Algorithm Engineering

Jens K. Mueller

jkm@informatik.uni-jena.de

Department of Mathematics and Computer Science
Friedrich Schiller University Jena

Monday 9th February, 2015

Summary

Exemplary Algorithm Engineering

1. Fibonacci numbers
2. Comparison-based sorting (unstable)
3. Matrix sum

Algorithm Design Paradigms

- Greedy
- Divide and conquer
- Dynamic programming
- Randomized algorithms
- Backtracking

Iterative Strategy

Design, Implement, Test, and Measure

- Simple
- Correct
- Efficient

Challenges

- Language, tooling, ...
- Follow the path between premature optimization and premature pessimization
- Generic programming
- Breaking own code by test cases (thinking outside the box)

Algorithmic Insights

- Good choice of algorithm and data structure most important
- Asymptotic complexity ignores constants and optimizes for infinite input size
- Care about realistic worst cases in your setting
- Non-optimal algorithm may be better on small input

Program Performance Metrics

- Running time for increasing input sizes
- FLOP/s
- cycle/element
- Cache misses, branch misprediction

Limits

- Instruction latency and throughput
- Theoretical performance and bandwidth

Guidelines

Simplicity

- Functions no longer than a display page
- Maximum level of indentation 3
- Use good names for functions and variables
- Avoid magic constants
- Aim for simple control flow
- Write simple expressions
- Refactor redundant code

Guidelines

Correctness

- Test your implementation
- Try to break your code
- Think about corner cases
- Use precondition, postconditions, and invariants
- Fail early, fail fast

Guidelines

Efficiency

- Measure before optimizing
- Prefer algorithmic improvements
- Go low level only when you have to
- Visualize your measurements

The Elements of Programming Style

Brian W. Kernighan and P. J. Plauger. The Elements of Programming Style. 2nd. McGraw-Hill, Inc., 1978. 168 pp. ISBN: 0-07-034207-5

The Elements of Programming Style

Introduction

- Write clearly – don't be too clever.

The Elements of Programming Style

Expression

- Say what you mean, simply and directly.
- Use library functions.
- Avoid temporary variables.
- Write clearly – don't sacrifice clarity for "efficiency."
- Let the machine do the dirty work.
- Replace repetitive expressions by calls to a common function.
- Parenthesize to avoid ambiguity.

The Elements of Programming Style (cont.)

Expression

- Choose variable names that won't be confused.
- Avoid the Fortan arithmetic `IF`.
- Avoid unnecessary branches.
- Use the good features of a language; avoid the bad ones.
- Don't use conditional branches as a substitute for a logical expression.
- Use the "telephone test" for readability.

The Elements of Programming Style

Control Structure

- Use `DO-END` and indenting to delimit groups of statements.
- Use `IF-ELSE` to emphasize that only one of two actions is to be performed.
- Use `DO` and `DO-WHILE` to emphasize the presence of loops.
- Make your programs read from top to bottom.
- Use `IF ... ELSE IF ... ELSE IF ... ELSE ...` to implement multi-way branches.
- Use the fundamental control flow constructs.
- Write first in an easy-to-understand pseudo-language; then translate into whatever language you have to use.

The Elements of Programming Style (cont.)

Control Structure

- Avoid `THEN-IF` and null `ELSE`.
- Avoid `ELSE GOTO` and `ELSE RETURN`.
- Follow each decision as closely as possible with its associated action.
- Use data arrays to avoid repetitive control sequences.
- Choose a data representation that makes the program simple.
- Don't stop with your first draft.

The Elements of Programming Style

Program Structure

- Modularize. Use subroutines.
- Make the coupling between modules visible.
- Each module should do one thing well.
- Let the data structure the program.
- Don't patch bad code – rewrite it.
- Write and test a big program in small pieces.
- Use recursive procedures for recursively-defined data structures.

The Elements of Programming Style

Input and Output

- Test input for validity and plausibility.
- Make sure input cannot violate the limits of the program.
- Terminate input by end-of-file or marker, not by count.
- Identify bad input; recover if possible.
- Treat end of file conditions in a uniform manner.
- Make input easy to prepare and output self-explanatory.

The Elements of Programming Style (cont.)

Input and Output

- Use uniform input formats.
- Make input easy to proofread.
- Use free-form input when possible.
- Use self-identifying input. Allow defaults. Echo both on output.
- Localize input and output in subroutines.

The Elements of Programming Style

Common Blunders

- Make sure all variables are initialized before use.
- Don't stop at one bug.
- Use debugging compilers.
- Initialize constants with `DATA` statements or `INITIAL` attributes; initialize variables with executable code.
- Watch out for off-by-one errors.
- Take care to branch the right way on equality.

The Elements of Programming Style (cont.)

Common Blunders

- Avoid multiple exits from loops.
- Make sure your code "does nothing" gracefully.
- Test programs at their boundary values.
- Program defensively.
- 10.0 times 0.1 is hardly ever 1.0.
- Don't compare floating point numbers just for quality.

The Elements of Programming Style

Efficiency and Instrumentation

- Make it right before you make it faster.
- Keep it right when you make it faster.
- Make it clear before you make it faster.
- Don't sacrifice clarity for small gains in "efficiency."
- Let your compiler do the simple optimizations.

The Elements of Programming Style (cont.)

Efficiency and Instrumentation

- Don't strain to re-use code; reorganize instead.
- Make sure special cases are truly special.
- Keep it simple to make it faster.
- Don't diddle code to make it faster – find a better algorithm.
- Instrument your programs. Measure before making "efficiency" changes.

The Elements of Programming Style

Documentation

- Make sure comments and code agree.
- Don't just echo the code with comments – make every comment count.
- Don't comment bad code – rewrite it.
- Use variable names that mean something.
- Use statement labels that mean something.

The Elements of Programming Style (cont.)

Documentation

- Format a program to help the reader understand it.
- Indent to show the logical structure of a program.
- Document your data layouts.
- Don't over-comment.

Reading

Jon Louis Bentley. Programming Pearls. May, 1989. Addison-Wesley, 1986. 195 pp. ISBN: 0-201-10331-1.

Jon Louis Bentley. More Programming Pearls. September, 1990. Addison-Wesley, 1988. 207 pp. ISBN: 0-201-11889-0.

Brian W. Kernighan and P. J. Plauger. The Elements of Programming Style. 2nd. McGraw-Hill, Inc., 1978. 168 pp. ISBN: 0-07-034207-5 (cit. on p. 11).

A long way . . .

Feedback