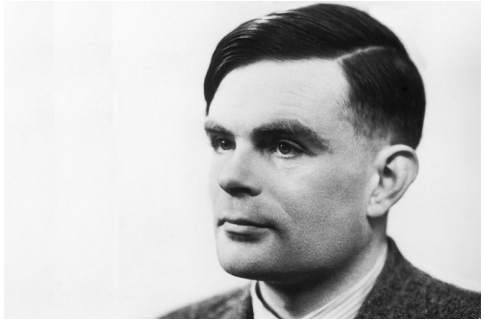
Asymptotic Analysis

C.S. Marron
cmarron@umbc.edu

CMSC 341 — Data Structures

Foundations of Computing

Alan Turing and Alonzo Church



...and me at Turing Labs in Sherborne

Computability

What is computable?

- Turing devised "Turing machines" to address this question.
- Church devised the λ-calculus.
- Both showed that there are logical statements that cannot be evaluated.
- Turing machines, λ-calculus, Random Access Machines, and several other computational models are all equivalanet. They can compute the same things.
- Church-Turing Thesis. The functions that can be computed with a "paper-and-pencil method" are exactly those which can be computed by a Turing machine (or λ-calculus, RAM, etc.).

Complexity

Do the models compute with the same speed?

- No!
- To determine the complexity of an algorithm — space and time requirements — we need to pick a model.
- We use Random Access Machines. Similar to “real” computers.
- We can count operations to estimate running times, but we need a concise way to describe it.

Big-Oh

Definition

- Let $T(n)$ and $f(n)$ be non-negative functions of the positive integers.
- We say that $T(n)$ is $O(f(n))$ if there exists a positive constant c and a positive integer n_0 such that

$$T(n) \leq cf(n)$$

  whenever $n \geq n_0$.
- $f(n)$ is an *asymptotic* upper bound on $T(n)$.
- $T(n)$ is any old function, but we think of it as representing the running time of an algorithm with input size n.

Example:Linear Search

Operation Counting

- We will count the basic operations.
- The result will be a polynomial in n.

```
int search( int target, int *data, int n) {
  for ( int i = 0; i < n; i++ ) {
    if ( data[i] == target ) {
      return i;
    }
  }
  return -1;
}
```

A Useful Theorem

Theorem

- Suppose $T(n)$ is a polynomial.
- Ex: $3n^2 + 7n + 2$ or $5n + 12$.
- . . .and the highest power in $T(n)$ is n^a for some non-negative integer a.
- Then $T(n)$ is $O(n^a)$.

Therefore, `search()` is $O(n)$.

Big-Omega

Definition

- Let $T(n)$ and $f(n)$ be non-negative functions of the positive integers.
- We say that $T(n)$ is $\Omega(f(n))$ if there exists a positive constant c and a positive integer n_0 such that

$$T(n) \geq cf(n)$$

  whenever $n \geq n_0$.
- These are *different* constants c and n_0 than for Big-Oh.
- $f(n)$ is an *asymptotic* lower bound on $T(n)$.

Example: Back to Linear Search

Counting for Big-Omega

- We need a `lower` bound on the running time of `search()`.
- Under what conditions does it finish most quickly?
- ...when it finds `target` at the first index of `data`!
- Count the operations in this case.
- We should get a constant.

For `search()`, the Big-Oh and Big-Omega bounds are different.

Example:Linear Search II, The Searchening

Determine Big-Oh and Big-Omega Bounds

```
int search2( int target, int *data, int n) {
  int indx = -1;
  for ( int i = 0; i < n; i++ ) {
    if ( data[i] == target ) {
      indx = i;
    }
  }
  return indx;
}
```

- `search2()` is $O(n)$ *and* $\Omega(n)$.

Big-Theta

Definition

- Let $T(n)$ and $f(n)$ be non-negative functions of the positive integers.
- We say that $T(n)$ is $\Theta(f(n))$ if there exists positive constants c, d and a positive integer n_0 such that

$$cf(n) \leq T(n) \leq df(n)$$

  whenever $n \geq n_0$.
- Theorem: $T(n)$ is $\Theta(f(n))$ iff $T(n)$ is $O(f(n))$ and $T(n)$ is $\Omega(f(n))$.

Example: Linear Search (cont.)

Big-Theta Bounds

- `search2()` is $\Theta(n)$ since it is $O(n)$ and $\Omega(n)$.
- `search()` is *not* $\Theta(n)$.

Reading Assignment

Read the following sections in the textbook

1. Section 4.2.1: Experimental Studies
2. Section 4.2.2: Primitive Operations
3. Section 4.2.3: Asymptotic Notation
4. Section 4.2.4: Asymptotic Analysis