Class 21

Homework setup until VSCode works for you
Analysis terminology
Big-O Theorems
Trees?
Homework Software Problems

- To use sketch.sh
  - Open a new sketch (click upper right)
  - add a "code block" by clicking < >

- Paste contents of CS17setup.re into first code block
- Remove the two
HOMEWORK setup cont.

- Paste contents of CS17setup.re into first block:

```ruby
try
/*    (ignore(input())); */
(Test_Failed (Actual_Result(input()), Expected_Error(expect))){
  | (Failure (err)) when err == expect => Test_Passed
  | (Failure (err)) => Test_Failed (Actual_Error(err), Expected_Error(expect))});
```

```ruby
60
61
62
63
64
65
66
67
```
HOMEWORK setup cont.

- Click on the "Run" button at the top left
  - If everything works, yay!
  - If you get errors because of Js.log..., you can comment out those lines until things work.

```java
try
    /* (ignore(input())); */
    (Test_Failed (Actual_Result(input())), Expected_Error(expect)) {
        | (Failure (err)) when err = expect ⇒ Test_Passed
        | (Failure (err)) ⇒ Test_Failed (Actual_Error(err), Expected_Error(expect))
```

HOMEWORK setup cont.

• Use the second block to write your program. Put
/* open CS17setup */
at the top so you don't forget it:

```
| (Failure (err)) ⇒ Test_Failed (Actual_
let check_error: (unit ⇒ 'a, string) ⇒ check_

/* open CS17setup */
type matrix('a) = list(list('a));
let horz_flip : matrix('a) ⇒ matrix('a) = ...
```
HOMEWORK setup cont.

- When your program is working, copy everything from the second block to "horz_flip.re" (or whatever name is given in the HW asst), and uncomment the "open CS17setup" line
- Then move on to the next problem:
  - Delete the second block (if you like)
  - Replace it with the next problem's code (again with the commented-out "open CS17setup;" line!)
Big-O categories again

• Goals
  • Learn how we use big-O (and big- \( \Omega \) and big- \( \Theta \)) to describe the runtime of procedures
  • Learn a practical style for showing that a function \( f \) is in some big-O class
  • Learn how various big-O classes are related to save grief later
Comparing functions

• $f \in O(g)$ means that $f$ is eventually less than $g$, up to constants.
• For list-procedures, if the op-counting function $f$ is in $O(n \leftrightarrow n)$, we're generally pretty happy; we say that the procedure "runs in at most linear time."
• There's a way to characterize a comparison in the other direction:
  • if $f$ is eventually less than $g$, up to constants, we say that $g \in \Omega(f)$.
  • Formally, $\Omega(f)$ denotes the set of all functions $g$ with the property that $f$ is eventually less than $g$, up to constants.
  • Very informally: $\Omega(f)$ consists of functions "that grow at least as fast as $f$"
A combination

• The set $\Theta(f)$ consists of functions $g$ with the properties that
  • $f$ is eventually less than $g$, up to constants, and
  • $g$ is eventually less than $f$, up to constants
• Again informally: $\Theta(f)$ consists of functions that "grow the same way as $f"$
• When we looked at the op-counter, $L$, for the length procedure, we found that
  \[
  \begin{align*}
  L(0) &= A \\
  L(n) &= B + L(n - 1) \text{ for } n > 0
  \end{align*}
  \]
  and via plug-n-chug, arrived at the (correct) conjecture that
  \[ L(n) = Bn + A. \]
\[ L(n) = Bn + A \]

where \( A, B > 0 \).

- It's not hard to see that \( L \in O(n \mapsto n) \); an \((M, c)\)-proof uses \( M = 1, c = A + B \).

- It's also true that the function \( n \mapsto n \) is in \( O(L) \); use \( M = 1, c = \frac{1}{B} \).

- Hence the runtime of the length procedure is in \( \Theta(n \mapsto n) \), and we can say that length is not merely "at worst linear time", but is in fact "linear time".

- Sadly, many CS folks say "such-and-such is \( O(n \mapsto n^2) \)" to mean that it's \( \Theta(n \mapsto n^2) \), for instance, conflating big-O and big-Theta classes.
Big-O theorems
(so you never have to think again)
Basic examples

• For any function \( f: \mathbb{N} \to \mathbb{N}, \ f \in O(f). \)
• Proof: Pick \( M = 1, \ c = 1. \)
• Suppose that \( n > M. \) [completely unnecessary in this case]
• I claim that \( f(n) \leq cf(n) \) for \( n > M. \)
• Rewrite: \( f(n) \leq 1 \cdot f(n) \) for \( n > M. \)
• Obviously true!
• This pick-M-and-c pattern is so common that we can replace a complete proof with "\((M, c) = (1, 1)" after a while."
Basic examples (2)

• For $f : \mathbb{N} \to \mathbb{N}$, and for $Z : \mathbb{N} \to \mathbb{N}$: $n \mapsto 0$, we have $Z \in O(f)$.
• Proof: Pick $M = 1, c = 1$.
• Suppose that $n > M$. [again unnecessary]
• I claim that $Z(n) \leq c f(n)$ for $n > M$.
• Rewrite: $0 \leq 1 \cdot f(n)$ for $n > M$.
• Obviously true!
Basic examples (3)

• For \( f : \mathbb{N} \to \mathbb{N} \), we have \( 2f \in O(f) \).
• Proof: Pick \( M = 1, c = 2 \).
• Suppose that \( n > M \). [still not needed!]
• I claim that \( (2f)(n) \leq cf(n) \) for \( n > M \).
• Rewrite: \( (2f)(n) = 2 \cdot f(n) \leq 2 \cdot f(n) \) for \( n > M \).
• Obviously true!
Basic examples (4)

- For $f : \mathbb{N} \to \mathbb{N}$, and any number $a > 0$ we have $af \in O(f)$.
- Proof: Pick $M = 1, c = a$.
- Suppose that $n > M$. [same]
- And $(af)(n) = a \cdot f(n)$
- I claim that $(af)(n) \leq cf(n)$ for $n > M$.
- Rewrite: $(af)(n) = a \cdot f(n) \leq a \cdot f(n)$ for $n > M$.
- Obviously true!
if \( f \in O(g) \) and \( g \in O(h) \), then \( f \in O(h) \)

Suppose that (1) for \( n > 100 \), we know that \( f(n) \leq 22 \ g(n) \).
And (2) for \( n > 33 \), we know that \( g(n) \leq 4h(n) \).

Then, from (2), we know that for \( n > 100 \), \( g(n) \leq 4h(n) \).
What can we say (for \( n > 100 \)), about the relationship of \( f(n) \) to \( h(n) \)?

\[
\begin{align*}
  f(n) &\leq 22 \ g(n) \\
  g(n) &\leq 4 \ h(n) \\
  22 \ g(n) &\leq 4 \cdot 22 \ h(n) \\
  f(n) &\leq 4 \cdot 22 \ h(n).
\end{align*}
\]
if $f \in O(g)$ and $g \in O(h)$, then $f \in O(h)$[cont]

Prior slide used specific numbers, but the pattern is clear:

if
for $n > M_1$ we have $f(n) \leq c_1 g(n)$ and
for $n > M_2$ we have $g(n) \leq c_2 h(n)$
then
for $n > \max(M_1, M_2)$, we have $f(n) \leq c_1 c_2 h(n)$.
Hence $f \in O(h)$.

Shorthand: if $[M_1, c_1]$ and $[M_2, c_2]$ show that $f \in O(g)$ and $g \in O(h)$, then $(M, c) = [\max(M_1, M_2), c_1 c_2]$ shows that $f \in O(h)$. 
...more to come...