Average Case

- WC is "pessimistic"
- For most algorithms, still representative, but for some, WC is not representative

Example:

```python
def search(L: List[int], x: int) -> bool:
    for item in L:
        if item == x:
            return True
    return False
```

\[ WC(n) \in \Theta(n) \]
\[ BC(n) \in \Theta(1) \]

Complication: need to define carefully all possible inputs — specifically, we need a set of inputs that captures all possible behaviours of the algorithm.

Approach: \( P_n = \{ \text{all permutations of } [1, 2, \ldots, n] \} \)

\( P_3 = \{ [1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1] \} \)

For search, define \( I_n = \{ (L, 1) \mid L \in P_n \} \)

(e.g., \( I_3 = \{ ([1, 2, 3], 1), ([1, 3, 2], 1) \ldots \} \))

\[ AC(n) = \left( \frac{1}{|I_n|} \sum_{(L, 1) \in I_n} RT(L, 1) \right) \]

\( (\text{Avg}(n) \text{ in course notes}) \)
scaling factor
for average
good value
of expression
for each elem.
in set $X_n$

$|X_n| = |P_n| = n! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1$

# possibilities: $[n, n-1, n-2, \ldots, 2, 1]$

$R_T(L, 1) = \# \text{ iterations of for-loop}$
$= \# \text{ times if-statement executes}$
$= 1 + \text{ index of value } 'i' \text{ in } L$

$[1, \ldots] [\ldots, 1, \ldots] \ldots [x, x, \ldots, x, 1, \ldots]$

$0 \leq i \leq c$

$A_C(n) = \frac{1}{n!} \sum_{(l, i) \in X_n} (1 + \text{ index of value } 'i' \text{ in } L)$

idea: introduce new variable $i$ to represent

$= \frac{1}{n!} \sum_{i=0}^{n-1} \sum_{\text{LeP}_n(L[i]=1}(1+i)$

how many lists in $P_n$

have $L[i] = 1$? $(n-1)!$

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

$= \frac{(n-1)!}{n!} \sum_{i=1}^{n} i = \frac{1}{n} \frac{n(n+1)}{2} = \sqrt{\frac{n+1}{2}}$
Choice of $I_n$?
- Does not account for $x \neq L$
- Does not account for repeated elements

Doing this "properly" for algo. search:

- $I_n = \{(1,2,...,n), x) \mid x = 0,1,2,...,n\}$
- Set up appropriate probability distribution over $I_n$
- Compute expected running time...