Thu 15 Feb 2018

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

Ch 5 Analyzing Algorithm Running Time

  focus: compute the right thing (correct)
  
  to use accepted style guidelines
  
  to will it finish on time
  
  to 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 modeling -> how does runtime change
t  if double # weather stations
Other factors:

- \texttt{CSC108}: insertion sort
- runtime depended on initial state of list
  - almost sorted \(\sim n \text{ (list length)}\)
  - 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 in \( \text{list[i]} \)
  \( \rightarrow \) 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: \( 10n + \ln \) basic operations
  \( = \frac{1}{10}n \)

• calling and returning from function takes time
  \( \sim 1.5 \) basic operations

  new total: \( 11n + 1.5 \) basic operations

4 descriptions of runtime \( n \) basic operations
  \( 2n \)
  \( \ln \)
  \( 11n + 1.5 \)

Which one is right?

\( \rightarrow \) can't say, but depends on unknowns
I can say all depend linearly on \( n \) double list size
double runtime

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

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

describe runtime using function:

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

since runtime \( \geq 0 \)

- b/c lists have discrete length
- will develop terminology to describe long term growth of functions
  (working towards formal description of)
  big-O notation
- comparing algorithms by comparing functions that describe their runtime
**def**

Let \( f, g: \mathbb{N} \rightarrow \mathbb{R}^+ \)
we say that \( g \) is absolutely dominated by \( f \) iff
\[
\forall n \in \mathbb{N}, \ g(n) \leq f(n)
\]

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

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

**def**

Let \( f, g: \mathbb{N} \rightarrow \mathbb{R}^+ \).
we say that \( g \) is dominated by \( f \)
up to a constant factor iff there exists a positive real number \( c \) s.t.
\[
\forall n \in \mathbb{N}, \ g(n) \leq c \cdot f(n)
\]
\[
( \exists c \in \mathbb{R}^+, \forall n \in \mathbb{N}, \ g(n) \leq c \cdot f(n) )
\]

Let \( c = 5 \) ...

**note**: constant factor idea allows us to consider \( n, 2n, 1/n \)
as describing 'equivalent functions'
Also: \[ f(n) = n^2 \quad g(n) = n + 90 \]

\[
\begin{align*}
\text{let } n_0 & \in \mathbb{N} \\
\exists n_0 \in \mathbb{R}^+ \text{ s.t. } \forall n \in \mathbb{N}, \text{ if } n > n_0 \implies g(n) \leq f(n)
\end{align*}
\]

- no constant \( C \) makes \( C \cdot f(n) \geq g(n) \) at \( n = 0 \)

\( n_0 \) need a way to exclude these values.

\( \text{let } f, g : \mathbb{N} \to \mathbb{R}^+ \text{. We say that } \)

\( g \) is eventually dominated by \( f \) if

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

( \( \exists n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n > n_0 \implies g(n) \leq f(n) \)

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

\[ \text{let } n_0 = 90 \implies n > 90 \implies g(n) \leq f(n) \]

- combine eventually dominating + dominating up to a constant factor
defn: Let \( f, g : \mathbb{N} \rightarrow \mathbb{R}^+ \). We say that \( g \) is eventually dominated by \( f \) up to a constant factor if \( g \) there exists a \( c, n_0 \in \mathbb{R}^+ \) s.t. for all \( n \in \mathbb{N} \), if \( n \geq n_0 \) then \( g(n) \leq c \cdot f(n) \)

too wordy: say \( g \) is big oh of \( f \);
\( g \in \mathcal{O}(f) \)

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

Prove that \( g \in \mathcal{O}(f) \)

i.e., \( 100n + 5000 \in \mathcal{O}(n^2) \)

translation:
\[
\exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n > n_0 \Rightarrow g(n) \leq c f(n)
\]

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

\[100n \text{ vs } cn^2 \quad \rightarrow \quad c = 100 \]

\[n_0 = 50 \quad 100n_0 \sim 5000 \]

\[
\begin{align*}
\text{Then} \quad 100n & \leq 100n^2 \\
n & > 1
\end{align*}
\]

\[
\sqrt{c n^2} \quad \frac{100n}{100n + 5000}
\]
Proof: Let $C = 100$ and $n_0 = 50$.
Let $n \in \mathbb{N}$ and assume $n \geq n_0$.
Want to show $100n + 5000 \leq Cn^2$.

Start with $n \geq 50$ \hspace{1cm} \leftarrow \text{assumption}

- $n + n \geq n + 50$ \hspace{1cm} (add $n$ to both sides)
- $2n \geq n + 50$

Since $n \cdot n \geq n + 50$ \hspace{1cm} (since $n > 2$)

Since $n \cdot n \geq 2n$ \hspace{1cm} $n^2 \geq n + 50$

Since $n \geq 50$ \hspace{1cm} $100n^2 \geq 100n + 5000$ \hspace{1cm} $\text{goal}$

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

\[ \therefore \text{have shown } \exists C, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \implies g(n) \leq C \cdot f(n) \]

\[ g \in O(f) \]