test 2 next Thu - details this Thu.

before break:

- analyzing runtime of algorithms
- goal: $\text{RT}_{\text{alg}}(n) \in \Theta(\text{some function of } n)$
- for trickier code, this may be hard to achieve. Want to at least show:

\[ \text{RT}_{\text{alg}}(n) \in O(h_1(n)) \]
\[ \text{RT}_{\text{alg}}(n) \in \Omega(h_2(n)) \]

where $h_1(n)$, $h_2(n)$ are simple functions that grow differently.
Last example: \( x_0 \), given, then \( x = \begin{cases} x_{i+1} = x_i / 2, & \text{for } x_i \text{ even}, \\ \quad i = 0, 1, 2, 3, \ldots \\ x_{i+1} = 3x_i + 1, & \text{for } x_i \text{ odd}. \end{cases} \)

Stop when \( x_{i+1} = 1 \).

No proof that this sequence terminates for an arbitrary starting point \( x_0 > 1 \).

So instead consider:

```python
def f(n: int) -> int:
    """Precondition: \( n > 1 \)""
    steps = 0
    x = n
    while x > 1:
        if x % 2 == 0:
            x = x // 2  # as before
        else:
            x = 2*x - 2  # change from Collatz
        steps += 1
    return steps
```

Goal: Find \( h_f(n), h_2(n) \) such that

- \( RT_f(n) \in O(h_f(n)) \)
- \( RT_f(n) \in \Omega(h_2(n)) \)

Stronger goal: \( \text{with } h_f(n) = h_2(n) \)
Try a few values of \( n \) and trace.

\[
\begin{array}{c|c}
\text{n=10} & \frac{10}{2} = 5 \\
\text{(even)} & \\
\end{array}
\]

\[
\begin{array}{c|c}
\text{n=13} & \frac{13}{2} = 6.5 \\
\text{(odd)} & \\
\end{array}
\]

\[
\begin{array}{c|c}
\text{x} & \{ x_{i+1} = \left\{ \begin{array}{ll} x_i/2, & x_i \text{ even} \\ 2x_i - 2, & x_i \text{ odd} \end{array} \right. \\
\text{# steps} & 5 \\
\end{array}
\]

Observe: once \( x \) is a power of \( 2^2 \), \( x \) stays even until 1.

\[
\begin{array}{c|c}
\text{n=16} & \frac{16}{2} = 8 \\
\text{=2^4} & \\
\end{array}
\]

\[
\begin{array}{c|c}
\text{x} & \{ x_{i+1} = \left\{ \begin{array}{ll} x_i/2, & x_i \text{ even} \\ 2x_i - 2, & x_i \text{ odd} \end{array} \right. \\
\text{# steps} & 4 \\
\text{= log_2 16} \\
\end{array}
\]

\# steps varies in an unknown way with \( n \).

Suppose in stead consider 2 consecutive executions of while loop
\[
\begin{align*}
\text{x odd:} \quad & x \rightarrow 2x - 2 \rightarrow (2x - 2)/2 = x - 1 \\
\text{check} & \\
\text{x even:} \quad & x \rightarrow x/2 \\
\text{even} & \rightarrow x/2/2 \\
\text{odd} & \rightarrow 2(x/2) - 2 = x - 2
\end{align*}
\]

So after two consecutive iterations of while loop \( x \) goes down by at least 1.

\[\therefore \text{total \# iterations} \leq 2 \cdot (n - 1)\]

while \( x > 1 \)

\[\therefore \text{RT}_f(n) \leq 2n\]

and could prove \( \text{RT}_f(n) \in O(n) \)

\[\rightarrow c = 2\]

What about \( \text{RT}_f(n) \in \mathcal{O}(?) \)

observed \( n = \text{power of 2} \)

\[\text{\# iterations is } \log_2(n)\]

so try to prove \( \text{RT}_f(n) \in \mathcal{O}(\log_2(n)) \)
Need to prove:
\[ \exists c_2, n_2 \in \mathbb{R}^+, \forall n \in \mathbb{N}, \quad n > n_2 \Rightarrow R_{T_f}(n) \geq c_2 \log_2(n) \] "at least"

How to prove?
\[ x \to \begin{cases} \frac{x}{1/2} \\ 2x - 2 \end{cases} \]

Each time through while loop
- \( x \) decreases by at most \( 1/2 \).

(a factor of)

So we get to \( x = 1 \) after at least \( k \) iterations where \( k \) is such that
\[ \frac{n}{2^k} \leq 1 \quad \Rightarrow \quad k \geq \log_2(n) \]
\[ R_{T_f}(n) \]

Conclusion:
\[ RT_f(n) \in O(n) \quad \text{not tight} \]
\[ RT_f(n) \in \mathcal{O}(\log n) \]

Can we get a \( \Theta \) bound?
Can show \( RT_f(n) \in \Theta(\log n) \)

Hint: look at 3 consecutive loop executions!

Exercise: plot \( n \) vs \( RT_f(n) \) and observe a trend.
Now consider the function:

```python
def has_even(numbers: List[int]) -> bool:
    for number in numbers:
        if number % 2 == 0:
            return True
    return False
```

**Description:**
Return True if and only if `numbers` contains an even item.

**How qualitatively different from previous examples:**
- runtime depends on both size of list and also contents of list and location of first even

<table>
<thead>
<tr>
<th>e.g. numbers</th>
<th>#iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>[3, 3, 3]</td>
<td>3</td>
</tr>
<tr>
<td>[3, 4, 3]</td>
<td>2</td>
</tr>
<tr>
<td>[4, 3, 3]</td>
<td>1</td>
</tr>
</tbody>
</table>
Q: How to describe the behavior of these "slices" as $n$ varies?

A: one option: focus on extreme values of slice.

Define: worst case runtime

$$WC_f(n) = \max \{ \text{runtime of } f(x) \mid x \text{ has size } n \}$$

best case runtime

$$BC_f(n) = \min \{ \text{runtime of } f(x) \mid x \text{ has size } n \}$$
max/min values of same set
(one set of runtimes per n value)

Consider \( WC_{\text{h.e.}} \)

\[ WC(n) \in O(n) \quad \text{means} \quad \forall c, n \in \mathbb{R}^+, \forall x \in X, n > n_0, \Rightarrow WC(n) \leq c \cdot n \]

\[ \max \{ RT_{\text{h.e.}}(x) \mid x \text{ has size } n \} \leq c \cdot n \]

\[ \forall \text{ inputs } x, \ x \text{ has size } n \Rightarrow RT_{\text{h.e.}}(x) \leq c \cdot n \]

[If largest value in set \( \leq A \), then all values in set \( \leq A \)].

\[ WC_{\text{h.e.}}(n) \in \Omega(n) \quad \text{means} \]

\[ \exists c_3, n_3 \in \mathbb{R}^+, \forall x \in X, n > n_3 \Rightarrow WC(n) \geq c_3 \cdot n \]

\[ \max \{ RT_{\text{h.e.}}(x) \mid x \text{ has size } n \} \geq c_3 \cdot n \]

can be proven by showing that there is an input \( x \) of size \( n \) such that...
takes at least $C_3 n$ steps.

$$\max \{ \mathcal{R}_{h, e}(x) \mid x \text{ has size } n \}$$

$\geq \mathcal{R}_{h, e}(x)$ for each $x$ of size $n$

So $\max \{ \} \geq C_3 n$

If $\mathcal{R}_{h, e}(x) \geq C_3 n$ for some input $x$

need to show $\exists$ input $x$ of size $n$, $\mathcal{R}_{h, e}(x) \geq C_3 n$