Unit #2: Complexity Theory and Asymptotic Analysis

CPSC 221: Algorithms and Data Structures

Lars Kotthoff[1]
larsko@cs.ubc.ca

[1]With material from Will Evans, Steve Wolfman, Alan Hu, Ed Knorr, and Kim Voll.

Unit Outline

- Brief proof reminder
- Algorithm Analysis: Counting steps
- Asymptotic Notation
- Runtime Examples
- Problem Complexity

Learning Goals

- Given code, write a formula that measures the number of steps executed as a function of the size of the input.
- Use asymptotic notation to simplify functions and to express relations between functions.
- Know the asymptotic relations between common functions.
- Understand why to use worst-case, best-case, or average-case complexity measures.
- Give examples of tractable, intractable, and undecidable problems.

Proof by . . .

- Counterexample
  - show an example which does not fit with the theorem
  - Thus, the theorem is false.
- Contradiction
  - assume the opposite of the theorem
  - derive a contradiction
  - Thus, the theorem is true.
- Induction
  - prove for a base case (e.g., $n = 1$)
  - assume for all $n \leq k$ (for arbitrary k)
  - prove for the next value ($n = k + 1$)
  - Thus, the theorem is true.

Example: Proof by Induction (worked) 1/4

Theorem:

A positive integer x is divisible by 3 if and only if the sum of its decimal digits is divisible by 3.

Proof:

Let $x_1 x_2 x_3 \ldots x_n$ be the decimal digits of x.
Let the sum of its decimal digits be

$$S(x) = \sum_{i=1}^{n} x_i$$

We'll prove the stronger result:

$$S(x) \bmod 3 = x \bmod 3.$$

How do we use induction?

Example: Proof by Induction (worked) 2/4

Base Case:

Consider any number x with one ($n = 1$) digit (0-9).

$$S(x) = \sum_{i=1}^{n} x_i = x_1 = x.$$

So, it's trivially true that $S(x) \bmod 3 = x \bmod 3$ when $n = 1$.

Example: Proof by Induction (worked) 3/4

Inductive hypothesis:

Assume for an arbitrary integer $k > 0$ that for any number x with $n \leq k$ digits:

$$S(x) \bmod 3 = x \bmod 3.$$

Inductive step:

Consider a number x with $n = k + 1$ digits:

$$x = x_1 x_2 \ldots x_k x_{k+1}.$$

Let z be the number $x_1 x_2 \ldots x_k$. It's a k-digit number so the inductive hypothesis applies:

$$S(z) \bmod 3 = z \bmod 3.$$

Example: Proof by Induction (worked) 4/4

Inductive step (continued):

$$
\begin{aligned}
x \bmod 3 &= (10z + x_{k+1}) \bmod 3 && (x = 10z + x_{k+1}) \\
&= (9z + z + x_{k+1}) \bmod 3 \\
&= (z + x_{k+1}) \bmod 3 && (9z \text{ is divisible by } 3) \\
&= (S(z) + x_{k+1}) \bmod 3 && (\text{induction hypothesis}) \\
&= (x_1 + x_2 + \cdots + x_k + x_{k+1}) \bmod 3 \\
&= S(x) \bmod 3
\end{aligned}
$$

QED (quod erat demonstrandum: "what was to be demonstrated")

A Task to Solve and Analyze

Find a student's name in a class given her student ID

Analysis of Algorithms

- Analysis of an algorithm gives insight into
  - how long the program runs (time complexity or runtime) and
  - how much memory it uses (space complexity).
- Analysis can provide insight into alternative algorithms
- Input size is indicated by a non-negative integer n (sometimes there are multiple measures of an input's size)
- Running time is a real-valued function of n such as:
  - $T(n) = 4n + 5$
  - $T(n) = 0.5n \log n - 2n + 7$
  - $T(n) = 2^n + n^3 + 3n$

Rates of Growth

Suppose a computer executes 1op per picosecond (trillionth):

$n =$	10
$\log n$	1ps
n	10ps
$n \log n$	10ps
n^2	100ps
2^n	1ns

Rates of Growth

Suppose a computer executes 1op per picosecond (trillionth):

$n =$	10	100
$\log n$	1ps	2ps
n	10ps	100ps
$n \log n$	10ps	200ps
n^2	100ps	10ns
2^n	1ns	1Es

Exasecond(Es) $=$ 32 billion years

Rates of Growth

Suppose a computer executes 1op per picosecond (trillionth):

$n =$	10	100	1,000
$\log n$	1ps	2ps	3ps
n	10ps	100ps	1ns
$n \log n$	10ps	200ps	3ns
n^2	100ps	10ns	1μs
2^n	1ns	1Es	10^{289}s

Exasecond(Es) $=$ 32 billion years

Rates of Growth

Suppose a computer executes 1op per picosecond (trillionth):

$n =$	10	100	1,000	10,000
$\log n$	1ps	2ps	3ps	4ps
n	10ps	100ps	1ns	10ns
$n \log n$	10ps	200ps	3ns	40ns
n^2	100ps	10ns	1μs	100μs
2^n	1ns	1Es	10^{289}s	

Exasecond(Es) $=$ 32 billion years

Rates of Growth

Suppose a computer executes 1op per picosecond (trillionth):

$n =$	10	100	1,000	10,000	10^5
$\log n$	1ps	2ps	3ps	4ps	5ps
n	10ps	100ps	1ns	10ns	100ns
$n \log n$	10ps	200ps	3ns	40ns	500ns
n^2	100ps	10ns	1μs	100μs	10ms
2^n	1ns	1Es	10^{289}s		

Exasecond(Es) $=$ 32 billion years

Rates of Growth

Suppose a computer executes 1op per picosecond (trillionth):

$n =$	10	100	1,000	10,000	10^5	10^6
$\log n$	1ps	2ps	3ps	4ps	5ps	6ps
n	10ps	100ps	1ns	10ns	100ns	1μs
$n \log n$	10ps	200ps	3ns	40ns	500ns	6μs
n^2	100ps	10ns	1μs	100μs	10ms	1s
2^n	1ns	1Es	10^{289}s			

Exasecond(Es) $=$ 32 billion years

Rates of Growth

Suppose a computer executes 1op per picosecond (trillionth):

$n =$	10	100	1,000	10,000	10^5	10^6	10^9
$\log n$	1ps	2ps	3ps	4ps	5ps	6ps	9ps
n	10ps	100ps	1ns	10ns	100ns	1μs	1ms
$n \log n$	10ps	200ps	3ns	40ns	500ns	6μs	9ms
n^2	100ps	10ns	1μs	100μs	10ms	1s	1week
2^n	1ns	1Es	10^{289}s				

Exasecond(Es) $=$ 32 billion years

Analyzing Code

```
// Linear search
find(key, array)
  for i = 0 to length(array) - 1 do
    if array[i] == key
      return i
  return -1
```

1) What's the input size, n?

Analyzing Code

```
// Linear search
find(key, array)
  for i = 0 to length(array) - 1 do
    if array[i] == key
      return i
  return -1
```

2) Should we assume a worst-case, best-case, or average-case input of size n?

Analyzing Code

```
// Linear search
find(key, array)
  for i = 0 to length(array) - 1 do
    if array[i] == key
      return i
  return -1
```

3) How many lines are executed as a function of n?

$T(n) =$

Are lines the right unit?

Analyzing Code

The number of lines executed in the worst-case is:

$$T(n) = 2n + 1.$$

- Does the "2" matter?
- Does the "1" matter?