4 Modelling Issues

What do you measure?

- ► Memory requirement
- ► Running time
- ► Number of comparisons
- ► Number of multiplications
- ► Number of hard-disc accesses
- ► Program size
- ► Power consumption
- ► . . .

4 Modelling Issues

How do you measure?

- ► Implementing and testing on representative inputs
  - ► How do you choose your inputs?
  - ► May be very time-consuming.
  - ► Very reliable results if done correctly.
  - ► Results only hold for a specific machine and for a specific set of inputs.

- ► Theoretical analysis in a specific model of computation.
  - ► Gives asymptotic bounds like "this algorithm always runs in time $\mathcal{O}(n^2)$".
  - ► Typically focuses on the worst case.
  - ► Can give lower bounds like "any comparison-based sorting algorithm needs at least $\Omega(n \log n)$ comparisons in the worst case".

4 Modelling Issues

Input length

The theoretical bounds are usually given by a function $f : \mathbb{N} \to \mathbb{N}$ that maps the input length to the running time (or storage space, comparisons, multiplications, program size etc.).

The input length may e.g. be

- ► the size of the input (number of bits)
- ► the number of arguments

Example 1

Suppose n numbers from the interval $\{1, \dots, N\}$ have to be sorted. In this case we usually say that the input length is n instead of e.g. $n \log N$, which would be the number of bits required to encode the input.

Model of Computation

How to measure performance

1. Calculate running time and storage space etc. on a simplified, idealized model of computation, e.g. Random Access Machine (RAM), Turing Machine (TM), . . .

2. Calculate number of certain basic operations: comparisons, multiplications, harddisc accesses, . . .

Version 2. is often easier, but focusing on one type of operation makes it more difficult to obtain meaningful results.

Turing Machine

- ▶ Very simple model of computation.
- ▶ Only the "current" memory location can be altered.
- ▶ Very good model for discussing computabiliy, or polynomial vs. exponential time.
- ▶ Some simple problems like recognizing whether input is of the form xx, where x is a string, have quadratic lower bound.

⟹ **Not a good model for developing efficient algorithms.**

control unit

state

state holds program and can act as constant size memory
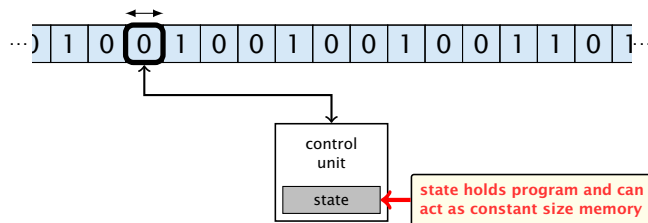
Random Access Machine (RAM)

- ▶ Input tape and output tape (sequences of zeros and ones; unbounded length).
- ▶ Memory unit: infinite but countable number of registers $R[0], R[1], R[2], \ldots$.
- ▶ Registers hold integers.
- ▶ Indirect addressing.

input tape

memory
$R[0]$
$R[1]$
$R[2]$
$R[3]$
$R[4]$
$R[5]$

control unit
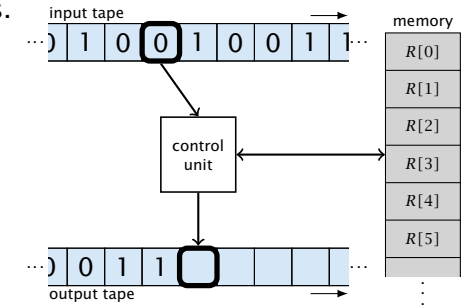
output tape

Note that in the picture on the right the tapes are one-directional, and that a READ- or WRITE-operation always advances its tape.

Random Access Machine (RAM)

Operations

- ▶ input operations (input tape → $R[i]$)
  - ▶ READ i
- ▶ output operations ($R[i]$ → output tape)
  - ▶ WRITE i
- ▶ register-register transfers
  - ▶ $R[j] := R[i]$
  - ▶ $R[j] := 4$
- ▶ **indirect** addressing
  - ▶ $R[j] := R[R[i]]$
    loads the content of the $R[i]$-th into the j-th register
  - ▶ $R[R[i]] := R[j]$
    loads the content of the j-th into the $R[i]$-th register

Random Access Machine (RAM)

Operations

- ▶ branching (including loops) based on comparisons
  - ▶ jump x
    jumps to position x in the program;
    sets instruction counter to x;
    reads the next operation to perform from register $R[x]$
  - ▶ jumpz x $R[i]$
    jump to x if $R[i] = 0$
    if not the instruction counter is increased by 1;
  - ▶ jumpi i
    jump to $R[i]$ (indirect jump);
- ▶ arithmetic instructions: $+, -, \times, /$
  - ▶ $R[i] := R[j] + R[k]$;
    $R[i] := -R[k]$;

The jump-directives are very close to the jump-instructions contained in the assembler language of real machines.

Model of Computation

- ▶ uniform cost model
  Every operation takes time 1.

- ▶ logarithmic cost model
  The cost depends on the content of memory cells:
  - ▶ The time for a step is equal to the largest operand involved;
  - ▶ The storage space of a register is equal to the length (in bits) of the largest value ever stored in it.

Bounded word RAM model: cost is uniform but the largest value stored in a register may not exceed w, where usually $w = \log_2 n$.

> The latter model is quite realistic as the word-size of a standard computer that handles a problem of size n must be at least $\log_2 n$ as otherwise the computer could either not store the problem instance or not address all its memory.

4 Modelling Issues

Example 2

Algorithm 1 RepeatedSquaring(n)

1: $r \leftarrow 2$;
2: **for** $i = 1 \rightarrow n$ **do**
3:      $r \leftarrow r^2$
4: **return** r

- ▶ running time:
  - ▶ uniform model: n steps
  - ▶ logarithmic model: $1 + 2 + 4 + \cdots + 2^n = 2^{n+1} - 1 = \Theta(2^n)$
- ▶ space requirement:
  - ▶ uniform model: $\mathcal{O}(1)$
  - ▶ logarithmic model: $\mathcal{O}(2^n)$

There are different types of complexity bounds:

- ▶ best-case complexity:
$$C_{\mathrm{bc}}(n) := \min\{C(x) \mid |x| = n\}$$
  Usually easy to analyze, but not very meaningful.

- ▶ worst-case complexity:
$$C_{\mathrm{wc}}(n) := \max\{C(x) \mid |x| = n\}$$
  Usually moderately easy to analyze; sometimes too pessimistic.

- ▶ average case complexity:
$$C_{\mathrm{avg}}(n) := \frac{1}{|I_n|} \sum_{|x|=n} C(x)$$

  more general: probability measure μ
$$C_{\mathrm{avg}}(n) := \sum_{x \in I_n} \mu(x) \cdot C(x)$$

$C(x)$	cost of instance x
$\|x\|$	input length of instance x
I_n	set of instances of length n

There are different types of complexity bounds:

- ▶ amortized complexity:
  The average cost of data structure operations over a worst case sequence of operations.

- ▶ randomized complexity:
  The algorithm may use random bits. Expected running time (over all possible choices of random bits) for a fixed input x. Then take the worst-case over all x with $|x| = n$.