Topics this week:

Hour 1 - Worst-case runtime analysis (will be on midterm)
Hour 2 - Worksheets: worst-case - has_duplicates / analyzing binary search
Hour 3 - Finish Hour 1 analysis / More runtime: twisty example from notes
Hour 4 - Worksheets: varying loop increments, helper functions, and more twisty analysis details,

CSC165 Student Questions / Announcements

• Problem Set 4 should be out later this week
• Information about Midterm 2 is posted on the course website

Go to menti.com and enter the code 64 20 49 (For Q+A during lecture)

CSC 165 Lecture 16

Worst-case runtime

Now we will consider cases where the runtime depends on the input size and the actual input value.
Review: Definitions from pages 104-105:

Upper bound on the worst-case runtime:

Lower bound on the worst-case runtime:
Example:

def is_palindrome(s: str) -> bool:
    # Return whether s is a palindrome.
    n = len(s)
    for i in range(n):
        if not s[i] == s[n - 1 - i]:
            return False
    return True

Analysis of the worst-case runtime of is_palindrome:

Part 1: Upper bound:
Part 2: Lower bound:
Example (page 108 course notes)

```python
def palindrome_prefix(s: str) -> int:
    n = len(s)
    for prefix_length in range(n, 0, -1):
        # Check whether s[0:prefix_length] is a palindrome
        is_palindrome = True
        for i in range(prefix_length):
            if s[i] != s[prefix_length - 1 - i]:
                is_palindrome = False
                break
    # If a palindrome prefix is found, return the current length.
    if is_palindrome:
        return prefix_length
```

Analysis of Worst-case runtime for palindrome_prefix:

Part 1 Upper bound:
Part 2 Lower bound:
CSC165 Student Questions / Announcements

• Problem Set 4 should be out now.
• Midterm 2 coming up soon.

Go to menti.com and enter the code 11 33 43 (For Q+A during lecture)

CSC 165 Lecture 17

Continuation from last lecture...
More Runtime Analysis

Recap of the 2 approaches - exact count / bounds

Twisty Example - a more complex increment example

(pg. 101 in course notes)

```python
def twisty(n: int) -> None:
    x = n
    while x > 1:
        if x % 2 == 0:
            x = x / 2
        else:
            x = 2*x - 2
```

Claim: For all values of x greater than 2, after two iterations the value decreases by at least 1.

Proof:
March 9-13

Analysis of twisty: