Describe the growth of function runtimes.

Have seen 3 situations

1. Code where all loops run to completion.
   - Plot might look like...
   - In this case, easy to determine a $\Theta$ bound.
   - E.g. $RT_f(n) \in \Theta(n^2)$

2. Code where loops sometimes finish early.
   - Only one runtime for each $n$.
   - In this case, want to determine $h_1(n)$, $h_2(n)$.
   - S.t. $RT_f(n) \in O(h_1(n)) \land RT_f(n) \in \Omega(h_2(n))$.
   - When $h_2 \in \Theta(h_1)$, conclude $RT_f(n) \in \Theta(h_1(n))$

3. Code where loops depend on $n$ differently.
in this case, want to describe the extreme values in each slice

\[ \begin{align*}
\max \to \mathcal{WC}_f(n) & \quad \text{Worst Case} \\
\min \to \mathcal{BC}_f(n) & \quad \text{Best Case}
\end{align*} \]

More formally:

\[ \begin{align*}
\mathcal{WC}_f(n) &= \max \left\{ \text{runtime of } f(x) \mid \text{input } x \text{ has size } n \right\} \\
\mathcal{BC}_f(n) &= \min \left\{ \text{runtime of } f(x) \mid \text{input } x \text{ has size } n \right\}
\end{align*} \]

- These are 2 different fans that we want to describe
  - determine \( g(n), g'(n) \) s.t.
  
  \[ \begin{align*}
  \mathcal{WC}_f(n) \in O(g(n)) & \land \mathcal{WC}_f(n) \in \Omega(g'(n)) \\
  \text{desire} & \quad g(n) \in \Theta(g'(n))
  \end{align*} \]
so then $\text{WC}_f(n) \in \Theta(\text{g}'(n))$

- similarly for $\text{BC}_f(n)$


determine $h(n), h'(n)$ s.t.

$$\text{BC}_f(n) \in O(h(n)) \land \text{BC}_f(n) \in \Omega(h'(n))$$

desire $h(n) \in \Theta(h'(n))$

so hence $\text{BC}_f(n) \in \Theta(h'(n))$

---

Unpack definition:

$\text{WC}_f(n) \in O(\text{g}(n))$

$\iff \exists c_0, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n > n_0 \Rightarrow \text{WC}_f(n) \leq c_0 \text{g}(n)$

$\Rightarrow \max \{\text{runtime of } f(x) \mid x \text{ has size } n\} \leq c_0 \text{g}(n)$

if $\max \text{ runtime } \leq c_0 \text{g}(n)$

then all runtimes $\leq c_0 \text{g}(n)$

on input of size $n$

$\forall$ inputs $x$ of size $n$, runtime of $f(x) \leq c_0 \text{g}(n)$
def has_even(x):
    for i in x:
        if i % 2 == 0:
            return True
    return False

for a description of WC, ask "what is most # operations?"

・ let \( n = \text{len}(x) \)
・ the # of loop iterations is at most \( n \)
・ 1 basic operation for "return False"

\[
\text{runtime of } \text{has_even}(n) \text{ is at most } n + 1 \\
\text{basic operations}
\]

\[
\therefore \text{ for } n \geq 1,
\text{ A inputs } x \text{ of size } n, \text{ runtime of } \text{has-even}(x) \leq 2 \cdot n \in \mathbb{O}
\]

\[
\therefore \text{ WC } \text{has-even}(n) \in \mathbb{O}(n).
\]

Goal: & desc. WC has-even(n)
    & consider \( \subseteq \).
Unpack definition: \( WC_f(n) \in \mathcal{O}(h(n)) \)

\[
\Rightarrow \quad \exists c, n \in \mathbb{R}^+, \forall n \geq n, \quad 0 \leq c \cdot h(n) \leq WC_f(n)
\]

\[
\Rightarrow \quad c \cdot h(n) \leq \max\{\text{runtime of } f(x) \mid x \text{ has size } n\}
\]

- not all runtime of \( f(x) \) need be \( \geq c \cdot h(n) \)
- but if \( \max\{\} \geq c \cdot h(n) \),

the runtime \( f(x) \geq c \cdot h(n) \) for some \( x \) of size \( n \)

\[\exists \text{ input } x \text{ of size } n, \quad c \cdot h(n) \leq \text{runtime of } f(x) \]

\[
\Rightarrow \text{ so we need to be able to describe an input for each size } n \text{ that has runtime } \geq c \cdot h(n).
\]

Back to has_even: desire is to show

\[
WC_{\text{has.even}}(n) \in \mathcal{O}(n)
\]

to match \( O(n) \).

- How can we force has_even to take at least \( n \) basic operations?
- make the list contain all odd values $\underbrace{\ldots}_{\text{items}}$

- $\forall n \in \mathbb{N}, \ n > 1$, define $x_n = \underbrace{1, 1, 1, \ldots, 1}_{\text{description of an input family}}$

- runtime of $\text{has\_even} (x_n)$ is at least $n$ basic operations.

Hence, for $n \geq 1$,

$\exists$ an input $x$ of size $n$, $\frac{1}{2}n \leq \text{runtime of } \text{has\_even}(x)$

$\therefore \text{WC}_{\text{has\_even}} (n) \in \mathcal{O}(n)$
A more complex example: palindrome prefix

**defn:** A string $s$ is a palindrome iff it reads the same forwards as backwards.

$$s[i] == s[-1-i], \text{ i} \in \text{range}(|s|)$$

e.g. 'racecar', 'bob', 'x'

**defn:** A string $s_1$ is a prefix of string $s_2$ iff $s_1[i] == s_2[i], \text{ i} \in \text{range}(|s_1|)$

**problem:** Given a nonempty string $s$, return the length of the longest prefix of $s$ that is a palindrome.

e.g. "attack"

<table>
<thead>
<tr>
<th>prefixes</th>
<th>palindrome?</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>'a'</td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>'at'</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>'att'</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>'atta'</td>
<td>T</td>
<td>4</td>
</tr>
<tr>
<td>'attack'</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>'attack'</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>
def palindrome_prefix(s: str) -> int:
    """Return the length of the longest prefix of a nonempty string s that is a palindrome."

>>> palindrome_prefix('attack')
4
"""

n = len(s)
for prefix_length in range(n, 0, -1):  # goes from n down to 1
    is_palindrome = True  # assume it is until know otherwise
    i = 0
    while is_palindrome and i < prefix_length:
        if s[i] != s[prefix_length - 1 - i]:
            is_palindrome = False
            i = i + 1
    if is_palindrome:
        return prefix_length

**Problem:** describe WC \((n)\) \(\text{palindrome_prefix}\)

The total \# of loop iterations is not more than:

\[
\sum_{i=0}^{n} i = \frac{n(n+1)}{2}
\]

\(\therefore \text{WC}(n) \in \Theta(n^2)\)

**Assumes** each loop goes through all possible values

(\(\text{won't happen but okay since looking for upper bound on \# operations}\))
\[ W \in \mathbb{Z}^2(?) \]

\[ \text{desire } n^2 \text{ to match} \]

- examples
  
  \[ s = 'a a a a ... a' \text{ returns } \text{len}(s) \]

- inner loop runs \( n \) times
- outer loop runs once only
  
  \[ \Rightarrow \text{runtime } \sim n \]

\[ s = 'a b b b ... b' \text{ returns } 1 \]

- inner loop runs once each time
- outer loop runs \( n \) times
  
  \[ \Rightarrow \text{runtime } \sim n \]

How to find input so runtime \( \sim n^2 \)

- \( \text{want } \# \text{ outer loop its. to } \sim n \)
- \( \text{and also } \# \text{ innerloop its. to } \sim n \)