Last time:

**palindrome**: racecar, hannah, aibohphobia (fear of palindromes)

**Prefix**: 'sum', 'summation', 'mis', 'mispelt'

**Problem**: return the length of the longest prefix of a nonempty string $s$ that is a palindrome

def palindrome_prefix(s):
    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 is_palindrome:
            return prefix_length

**Problem**: describe $WC(n)$, palindrome prefix
\[ \max \{ \text{runtime of } \text{p-pon } s / \text{len}(s) = n \} \]

Our runtime plots might look like:

\[ g(n) \]
\[ g'(n) \]
\[ h(n) \]
\[ h'(n) \]

#basic operations

\[ n = \text{len}(s) \]

\[ WC(n) \in O(g(n)) \text{ means} \]

**Palindrome prefix**

\[ \exists c_0, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n > n_0 \Rightarrow \]

\[ \forall \text{inputs } s \text{ with } \text{len}(s) = n, \text{ runtime } p-p(s) \leq c_0 g(n) \]

\[ WC_{p-p}(n) \in \Omega(g'(n)) \text{ means} \]

\[ \exists c_1, n_1 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n > n_1, \Rightarrow \]

\[ \exists \text{input } s \text{ with } \text{len}(s) = n, \]

\[ c_1 g'(n) \leq \text{runtime } p-p(s) \]
\(g'(n)\) is a lower bound on \(\max\{\text{runtime}\}\)

**Last time:** \(\text{WC}_{\text{pp}}(n) \in O(n^2)\)

**Stopped:** \(\text{WC}_{\text{pp}}(n) \in \Omega(C?)\)

I would like \(n^2\) so then \(\text{WC}_{\text{pp}}(n) \in \Theta(n^2)\)

*Need to construct an example input, for each \(n\), for which the runtime is at least \(C \cdot n^2\) basic operations.*

*try *

*\(e.g., 1, S_n = 'aaa \ldots a'\)*

**Description:** \(\forall n \in \mathbb{N}, S_n\) is a string
\(\wedge \text{len}(S_n) = n\)
\(\wedge \forall i \in \text{range}(n), S[i] = 'a'\)

*The outer loop runs once.*
*The inner loop runs \(n\) times.*
*Return \(n\), runtime \(n\) basic operations.*

*try \(S_n = 'b aa \ldots a'\)*

**Describe:** \(\forall n \in \mathbb{N}, S_n\) is a string
\(\wedge \text{len}(S_n) = n\)
\(\wedge n > 0 \Rightarrow S[0] = 'b'\)
$s[i] = 'a'$

- The outer loop runs $n$ times
- Each inner loop runs once
- Return $1$, runtime $n$ basic operations

What property do we need from $S_n$ to get an $n^2$ runtime?

- Need both inner + outer loop to run $\#_{obt}$ times that depends on $n$

How? To get inner running can’t fail early

$S_n = \text{'aaa\ldots a'}$

- To get outer loop to keep running, $S$ can’t be a palindrome & failure point must depend on $n$. $\Rightarrow$ put a “b” in middle

\[ S_n = \text{'aaa\ldots a b a\ldots a'} \]

\[ \uparrow \left\lceil \frac{n}{2} \right\rceil + 1 \]

\[ \Rightarrow \text{not a palindrome} \]

An input family

Need $n$

s.t. $n \geq n$, $\left\lceil \frac{n}{2} \right\rceil + 1$

is valid index

It turns out that when index starts at 0, we don’t need the +1

\[ \Rightarrow \text{not a palindrome} \]

to make off centre

but +1 doesn’t
description:

The number of iterations will be roughly:

\[ \frac{n}{2} + \left( \frac{n}{2} - 1 \right) + \left( \frac{n}{2} - 2 \right) + \ldots + 1 + \frac{n}{2} \]

inner inner inner

\[ \frac{n}{2} \left( \frac{n}{2} + 1 \right) \]

\[ \frac{n^2}{8} + \frac{n}{4} \]

\[ \geq c_1 \cdot n^2 \]

\[ c_1 = \frac{1}{8} \]

So at least

\[ \frac{n^2}{2} = \frac{n^2}{8} + \frac{n}{4} \]

\[ \geq c_1 \cdot n^2 \]

\[ c_1 = \frac{1}{8} \]

\[
\therefore \text{WC}_{\text{P-P}}(n) \in \Omega(n^2)
\]

\[
\therefore \text{WC}_{\text{P-P}}(n) \in \Theta(n^2)
\]

What about \( BC_{\text{P-P}}(n) = \min \{ \text{runtimeons} \} \)

Want \( BC(n) \in \Theta(\text{some fun}) \)

find \( BC(n) \in O(L(n)) \)

BC(n) \( \in \Omega(L(n)) \)

\[ BC(n) \in \Omega(\text{some fun}) \]
Our runtime plots might look like:

\[ g(n) \]
\[ g'(n) \]
\[ h(n) \]
\[ h'(n) \]

\[ n = \text{len}(s) \]

\[ BC_{p_p}(n) \in \Omega(h(n)) \text{ means} \]
\[ \exists c_2, n_2 \in \mathbb{R}^+, \forall n \in \mathbb{N}, \ n > n_2 \Rightarrow \]
\[ \forall \text{inputs } s \text{ with } \text{len}(s) = n, \]
\[ c_2 h(n) \leq \text{runtime}_{p_p}(s) \]

\[ BC_{p_p}(n) \in O(h'(n)) \text{ means} \]
\[ \exists c_3, n_3 \in \mathbb{R}^+, \forall n \in \mathbb{N}, \ n > n_3 \Rightarrow \]
\[ \exists \text{input } s \text{ with } \text{len}(s) = n, \]
\[ \text{runtime}_{p_p}(s) \leq c_3 h'(n) \]
Notice how quantifiers switched from WC back to palindrome prefix

Consider \( \forall n \in \mathbb{N}, \text{len}(S_n) = n \land \forall i \in \text{range}(n), S[i] = \text{'}a\text{'} \)

then the \# basic operations is \( n+1 \leq 2n \) for \( n \geq 1 \)

\[ \rightarrow BC_{p-p}(n) \in O(n) \]

\[ \therefore \text{to show } BC_{p-p}(n) \in \Omega(n) \]

need to show at least \( c \cdot n \) basic steps are performed for all inputs

Let \( S \) be an arbitrary string,
let \( n = \text{len}(S) \)
and \( k = \text{palindrome-prefix}(S) \)
know \( 1 \leq k \leq n \)
The outer loop executes \( n-k \) times without detecting palindrome.
- In each case, the inner loop executes at least once.
- Then, on final iteration of outer loop.
  - The inner loop runs \( k \) times.

The total # of iterations is at least
\[
(n-k)(1+k) = n \\
\geq 1 \cdot n
\]

\[\therefore \quad BC_{p-p}(n) \in \Omega(n)\]

\[\therefore \quad BC_{p-p}(n) \in \Theta(n)\]

\[\therefore \quad \text{phew!}\]
Average Case Analysis

\[ \text{Runtime} \]

\[ n \]

Each point corresponds to a runtime for a particular input.

Put data from \( \text{Times}_{f,n} = \{ \text{runtime of } f(x) \mid x \text{ is } n \} \) into set.

\[ \text{WC}_{f}(n) = \max \text{Times}_{f,n} \]

\[ \text{BC}_{f}(n) = \min \text{Times}_{f,n} \]

Could be misleading. Runtime for a "typical" input could be quite different from extreme values today. A different flavor of runtime.
Given a finite set \( D = \{ d_i \} \):

- add up elements
- \( \% \) by \# of elements

\[
\text{Avg}_D = \frac{1}{|D|} \cdot \sum_{i=1}^{101} d_i
\]

to code:
Code to consider:

```python
def has_1(L: List[int]) -> bool:
    for i in range(len(L)):
        if L[i] == 1:
            return True
    return False
```

- Let \( n \) represent the length of \( L \).
- Runtime plot:

\[
\begin{align*}
\text{plot of } R \text{ is not in } L \\
L[0] == 1
\end{align*}
\]

- Observe \( WC_{\text{has-1}}(n) \in \Theta(n) \)
- \( BC_{\text{has-1}}(n) \in \Theta(1) \)
- What about \( A_{\text{has-1}}(n) \)?
\[ \text{Aug}_{\text{has-1}}(n) = \frac{1}{\text{# inputs}} \sum_{\text{inputs x}} \text{runtime of has-1}(x) \]

How many lists do int to size n?

on \( \infty \)

so we need to constrain the set of inputs under discussion

**def.** Let \( I \) be a finite set of \( f \mid n \) "allowable inputs" to \( f \) of size \( n \).

Then can compute

\[ \frac{1}{| I_{f,n} |} \sum_{x \in I_{f,n}} \text{runtime of } f(x) \]

**example** Consider \( I = \{ L \mid L \text{ is a permutation} \} \)

**def.** A permutation of the set \( \{1, 2, 3, \ldots, n\} \)
is an ordering of the elements into a list

e.g. \([1, 3, 2, \ldots, n]\) \([2, 1, 3, \ldots, n]\) appears exactly one.
Let \( S_n \) represent the set of all permutations of \( n \) \( \in \{1, 2, 3, \ldots, n\} \)

\[
S_1 = \{[1]\}, \quad S_2 = \{[1, 2], [2, 1]\}
\]

\[
S_3 = \{[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]\}
\]

\[
|S_n| = n! \quad \text{n choices for first item in list}
\]

\[
|S_n| = n \times (n-1) \times \ldots \times 2 \times 1 \\
= n!
\]

Now compute \( \text{Avg}(n) \) for \( I = S_n \)

By def's.\( \text{Avg}(n) = \frac{1}{|I|} \sum_{L \in I} \text{runtime } A \)

\# allowable inputs

\[\sum_{L \in I} \text{runtime } A \]

\[\text{sum overall allowable inputs} \]

\[\text{runtime } d \]
Now runtime \( \text{has}_1(n) = \# \text{ loop iterations performed} \)
\[
= (\text{index}_0 1 \in L) + 1
\]
\[
\text{Aug}_{\text{has}_1}(n) = \frac{1}{n!} \sum_{\text{LES}_n} [(\text{index}_0 1 \in L) + 1]
\]

\[
\text{split } S_n \text{ up based on index of the 1}
\]
\[
= \frac{1}{n!} \sum_{i=0}^{n-1} \left( \sum_{\text{LES}_n} \left[ (\text{index}_0 1 \in L) + 1 \right] \right)
\]
\[
\text{sum over all possible indices}
\]
\[
= \frac{1}{n!} \sum_{i=0}^{n-1} \left( \sum_{\text{LES}_n} (i+1) \right)
\]
\[
= \frac{1}{n!} \sum_{i=0}^{n-1} \left[ (i+1) \left( \sum_{\text{LES}_n} (i) \right) \right]
\]
\[
\text{# of lists with}
\]
How many lists have a 1 in posn \( i \)?

\[
L = [ a, q, q, \ldots, 1, q, \ldots, q, q, 1 ]
\]

So \( \text{Aug}_{\text{has-1}}(n) = \frac{1}{n!} \sum_{i=0}^{n-1} [(i+1)(n-i)!] \)

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

let \( i+1 = j \)

\[
= \frac{1}{n} \sum_{j=1}^{n} j
\]

\[
= \frac{1}{n} \cdot \frac{n(n+1)}{2}
\]

\[
= \frac{n+1}{2}
\]

Since this is an exact count, the average can be found through some basic operations.

\[ \text{Aug}_{\text{has-1}}(n) \in \Theta(n) \]

Note: \( \text{Aug}_f(n) \) depends on what input you
allow \( I \)

\[ \text{wc } (n) \in \Theta(n) \]

has 1

and \( \text{Avg } (n) \in \Theta(n) \)

\[ \text{has-1} \]

Tells us that average case runtime grows in same way as worst case, and so worst case is not atypical.

Will study further in CSC 263 where you will also include the likelihood of particular inputs.