How to Write Fast Code

18-645, spring 2008
3rd Lecture, Jan. 23rd

Instructor: Markus Püschel

TAs: Srinivas Chellappa (Vas) and Frédéric de Mesmay (Fred)

Miscellaneous

- First homework goes up today
- Blackboard: only for emailing and grades
- On cheating
- Computing platforms for programming exercises

Project

- **Project: topics and matching**
  - Find partner through mailing list
  - Teams of 3 are fine but more work expected
  - Email me team members and project suggestion
  - No project idea: Send me your area of interest and highest courses taken

- **Signal processing**
  - Motion estimation, Kalman filter, wavelet/frame decompositions, other image processing
- **Linear algebra**
  - SVD, LU factorization/linear system solving, many others
- **Others**
  - Coding, control, graph theory, bioimaging, …

Today

- Asymptotic analysis: multiple parameters, remarks
- Cost analysis

Asymptotic Analysis (cont'd)

- **O, Θ, Ω can be extended to multiple parameters (blackboard)**
  - Definition of O for two parameters
  - Mat-mat multiplication
  - Polynomial multiplication

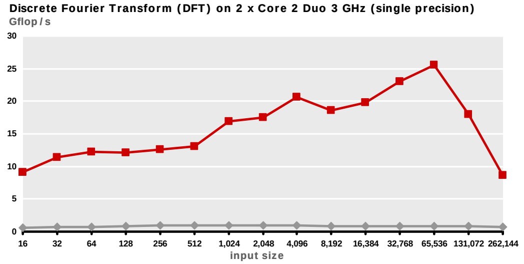
- **Avoid things like**
  - O(1000) to say "about 1000"
  - $O(2n)$, $O(\log_2(n))$, $O(n^2 + n)$, $O(mn + n)$ use instead $O(n)$, $O(\log(n))$, $O(n^2)$, $O(mn)$
  - But $n^2 + O(n)$ is ok (more precise than $O(n^2)$)

Asymptotic Analysis: Remarks

- **Asymptotic runtime analysis works because:**
  - It is independent of the exact runtime of the elementary steps counted (including memory latencies) and hence
  - It is independent of the implementation platform
  - This excludes multiple processors which introduces p = #processors as additional parameter. For example:
    - MMM (of n x n matrices) by definition is $O(n^3)$
    - On p processors one can do it in $O(n^3/p)$ (linear speed-up)

- **Problem: asymptotic analysis gives only an asymptotic idea of the runtime, but in real implementations:**
  - Constants matter:
    - n^2 is better then $10n^2$
    - 1,000,000n is probably worse than n^2 for all relevant input sizes
  - Algorithmic structure and implementation style matters: Remember?

Same operations count,
12-30x performance difference



Remember

- Complexity of a problem is usually stated using "O" and not "Θ" since every algorithm provides an upper bound, but lower bounds are often not available
- People often talk about "complexity of an algorithm" which, in a strict sense, is wrong

Cost Analysis

Refined Analysis for Numerical Problems

- **Goal: determine exact static "cost" of algorithms**

- **Approach (use MMM as running example):**
  - Fix an appropriate cost measure C: "what do I count"
  - For numerical problems typically floating point operations
  - Determine cost of algorithm as function C(n) of input size n, or, more general, of all relevant input parameters:

$$C(n_1,..,n_k)$$

  - Cost can be multi-dimensional

$$C(n_1,..,n_k) = (c_1,..,c_m)$$

- **Exact cost gives a more precise idea of runtime (constants are taken into account) but by no means the exact runtime**

Cost Analysis

- **Example**
  - Count additions and multiplications in MMM

- **Cost analysis of divide-and-conquer algorithms: Solving recurrences**
  - Great book: Graham, Knuth, Patashnik, “Concrete Mathematics,” 2nd edition, Addison Wesley 1994
  - Blackboard

For Publications and Presentations

- **Formally state the problem that you solve (see last lecture)**
- **State what is known about its complexity**
- **Analyze your algorithm (Example MMM):**
  - Define your cost measure
  - Give cost as precisely as possible/meaningful
    - Dependent on all relevant input sizes
    - At least asymptotic: “O” $\longrightarrow$ gives asymptotic runtime
    - If possible exact count since it enables performance analysis (measuring operations per second – more later)