test 2 next Thu - details this Thu.

before break:

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

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

\( h_1(n), h_2(n) \) are simple functions that grow differently.
Collatz sequence

Last example:

$x_0$ given, then $x = \begin{cases} \frac{x_i}{2}, & \text{for } x_i \text{ even;} \\ 3x_i + 1, & \text{for } x_i \text{ odd.} \end{cases}

$$i = 0, 1, 2, 3, \ldots$$

Stop when $x_{i+1} = 1$.

No proof that this sequence terminates for an arbitrary starting point int $x_0 > 1$.

So instead consider:

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
        return steps + 1

Goal: Find $h_1(n)$, $h_2(n)$ such that

$$RT_f(n) \in \Omega(h_1(n))$$
and $$RT_f(n) \in \Omega(h_2(n))$$

Stronger goal: with $h_1(n) = h_2(n)$
\[ x \rightarrow \begin{cases} \frac{x}{2} & \text{if } x \text{ even} \\ 2x - 2 & \text{if } x \text{ odd} \end{cases} \]

\begin{align*}
n=10 & \quad x & \quad n=13 & \quad x \\
(\text{even}) & \quad \underline{10} & (\text{odd}) & \quad \underline{13} \\
& \quad \underline{5} & & \quad \underline{24} \\
& \quad \underline{4} & & \quad \underline{12} \\
& \quad \underline{2} & & \quad \underline{6} \\
& \quad \underline{1} & & \quad \underline{3} \\
& & \quad \underline{4} & \quad \underline{2} \\
& & \quad \underline{1} & \quad \underline{1} \\
\#\text{steps}=5 & & \#\text{steps}=7
\end{align*}

Observations:

Once \( n \) is a power of 2, we stay even.

\begin{align*}
n=16 & \quad x \\
& \quad \underline{16} \\
& \quad \underline{8} \\
& \quad \underline{4} \\
& \quad \underline{2} \\
& \quad \underline{1} \\
\#\text{steps}=4 & = \log_2 16
\end{align*}

Suppose we consider 2 consecutive executions of the while loop.
\[ x \text{ odd: } x \rightarrow 2x - 2 \rightarrow \left(\frac{2x-2}{2}\right) = x - 1 \Rightarrow \text{ even} \]
\[ x \text{ even: } x \rightarrow x/2 \rightarrow \frac{x}{2}/2 \]
\[ \text{odd} \rightarrow 2 \left(\frac{x}{2}\right) - 2 = x - 2 \Rightarrow \text{even} \]

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

\[ \begin{align*}
2i+1 & \quad \text{at most } n-1 \\
2i & \quad \text{at most } n-2 \\
\vdots & \quad \text{at most } 2 \\
& \quad \text{at most } n-2 \text{ times } 1
\end{align*} \]

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

\[ \therefore RT_f(n) \leq 2n \]

and we could prove \( RT_f(n) \in O(n) \)

\[ \exists c=2, n= \ldots \]

What about \( RT_f(n) \in \Omega(\cdot) \)?

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

\[ \# \text{ iterations } \log_2 (n) \]
So let's try to show $RT_f(n) \in \Omega(\log_2 n)$

Need to prove:

$\exists c_2, n_2 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n > n_2 \Rightarrow RT_f(n) > c_2 \log_2 n$

How to prove:

$O$ bound: showed $x$ decreases by at least 1 each it.

$\Omega$ bound: show $x$ decrease by at least something

Every time we go through while loop $x$ decrease by a factor of at most $1/2$.

So we get to $x=1$ after at least $k$ iterations where $k$ is such that $\frac{n}{2^k} \leq 1$.

So $k \geq \log_2 n$ iterations are required.

So can prove $RT_f(n) \in \Omega(\log_2 n)$

Summary: we know $RT_f(n) \in O(n)$ or maybe tight $RT_f(n) \in \Omega(\log_2 n)$ tight

Can we do better? (i.e. get a $\Theta$ bound?)

You can show $RT_f(n) \in \Theta(\log_2 n)$

Hint: look at 3 consecutive iterations.
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 value.

**How qualitatively different from previous examples:**

<table>
<thead>
<tr>
<th>examples</th>
<th>numbers</th>
<th># iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[3,3,3]</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>[3,4,3]</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>[4,3,3]</td>
<td>1</td>
</tr>
</tbody>
</table>

3 different
6 list of size n = 3

In addition to depending on the size of the list, the runtime also depends on the contents.
of the list

Runtime graphs:

\[ \text{Runtime} \]

\[ \text{fixed list size} \]

but runtime varies

How to describe the behaviour of these slices as \( n \) varies?

One option: focus on extreme values in the slice.

We define:

Worst Case Runtime

\[ WC_f(n) = \max \left\{ \frac{\text{runtime of } f(x)}{f(x)} \mid x \text{ has size } n \right\} \]

Best Case Runtime

\[ BC_f(n) = \min \left\{ \frac{\text{runtime of } f(x)}{f(x)} \mid x \text{ has size } n \right\} \]

\( \max/\min \text{ values of same set (one set per value of } n) \)
Consider \( WC_{h.e}(n) \).

Looks like \( WC_{h.e}(n) \in O(n) \)

means
\[
\exists c, n \in \mathbb{R}^+, \forall x \in X, \ n > n_1 \implies WC_{h.e}(n) \leq c \cdot n
\]

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

\( \forall \text{ input } x, \ x \text{ has size } n \implies RT_{h.e}(x) \leq c \cdot n \)

\( WC_{h.e} \in \mathcal{O}(n) \) means
\[
\exists c_2, n_2 \in \mathbb{R}^+, \forall x \in X, n > n_2 \implies WC_{h.e}(n) \geq c_2 n
\]

\[
\Rightarrow \max \{ RT_{h.e}(x) \mid x \text{ has size } n \} \geq c_2 n
\]

\( \) can be proven by showing that there is an input \( x \) of size \( n \) that takes at least \( c_2 n \) steps.

\[\max \{ RT_{h.e}(x) \mid x \text{ has size } n \} \]
2. \( R_{h.e}(x) \) for each \( x \)

So \( \max \{ \} \) will be \( \geq c_2 n \)

If \( R_{h.e}(x) \geq c_2 n \) for some \( x \)

\[ \exists \text{ input } x \text{ of size } n, \quad R_{h.e}(x) \geq c_2 n. \]