Describe the growth of function runtimes.

Have seen 3 situations

1. Code where all loops run to completion.
   - Plot might look like:
     \[ \text{RT}_f(n) \]
   - In this case, easy to determine a \( \Theta \) bound.
     \( e.g. \text{RT}_f(n) \in \Theta(n^2) \)

2. Code where loops sometimes finish early.
   - Only one runtime for each \( n \).
   - In this case, want to determine \( h_1(n), h_2(n) \)
     \( s.t. \text{RT}_f(n) \in O(h_1(n)) \land \text{RT}_f(n) \in \Omega(h_2(n)) \)
     - When \( h_2 \in \Theta(h_1) \), conclude \( \text{RT}_f(n) \in \Theta(h_1(n)) \)

3. Code where loops depend on \( n \)
In this case, want to describe the extreme values in each slice:

max \rightarrow WC_f(n) \quad Worst\ Case

min \rightarrow BC_f(n) \quad Best\ Case

\[ WC_f(n) = \max \{ \text{runtime of } f(x) | \text{input } x \text{ has size } n \} \]

\[ BC_f(n) = \min \{ \text{runtime of } f(x) | \text{input } x \text{ has size } n \} \]
Similarly for $BC_f(n)$ we want $h, h'$ s.t.

$$BC_f(n) \in O(h(n)) \land BC_f(n) \in \Omega(h'(n))$$

Because $h(n) \in \Theta(h'(n))$ so then

$$BC_f(n) \in \Theta(h(n))$$

Unpacking the def:

$$WC_f(n) \in O(g(n))$$

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

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

If $\max \text{ runtime } \leq c_0 g(n)$ then all runtimes $\leq c_0 g(n)$ for fixed $n$

$\forall$ inputs $x$ of size $n$, runtime of $f(x) \leq c_0 g(n)$
Back to has_even function:

def has_even(x):
    for i in x:
        if i % 2 == 0:
            return True
    return False

- the # of loop iterations is at most $n$
- 1 basic operation for 'return False'

: runtime of has_even($x$) is at most $n+1$

basic operations.

- to prove band:

  for $n \geq 1$,

  \[
  \forall \text{ inputs } x \text{ of size } n, \text{ runtime } \text{ of } \text{has-even}(x) \leq 2 \cdot n
  \]

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

  unpack the \text{ def}\n
  \[
  \text{WC}_{f}(n) \in \Omega(h(n))
  \Rightarrow \exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n > n_0, \Rightarrow c, h(n) \leq \text{WC}_{f}(n)
  \]
\[
\max_{x \text{ has size } n} \{ \text{runtime of } f(x) \} \leq c \cdot h(n)
\]

- not all runtimes \( f(x) \) need to be \( \geq c \cdot h(n) \).
- but if \( \max \{ \} \geq c \cdot h(n) \), then runtime \( f(x) \geq c \cdot h(n) \) for some \( x \) of size \( n \).

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

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

- back to has. even
  - our desire is to show \( WC(n) \in \mathcal{O}(n) \) has even
    - to match \( \mathcal{O}(n) \)
    - how to force has. even to take at least \( n \) basic op?
    - make a list that only contains odds

- \( \forall n \in \mathbb{N}, \ n \geq 1 \), define \( x_n \) with \( n \) items

\[
x_n = [1, 1, 1, \ldots, 1]
\]
called an input family

- runtime of \text{has-eve}_n (X_n) is at least \( n \) basic operations

Hence for \( n \geq n \),

If an input \( x \) of size \( n \), \( 1 \leq n \leq \text{runtime of } \text{has-eve}_n(x) \)

\[ \therefore \text{WC}_{\text{has-eve}}(n) \in \Omega(n) \]

\[ \therefore \text{WC}_{\text{has-eve}}(n) \in \Theta(n) \]

\[ \max \{ \} \leq c \cdot g(n) \]
\[ \rightarrow \text{A value in set, value } \leq c \cdot g(n) \]

\[ c \cdot h(n) \leq \max \{ \} \]
\[ \rightarrow \text{A value in set, } c \cdot h(n) \leq \text{value} \]

\[ \text{could consider BC but will instead do?} \]
A more complex example: palindrome prefix

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

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

e.g. "racecar", "bob", "x"

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

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>
</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
        # Check whether s[0:prefix_length] is a palindrome
        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 current prefix is a palindrome, return its length
        if is_palindrome:
            return prefix_length

Problem:  describe WC \( n \)

\[ \text{prefix length} = n \quad n-1 \quad n-2 \]
\[ = \frac{n(n+1)}{2} \]
\[ \therefore \text{WC} (n) \in \Theta(n^2) \]
\[ \text{P-P} \]

[argument assumes each loop goes through all possible values]

\[ \text{WC} (n) \in \Theta(n^2) \]
\[ \text{desire } n^2 \text{ if possible.} \]
try a few examples

\[ s = 'aaaa \ldots a' \quad \text{return} \quad n \]

- inner loop runs \( n \) times
- outer loop runs once

\[ \text{runtime} \sim n \]

\[ s = 'abbbb \ldots b' \quad \text{return} \quad 1 \]

- inner loop runs once each time
- outer loop runs \( n \) times

\[ \text{runtime} \sim n \]

- runtime \( \sim n^2 \)
  - outer loop runs \( a \) times \( \sim n \)
  - inner loop runs \( a \) times \( \sim n \) each time