Algorithm Analysis and Modeling

Data Structures and Algorithms

Warm Up

From Last Lecture:

Imagine you have implemented a Map with the following functions:

- put(key,value)
- get(key)
- size()

What are some test cases you would use to check it works?

Socrative:

www.socrative.com

Room Name: CSE373

Please enter your name as: Last, First

Administrivia

Due dates:

- Partner form Thursday April 5th at 1:59pm
- Homework #1 Friday April 6th at 11:59pm
- Class Survey Friday April 6th at 11:59pm

Notes:

- Sign up for piazza!
- Set up your development environment
- Project 1 to go out Friday

Asymptotic Analysis

asymptotic analysis: how the runtime of an algorithm grows as the data set grows

Approximations

- Basic operations take “constant” time
  - Assigning a variable
  - Accessing a field or array index
- Consecutive statements
  - Some of time for each statement
- Function calls
  - Time of function’s body
- Conditionals
  - Time of condition + maximum time of branch code
- Loops
  - Number of iterations x time for loop body

Modeling Case Study

Goal: return ‘true’ if a sorted array of ints contains duplicates

Solution 1: compare each pair of elements

```
public boolean hasDuplicate1(int[] array) {
   for (int i = 0; i < array.length; i++) {
      for (int j = 0; j < array.length; j++) {
         if (i != j && array[i] == array[j]) {
            return true;
         }
      }
   }
   return false;
}
```

Solution 2: compare each consecutive pair of elements

```
public boolean hasDuplicate2(int[] array) {
   for (int i = 0; i < array.length - 1; i++) {
      if (array[i] == array[i + 1]) {
            return true;
      }
   }
   return false;
}
```

Modeling Case Study: Solution 2

T(n) where n = array.length

-> work inside out

Solution 2: compare each consecutive pair of elements

```
public boolean hasDuplicate2(int[] array) {
    for (int i = 0; i < array.length - 1; i++) {
        if (array[i] == array[i + 1]) { +4
            return true; +1
        }
    }
    return false; +1
}
```

If statement = 5

x n - 1

$T(n) = 5\,(n-1) + 1$

linear time complexity class O(n)

Modeling Case Study: Solution 1

Solution 1: compare each consecutive pair of elements

```
public boolean hasDuplicate1(int[] array) {
    for (int i = 0; i < array.length; i++) { x n
        for (int j = 0; j < array.length; j++) { x n
            if (i != j && array[i] == array[j]) { +5
                return true; +1
            }
        }
    }
    return false; +1
}
```

6

6n

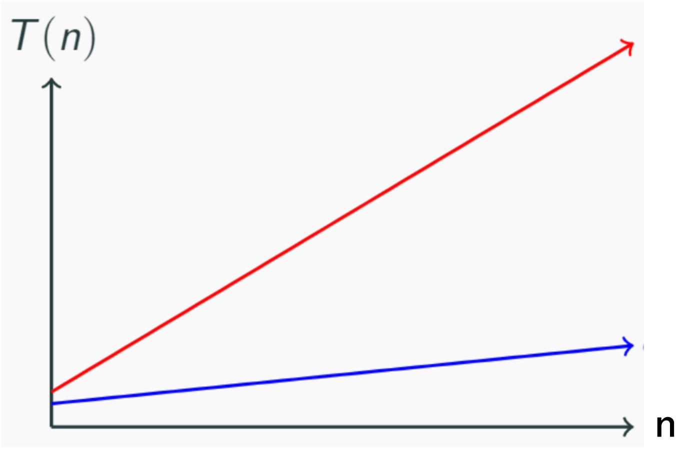
$6n^2$

$T(n) = 6\,n^2 + 1$

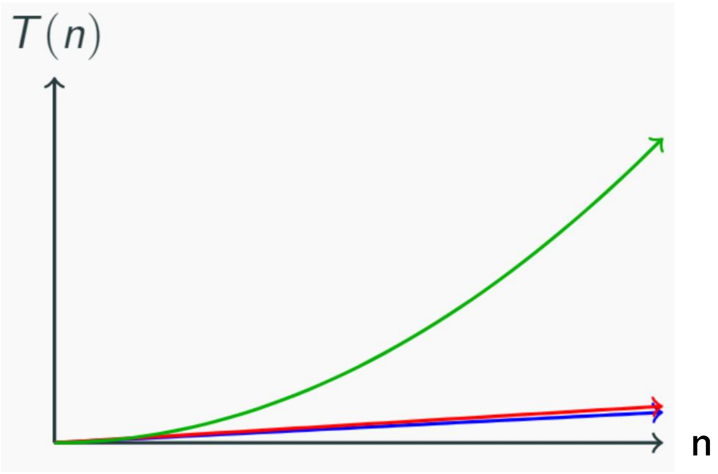
quadratic time complexity class $O(n^2)$

Comparing Functions

Function growth

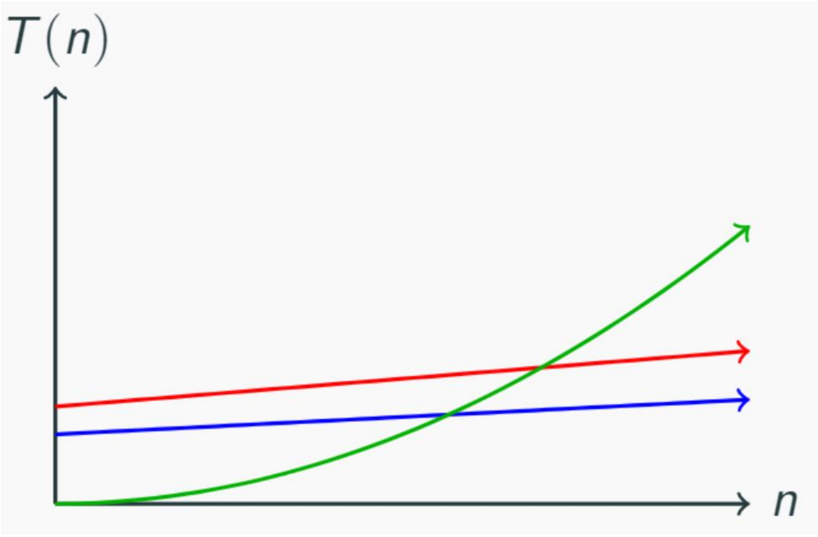


n and 4n look very different up close



n and 4n look the same over time

n^2 eventually dominates n



n^2 doesn't start off dominating the linear functions

It eventually takes over…

Function comparison: exercise

$f(n) = n \leq g(n) = 5n + 3$? True – all linear functions are treated as equivalent

$f(n) = 5n + 3 \leq g(n) = n$? True

$f(n) = 5n + 3 \leq g(n) = 1$? False

$f(n) = 5n + 3 \leq g(n) = n^2$? True – quadratic will always dominate linear

$f(n) = n^2 + 3n + 2 \leq g(n) = n^3$? True

$f(n) = n^3 \leq g(n) = n^2 + 3n + 2$? False

Definition: function domination

Definition: Domination

A function f(n) is dominated by g(n) when…

There exists two constants $c > 0$ and $n_0 > 0$

Such that for all values of $n \geq n_0$

$f(n) \leq c * g(n)$

Example:

Is $f(n) = n$ dominated by $g(n) = 5n + 3$?

$c = 1$

$n_0 = 1$

Yes!

Exercise: Function Domination

Demonstrate that $5n^2 + 3n + 6$ is dominated by n^3 by finding a c and n_0 that satisfy the definition of domination

$5n^2 + 3n + 6 \le 5n^2 + 3n^2 + 6n^2$ when $n \ge 1$

$5n^2 + 3n^2 + 6n^2 = 14n^2$

$5n^2 + 3n + 6 \le 14n^2$ for $n \ge 1$

$14n^2 \le c*n^3$ for c = ? n >= ?

$\frac{14}{n}$ -> c = 14 & n >= 1

Definition: Big O

If $f(n) = n \leq g(n) = 5n + 3 \leq h(n) = 100n$ <u>and</u>

$h(n) = 100n \leq g(n) = 5n + 3 \leq f(n)$ n

Really they are all the "same"

Definition: Big O
O(f(n)) is the "family" or "set" of <u>all</u> functions that are <u>dominated by</u> f(n)

Question: are O(n), O(5n + 3) and O(100n) all the same?

True! By convention we pick simplest of the above -> O(n) ie "linear"

Definitions: Big Ω

“f(n) is greater than or equal to g(n)”

F(n) dominates g(n) when:

There exists two constants such that c > 0 and n0 > 0

Such that for all values n >= n0

F(n) >= c * g(n) is true

Definition: Big Ω

Ω(f(n)) is the family of all functions that dominates f(n)

∈ Element Of

f(n) is dominated by g(n)

Is that the same as

"f(n) is contained inside O(g(n))"

Yes!

f(n) ∈ g(n)

Examples

$4n^2 \in \Omega(1)$

true

$4n^2 \in \Omega(n)$

true

$4n^2 \in \Omega(n^2)$

true

$4n^2 \in \Omega(n^3)$

false

$4n^2 \in \Omega(n^4)$

false

$4n^2 \in O(1)$

false

$4n^2 \in O(n)$

false

$4n^2 \in O(n^2)$

true

$4n^2 \in O(n^3)$

true

$4n^2 \in O(n^4)$

true

Definition: Big O

O(f(n)) is the "family" or "set" of all functions that are dominated by f(n)

Definition: Big Ω

Ω(f(n)) is the family of all functions that dominates f(n)

Definitions: Big Θ

We say $f(n) \in \Theta(g(n))$ when both

$f(n) \in O(g(n))$ and $f(n) \in \Omega\ (g(n))$ are true

Which is only when $f(n) = g(n)$

Definition: Big Θ
$\Theta(f(n))$ is the family of functions that are equivalent to $f(n)$

Industry uses “Big Θ” and “Big O” interchangeably