Thu 15 Feb 2018.

Test 1 tomorrow.
- no lecture 11-12 tomorrow
- test 12-1

Ch5 Analyzing Algorithm Running Time

- focus: compute the right thing (correct)
  - use accepted style guidelines
  - will it finish on time
  - compare algorithms to apply in your situation. Which is best?

How to measure runtime?

- stopwatch?
- flaws: influenced by external factors
  - CPU type, memory type/amount
  - other jobs on computer running
- average over a few observations
  - hard to extend to other situations
- climate modelling: how does runtime change
  - if double # weather stations
- stopwatch not enough.

Other factors:

- Insertion sort
- runtime depended on initial state of list
  - almost sorted $\sim n \log(n)$
  - reversed $\sim n^2$

- how to describe runtime for an "average" initial list.
  - worst case runtime
  - best case runtime
  - average case runtime

---

How to describe runtime of:

```python
def print_items(lst):
    for item in lst:
        print(item)
```

- count "basic operations" that are performed
- list has length $n$
• print items $n$ print operations
  $\rightarrow n$ basic operations

• count loop operation cost items $\text{list}[i]$  
  $\rightarrow another \ n$ basic operations  
  $i: 0 \ldots \text{len(list)}$

  total: $2n$ basic operations

• try to account for difference between time to 
  print and time to do variable assignment  
  say factor is 10.

  total: $10n + 1n$ basic operations  
  $= 11n$

• calling + returning from func takes time

  $\sim 1.5$ basic operations

  new total: $11n + 1.5$ basic operations

4 descriptions of routine $n$ basic operation  
$2n$  
$11n$  
$11n + 1.5$

Which one is right?

$\rightarrow$ can't say b/c depends on unknowns
- O can say all depend linearly
  on n, double list size
  and double runtime

New Q: not how long will it take
  but "how does runtime change as the list size changes?"

Want to be able to describe how runtime change .... do this formally
  describe how functions grow ... 

Describe runtime using function:

\[ f : \mathbb{N} \rightarrow \mathbb{R}^+ \]

\[ \forall a, b \text{ lists have discrete length} \]

- will develop terminology to describe
  long term growth of functions
  (working towards
  formal description of
  \textit{big-Oh})

- comparing algorithms
  by comparing functions that describe their runtime
**def:** Let \( f, g : \mathbb{N} \to \mathbb{R}^+ \)

we say that \( g \) is absolutely dominated by \( f \) iff \( \forall n \in \mathbb{N}, \ g(n) \leq f(n) \)

\[
f(n) = n^2 ; \quad g(n) = n \]

Now consider \( f(n) = n^2 \) \quad \( g(n) = 5n \)

**def:** Let \( f, g : \mathbb{N} \to \mathbb{R}^+ \).

we say that \( g \) is dominated by \( f \) up to a constant factor if there exists a positive real number \( c \) s.t.

\[\forall n \in \mathbb{N}, \ g(n) \leq c \cdot f(n)\]

\[\exists c \in \mathbb{R}^+, \forall n \in \mathbb{N}, \ g(n) \leq c \cdot f(n)\]

\[\implies \text{Let } c = 5 \quad \ldots\]

**Note:** constant factor idea allows us to consider \( n, 2n, 3n \) as describing equivalent functions
Also: \( f(n) = n^2 \) \( g(n) = n + 90 \)

- No constant \( c \) makes \( c \cdot f(n) \geq g(n) \) at \( n = 0 \)

Need a way to exclude these values.

**Def.** Let \( f, g : \mathbb{N} \rightarrow \mathbb{R}^+ \). We say that \( g \) is eventually dominated by \( f \) if:

\[
\exists n_0 \in \mathbb{R}^+ \text{ s.t. } \forall n \in \mathbb{N}, \text{ if } n \geq n_0 \text{ then } g(n) \leq f(n)
\]

\[
\exists n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, \text{ if } n \geq n_0 \Rightarrow g(n) \leq f(n)
\]

E.g. \( f(n) = n^2 \), \( g(n) = n + 90 \)

Let \( n_0 = 90 \) \( \Rightarrow \) \( n \geq 90 \) \( \Rightarrow \) \( g(n) \leq f(n) \)

- Combine eventually dominating + dominating up to a constant factor
**def:** Let $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$ be functions. We say that $g$ is eventually dominated by $f$ up to a constant factor if there exists a $c, n_0 \in \mathbb{R}^+$ such that for all $n \in \mathbb{N}$, if $n > n_0$, then $g(n) \leq c \cdot f(n)$.

**too wordy:** say $g$ is big oh of $f$.

$g \in O(f)$

**example:** Let $f(n) = n^2$ and $g(n) = 100n + 5000$

Prove that $g \in O(f)$

i.e., $100n + 5000 \in O(n^2)$

**translation:**

$\exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n > n_0 \Rightarrow g(n) \leq cf(n)$

$\exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n > n_0 \Rightarrow 100n + 5000 \leq c \cdot n^2$

$100n \ vs \ cn^2 \rightarrow c = 100$

$n_0 = 50 \quad 100n_0 \sim 5000$

Then \[ 100n \leq 100n^2 \]

$\quad \sqrt{cn^2} \quad 100n + 500$
Proof: Let \( c = 100 \) and \( n_0 = 50 \).

Let \( n \geq n_0 \) and assume \( n \geq n_0 \).

Want to show \( 100n + 5000 \leq cn^2 \)

Start with \( n \geq 50 \) \( \iff \) assumption

\[ n + n \geq n + 50 \] (add \( n \) to both sides)

\[ 2n \geq n + 50 \]

Since \( n \cdot n \geq n + 50 \) \( \quad \) (since \( n \geq 2 \))

\[ n^2 \geq n + 50 \]

Since \( n^2 \geq 50 \cdot 2 \)

\[ 100n^2 \geq 100n + 5000 \) \( \iff \) goal

\[ cn^2 \geq 100n + 5000 \]

\( \therefore \) have shown \( \exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow g(n) \leq c \cdot f(n) \)

\( \therefore \quad g \in O(f) \)
Announcements: PS2, Midterm, PS3

Last time: Ch 5 Runtime Analysis

- Focus: how runtime changes (grows) as a function of problem size (n ∈ N)

- Runtime: \( f : \mathbb{N} \rightarrow \mathbb{R}^+ \)
  \[
  \underbrace{\text{problem size}}_{\text{time}}
  \]

- Compare functions: eventually greater?

\[ \rightarrow \text{big-O}\]

"Is eventually dominated by up to a constant factor."

We say \( f, g : \mathbb{N} \rightarrow \mathbb{R}^+ \),

\( g \in \Theta(f) \iff \) \( g(n) \leq f(n) \)

but want to ignore constant factors

\[ \exists c \in \mathbb{R}^+ \quad g(n) \leq c \cdot f(n) \]

focus on long-term behavior

not all \( n \) but eventually
\exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n > n_0 \Rightarrow g(n) \leq cf(n)

\text{Goal:} \quad 100n \in O(n) \quad n \in O(100n)

\text{choose "simplest" big \(O\) function}

\(O(100n) = O(n)\)

\text{A more complex example:}

Prove that \(2n^3 - 5n^4 + 7n^6 \in O(n^2 - 4n^6 + 6n^8)\)

\text{translate:}

\exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n > n_0 \Rightarrow

\[2n^3 - 5n^4 + 7n^6 \leq c(n^2 - 4n^6 + 6n^8)\]

\text{discuss:}

Let \(c = \_\) and \(n_0 = \_

\text{Assume } n \in \mathbb{N} \text{ and } n > n_0

\[2n^3 - 5n^4 + 7n^6 \leq 2n^3 + 7n^6 \leq 2n^6 + 7n^6 = 9n^6 \leq 9n^8\]

\text{want bigger terms.}

\text{work towards } n^8

\text{work backwards from } \text{RHS:}

\[c(n^2 - 4n^6 + 6n^8) \geq c(-4n^6 + 6n^8) \geq c(-4n^8 + 6n^8)\]
\[ = 2^c n^8 \\
= 2^{-9/2} n^8 \text{ for } c = 9/2 \\
= 9 n^8 \\
\geq 2 n^3 - 5 n^4 + 7 n^6 \]

[ Formal Proof - exercise: \( c = 9/2 \), \( n_0 = 0 \) works ]

[ above is a little tedious \( \sim \) refer to function hierarchy ]

Theorem [Big-Oh Hierarchy Theorem] [BOTT]

For all \( a, b \in \mathbb{R}_+ \), the following are True:

1. if \( a > 1 \) and \( b > 1 \) then \( \log_a n \in O(\log_b n) \)

   \text{consequence: } \log_2 n \in O(\log_{10} n) \\
   \text{and } \log_{10} n \in O(\log_2 n) \]

2. if \( a \leq b \), then \( n^a \in O(n^b) \)

   \text{consequence: } n^2 \in O(n^3) \]

3. if \( 1 \leq a \leq b \), then \( a^n \in O(b^n) \)

   \text{consequence: } 2^n \in O(3^n) \]
4. If $a > 1$ and $b > 0$ 
   \[ \log_a n \in \mathcal{O}(n^b) \]
   
   - logs dominated by polynomials
   
   \[ \log_2 n \in \mathcal{O}(n^{0.0001}) \]

5. If $b > 1$ then $n^a \in \mathcal{O}(b^n)$
   
   - polynomials dominated by exponential
   
   \[ n \in \mathcal{O}(1.01^n) \]

All relate to single term functions

Need results to say what happens when combine terms

**Theorem:** 2. For $f, g : N \to \mathbb{R}^+$, if $g \in \mathcal{O}(f)$ then $f + g \in \mathcal{O}(f)$

\[ f + g \rightarrow (f + g)(n) = f(n) + g(n) \]

Example: Prove $n^2 + n \in \mathcal{O}(n^2)$
Proof: By \( OHT \), we know \( n \in O(n^2) \).

By Theorem, we know \( f, g \in O(f) \)

\[ n^2 + n \in O(n^2) \]

( cleaner than earlier proof )

Theorem (general sum)

\[ \forall f, g, h : \mathbb{N} \rightarrow \mathbb{R}^+ \]

\[ f \in O(h) \land g \in O(h) \]

\[ \Rightarrow f + g \in O(h) \]

Theorem (constant multiple)

\[ \forall f, g : \mathbb{N} \rightarrow \mathbb{R}^+ , \forall a \in \mathbb{R}^+ , \]

\[ g \in O(f) \Rightarrow \left[ a \cdot g \right] \in O(f) \]

\[ \forall n \in \mathbb{N} , \ (a \cdot g)(n) = a \cdot g(n) \]
\[ \frac{2}{3} n^3 + 5n^2 + 7 \in O(n^3) \]
also \( \in O(n^{47}) \)

exercise: \[ 100 n^2 + (\frac{1}{2})^{10} 2^n \in O(2^n) \]

Theorem: general product

\[ \forall f_1, f_2, g_1, g_2 : N \to \mathbb{R} \]
\[ g_1 \in O(f_1) \land g_2 \in O(f_2) \]
\[ \Rightarrow g_1 \cdot g_2 \in O(f_1 \cdot f_2) \]