ms	<= 1 ms
400000	11 ms	5.9 ms	<= 1 ms	<= 1 ms
500000	12 ms	7.5 ms	<= 1 ms	<= 1 ms
600000	14 ms	8.6 ms	<= 1 ms	<= 1 ms
700000	17 ms	9.5 ms	<= 1 ms	<= 1 ms
800000	20 ms	11.9 ms	<= 1 ms	<= 1 ms
900000	22 ms	13.5 ms	<= 1 ms	<= 1 ms
1000000	25 ms	15.6 ms	<= 1 ms	<= 1 ms