Algorithms and Data Structures

Module 1

Lecture 1

Introduction to algorithmic complexity

Adigeev Mikhail Georgievich

mgadigeev@sfedu.ru

adimg@yandex.ru

Problems and algorithms

- What is an 'algorithm'?
- Algorithms solve problems.
- Unsolvable problems.
- Classes and instances of problems.
- Tractable vs intractable problems.

Algorithmic/Computational complexity

Informal definition:

***Complexity** of an algorithm is the amount of resources the algorithm needs to successfully solve the problem*.

Resource types:

- Time
- Space
- …

Algorithmic/Computational complexity

- Let *x* be an instance of a problem.
- *T*(*x*) = time spent by the algorithm to solve instance *x*.
- T(x) depends on the *size* (*length*) of *x*. Size of x = |x| = *n*.

Column addition:

x=(a,b)

~~T(x) = min{|a|,|b|}+1~~

T(x) = max{|a|,|b|}+1

			1		1	
	1	2	5	6	3	7
+			1	5	3	4
	1	2	7	1	7	1

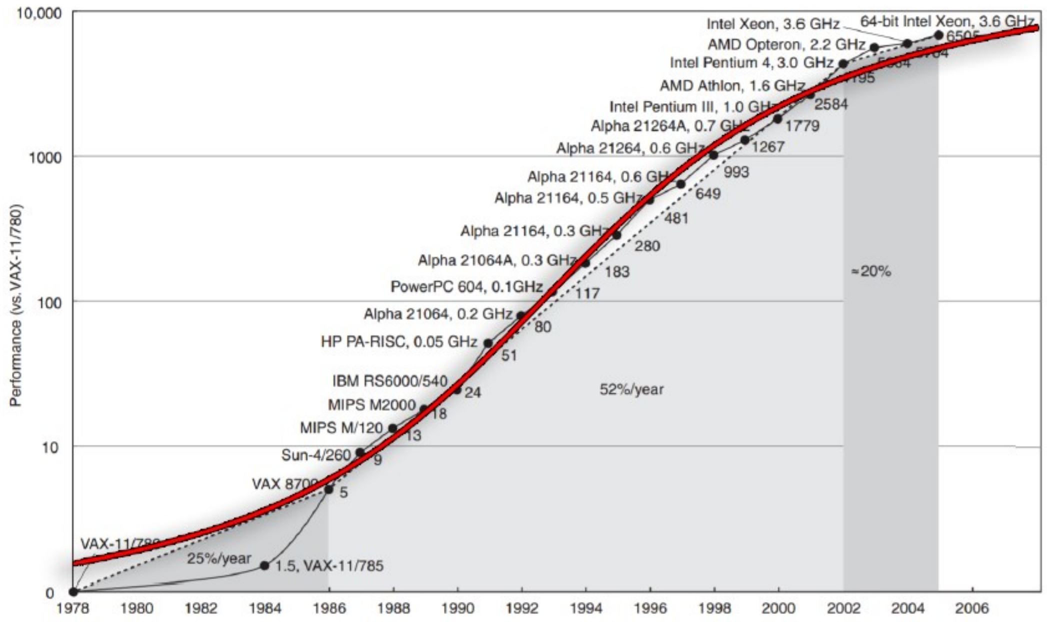
- In general case, |x| = *number of bits* needed to represent *x* (=bit length of *x*).
- But for practical purposes other measures are often used.

Algorithmic/Computational complexity

- T(n) = T(x) where $|x|=n$.
- Problem: find element b in the given array A. $|A|=n$. $T(n)=?$
- Worst case complexity: $T(n)=max\{T(x): |x|=n\}$
- Average complexity: $T_{\text{avg}}(n) = \sum T(x) \cdot p(x)$
- *Which one is more useful for practical computations?*

Algorithmic/Computational complexity

- How do we measure time complexity?
  - ✓ milliseconds, seconds, hours
  - ✓ number of basic operations
- Why bother with number of operations?
  - ✓Implementation issues
  - ✓Moore’s law



Algorithmic/Computational complexity

Asymptotic evaluation. O (Ω,Θ) notation

- ✓ *T(n) = O(f(n))* $\Leftrightarrow$ for sufficiently large *n*, *T(n)* is bounded *above* by $c \cdot f(n)$.
- ✓ *T(n) =* Ω*(f(n))* $\Leftrightarrow$ for sufficiently large *n*, *T(n)* is *at least* $c \cdot f(n)$.
- ✓ *T(n) =* Θ*(f(n))* $\Leftrightarrow$ both *T(n) = O(f(n))* and Ω*(f(n))*

Examples: $O(n)$, $O(n \cdot \log n)$, $O(n^2), O(2^n), O(n!), O(n^n)$.

Why can we omit multiplication constant?

Algorithmic/Computational complexity

Let us consider two algorithms for a problem with time complexities $O(n)$ and $O(2^n)$.

n	O(n)	$O(2^n)$
50	1.00 sec	1 sec
51	1.02 sec	2 sec
52	1.04 sec	4 sec
60	1.20 sec	17 min
70	1.40 sec	12 days
80	1.60 sec	34 years
90	1.70 sec	~ 35 000 years