Computer Science 331

Introduction to Analysis of Algorithms

Mike Jacobson

Department of Computer Science
University of Calgary

Lecture #5

Outline

Measuring Efficiency

What sorts of measures could we use? The following are all equally valid:

- **Running Time** — no one wants to wait too long for programs to execute
- **Memory Used by Data (Storage Space)** — time is (sort of) unconstrained, but any computer can run out of memory
- **Memory Used by Code** — an issue if a program is to be stored on a low-memory device (like a smart card)
- **Time to Code** —- programmers must be paid and software development usually has deadlines!

Our focus will be on *running time* and *storage space*.

How Do We Measure Efficiency?

How can we compare algorithms or programs?

1. **Run the Code and Time the Execution.**
   Problem: Execution time is influenced by many factors:
   - *Hardware* (How fast is the CPU? How many of them?)
   - *Compiler and System Software:* (Which OS?)
   - *Simultaneous User Activity:* (Potentially affected by the time of day when the program was executed)
   - *Choice of Input Data:* (Running times can vary on inputs, even inputs of the same “size”)
   - *Programmer’s Skill*
2. **Analyze the Code**
   Advantage: Only influenced by choice of data
   Disadvantage: Can be quite difficult!

We typically try to do *both* (analysis supported by execution timings).

What Will We Measure?

Most of the time, in this course, running time and storage space will be measured in an abstract *machine-independent* way.

Running Time:

- Number of primitive operations or "steps" (programming language statements) used
- Ignores: different costs between operations (eg. multiply vs. add)

Storage Space:

- Number of words of machine memory used, assuming each word can store the same (fixed) number of bits
- Ignores: memory hierarchy differences, eg. cache vs. main memory

How Do We Wish To Measure Resources?

We will try to measure the amount of resources (time or space) used as a function of the "input size."

As described in the textbook and in this course, this will be dependent on type of input considered.

Example: if the input is an array, the appropriate measure of input size is (usually):

- number of elements

Example: if the input is a single integer, which can be virtually as large as we want, the appropriate measure of input size is:

- the bit-length of the integer

Worst-Case Analysis

Consider the *maximal* amount of resources (such as *longest* running time) used by the algorithm, on any input of a given size

Advantages of This Type of Analysis:

- upper bound on running time (guarantee that the algorithm will not take any longer)
- for some algorithms, worst-case occurs fairly often (eg. searching an array for an element not in it)

Disadvantage of This Type of Analysis:

- for some cases, the worst case rarely occurs (eg. array in reverse order is the worst case for one variation of quicksort)

Average-Case Analysis

Consider the **average** amount of resources (such as **average** running time) used by the algorithm, for an input of a given size

Advantage of This Type of Analysis:

- captures resource consumption for typical inputs

Disadvantages of This Type of Analysis:

- may be difficult to determine what the average case actually is
- typically requires probabilistic analysis

In some, but not all, cases, the worst-case and average-case running times (or amount of storage space used) are approximately the same.

Other Kinds of Analysis

Best-case Analysis:

- Consider the *minimal* amount of resources (such as *shortest* running time) used by the algorithm, on any input of a given size
- Eg. sorted array input to insertion sort

Amortized Analysis:

- time required to perform a sequence of operations is averaged over all operations performed
- different from average case — guarantees average performance per operation in the worst case
- Eg. if $T(n)$ is the worst case cost to perform n operations, then $T(n)/n$ is the amortized (average) cost per operation

Further Reading

Textbook, Section 2.8

- also includes material covered in classes later this week

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms*, Second Edition

- on reserve in the library (and free online)
- includes *much* more material about this topic