Example

def twisty(n: int):
    \[x = n\]
    while \[x > 1\]:  # iterations?
        if \[x \mod 2 == 0\]:
            \[x = \frac{x}{2}\]
        else:
            \[x = 2x - 2\]

    \[\Theta(1)\]

- Upper bound (on # iterations)

ROUGH WORK!

\[\begin{align*}
\text{odd} & \rightarrow 2x - 2 \quad \text{even} \rightarrow \frac{2x - 2}{2} = x - 1 \\
\text{\textit{iteration \#1}} & \rightarrow \frac{x}{2}
\end{align*}\]

\[\begin{align*}
\text{odd} & \rightarrow 2(\frac{x}{2}) - 2 = x - 2 \\
\text{even} & \rightarrow \frac{x}{4}
\end{align*}\]

\[\text{iteration \#2}\]

Since \[x\] becomes \[\leq x - 1\] after 2 iterations, there are at most \[2x\] iterations (with \[x = n\] initially) \(\Rightarrow\) \[\leq 2n\] iterations

So \[RT(n) \in O(n)\].


def find(lst: List[int], x: int) -> int:
    for i in range(len(lst)):
        if lst[i] == x:
            return i  # early termination
    return -1  # x not found

    \[\Theta(1)\]
Notice: runtime depends not only on input size \( n = \text{len}(\text{lst}) \), but also on the contents/value of \( \text{lst} \) and \( x \).

\[ WC(n) = \max \{ RT(x) : x \text{ is an input of size } n \} \]

- upper bound \( U(n) \) on \( WC(n) \) satisfies:
  
  For all inputs \( x \) of size \( n \), \( RT(x) \leq U(n) \)

- lower bound \( L(n) \) on \( WC(n) \) satisfies:
  
  For some input \( x \) of size \( n \), \( RT(x) \geq L(n) \)
Example

def is_palindrome(s: str) -> bool:
    n = len(s)
    for i in range(n//2):
        if s[i] != s[n-1-i]:
            return False
    return True

Upper bound [want: general argument for all inputs of size n, for arbitrary n]  
- the loop cannot iterate more than \( n/2 \) times for any input  
- so \( WC(n) \in O(n) \)

Lower bound [want: describe specific input of size n, for arbitrary n, where \( is\_palindrome \) takes "long enough" — try to match upper bound]  
- consider \( s = a^a \ldots a \)  
- loop condition \( s[i] \neq s[n-1-i] \) evaluates to False for \( i = 0, 1, \ldots, n/2 \)  
- so loop iterates at least \( \lceil \frac{n}{2} \rceil \) times  
- so \( WC(n) \in \Omega(n) \)

Conclusion: \( WC(n) \in \Theta(n) \)
def pal_prefix(s: str) -> int:
    n = len(s)
    for p in range(n, 0, -1):  # p = n, n-1, ..., 1
        if is_palindrome(s[0:p]):
            return p

careful: generating s[0:p]
takes time $O(p)$, not constant!

Upper bound
- loop body takes time $O(p) \in O(n)$
- loop iterates at most $n$ times
- $WC(n) \in O(n^2)$.

NOTE: - ignore early termination
    - overestimate
    - simplify calculation as much as possible
    - only danger is that bound is not tight
    - check with lower bound

Lower bound

ROUGH WORK:
- difficulty: strings that make loop iterate many times are those that do not contain palindromes — at the extreme, $s =$ all diff. characters, is_palindrome(s[0:p]) takes time $O(1) - \text{total is } O(n)$.
- at the other extreme, $s =$ palindrome, for-loop iterates only once and is_palindrome takes time $O(n)$ — total $O(n)$. 
Consider $s = \left\lceil \frac{n}{2} \right\rceil$ a's, followed by $b$, followed by $\left\lfloor \frac{n}{2} \right\rfloor - 1$ a's

Example: $s = aaaaaabaaaaa$ ($n = 9$)

Then, `is_palindrome(s[0:p])` returns False for $p = n, n-1, \ldots, \left\lceil \frac{n}{2} \right\rceil + 1$ — this requires $p - \left\lfloor \frac{n}{2} \right\rfloor$ iterations of the loop inside `is_palindrome`.

Next, `is_palindrome(s[0:floor(n/2)])` returns True, in $\left\lceil \frac{n}{2} \right\rceil$ iterations.

Total number of iterations of the loop inside `is_palindrome` is:

$$\left\lceil \frac{n}{2} \right\rceil + \sum_{p=\left\lfloor \frac{n}{2} \right\rceil + 1}^{n} (p - \left\lfloor \frac{n}{2} \right\rceil) = \left\lceil \frac{n}{2} \right\rceil + \sum_{j=1}^{n-\left\lfloor \frac{n}{2} \right\rceil} j$$

$$= \left\lceil \frac{n}{2} \right\rceil + \sum_{j=1}^{\left\lfloor \frac{n}{2} \right\rceil} j$$

$$= \left\lceil \frac{n}{2} \right\rceil + \frac{\left\lfloor \frac{n}{2} \right\rceil(\left\lfloor \frac{n}{2} \right\rceil + 1)}{2} \in \Theta(n^2)$$

Conclusion: $WC(n) \in \Omega(n^2)$, so $WC(n) \in \Theta(n^2)$