DSK814: Algorithms and Data Structures

SDU - Kolding

Algorithm Analysis

Algorithm analysis

Minimum requirements for algorithms are correctness:

- Stops for all inputs (never infinite loop).
- Provides the sought output when it stops (i.e. provides an answer to our problem).

Algorithm analysis

Minimum requirements for algorithms are correctness:

- Stops for all inputs (never an infinite loop).
- Provides the sought output when it stops (i.e. provides an answer to our problem).

Correct algorithms can have different qualities:

- Speed
- Space usage
- Complexity of implementation
- Additional properties (problem-specific), such as stability of sorting.

Algorithm analysis

Minimum requirements for algorithms are correctness:

- Stops for all inputs (never an infinite loop).
- Provides the sought output when it stops (i.e. provides an answer to our problem).

Correct algorithms can have different qualities:

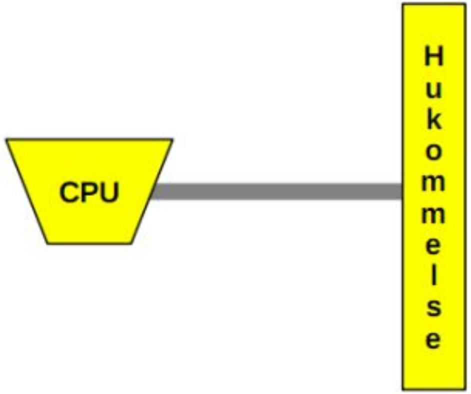
- Speed
- Space usage
- Complexity of implementation
- Additional properties (problem-specific), such as stability of sorting.

In this course, we will describe existing algorithms, develop new algorithms, and analyze algorithms for correctness and quality.

We need for Algorithm Analysis:

- Model of problem. Individual for each problem.
- Model of machine. We are using the RAM model.
- Quality goals. We focus (mostly) on time consumption.
- Mathematical analysis tools: Loop invariants, induction, recursion equations, asymptotic notation.

The RAM model



- A CPU
- A memory (~infinite array of cells).
- A number of basic operations:add,sub,shift,compare,move data item,jump into program(loop,branching,method call). These are all assumed to take the same amount of time.
- Time for an algorithm: number of basic operations performed.
- Space for an algorithm: maximum number of occupied memory cells.

Measure time spent

For a given size n of input there are often many different specific inputs.

Example: for sorting of n=8 numbers, the following are some of the possible inputs:

7,2,3,1,8,5,4,6

1,8,2,7,3,6,4,5

1,2,3,4,5,6,7,8

8,7,6,5,4,3,2,1

Measure time spent

For a given size n of input there are often many different specific inputs.

Example: for sorting of n=8 numbers, the following are some of the possible inputs:

7,2,3,1,8,5,4,6

1,8,2,7,3,6,4,5

1,2,3,4,5,6,7,8

8,7,6,5,4,3,2,1

An algorithm usually has different time consumption for each of these.

Measure time spent

For a given size n of input there are often many different specific inputs.

Example: for sorting of n=8 numbers, the following are some of the possible inputs:

7,2,3,1,8,5,4,6

1,8,2,7,3,6,4,5

1,2,3,4,5,6,7,8

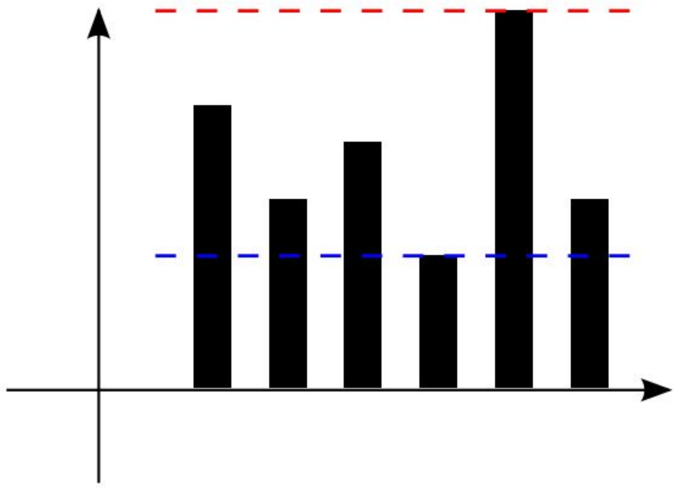
8,7,6,5,4,3,2,1

An algorithm usually has different time consumption for each of these.

What inputs should we use to evaluate the time consumption?

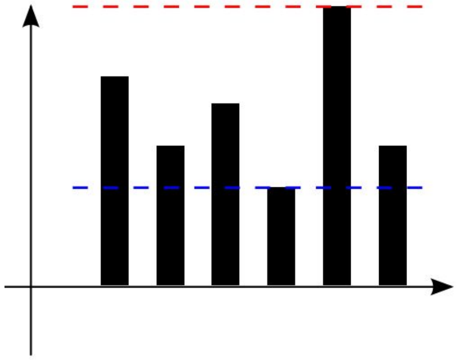
Measure time spent

- Worst case(max over all inputs of size n)
- Average case (average over a distribution of inputs of size n)
- Best case(min over all inputs of size n)



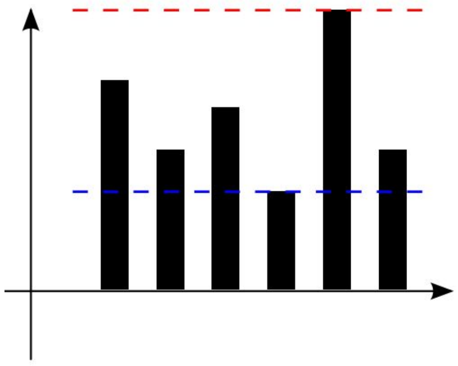
Running time for the different input sizes n

Worst case time consumption



Worst case gives a guarantee. It is also often representative of the average case (but can also be significantly more pessimistic)

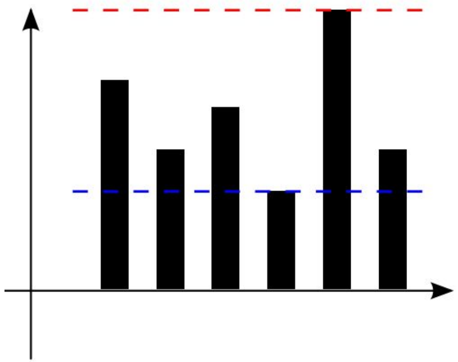
Worst case time consumption



Worst case gives a guarantee. It is also often representative of the average case (but can also be significantly more pessimistic)

Average case: What is the distribution of input? Why is this distribution realistic/ relevant? Analysis is often (mathematically) difficult to carry out.

Worst case time consumption

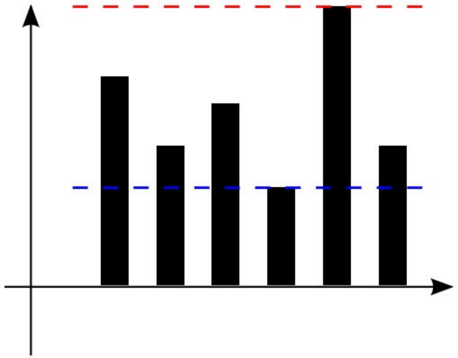


Worst case gives a guarantee. It is also often representative of the average case (but can also be significantly more pessimistic)

Average case: What is the distribution of input? Why is this distribution realistic/relevant? Analysis is often (mathematically) difficult to carry out.

Best case: Often does not provide much relevant information.

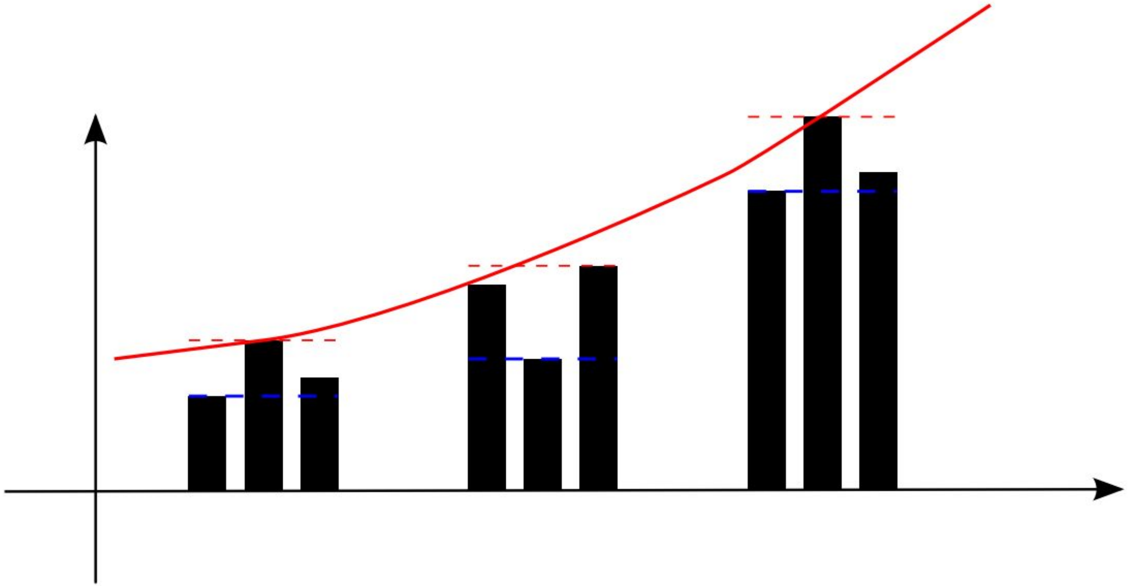
Worst case time consumption



Almost all analyses in this course are worst case.

Different input sizes

Worst-case running time is usually an increasing function of the input size n:

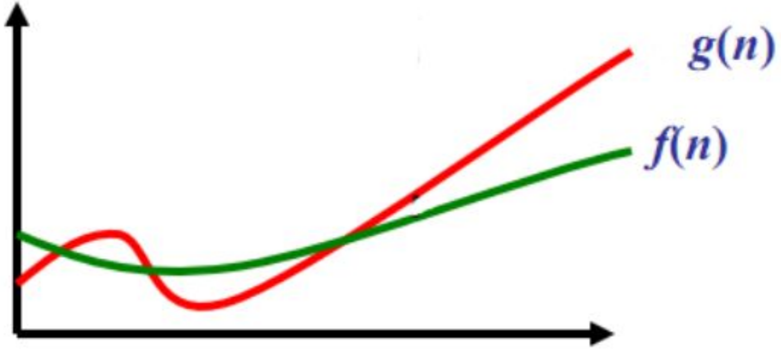


Running time for the various inputs of increasing size n

Growth rate

An analysis must therefore provide a function f(n) of the input size n.

We therefore need to compare functions. The relevant target is the growth rate of f(n) when n increases - a faster growing function will always overtake a slower growing function when n becomes big enough. And too small n(almost) all algorithms are fast.



Growth rate

Examples (increasing growth rate):

$$1, \quad \log n, \quad \sqrt{n}, \quad n, \quad n \log n,$$

$$n\sqrt{n}, \quad n^2, \quad n^3, \quad n^{10}, \quad 2^n$$

Later: definition of asymptotic growth rate for functions and comparisons of them.