Average Case Analysis

Each $x$ corresponds to runtime for particular inputs.

Collect data from slice into a set of multiple inputs.

\[ \text{Times}_{f,n} = \{ \text{runtime of } f(x) \mid x \text{ is } n \} \]

\[ \text{WC}(n) = \max_f \text{Times}_{f,n} ; \quad \text{BC}(n) = \min_f \text{Times}_{f,n} \]

- Runtime for a "typical" input could be quite different than extreme values.

- To get another description of runtimes determine $\text{Avg}(n)$, the average of all values $f$ in $\text{Times}_{f,n}$.
Given \( D = \{ di \} \), \( \text{Avg}_D = \frac{1}{|D|} \sum_{i=1}^{\text{runtime of } \text{has}_1(L)} \),

Consider:

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

1 is not in list

### Runtime Analysis

- WC: \((n)\in \Theta(n)\), \(\text{has}_1\) \(\in \Theta(1)\)
- BC: \((n)\in \Theta(1)\), \(\text{has}_1\) \(\in \Theta(n)\)

What about \(\text{Avg}_\text{has}_1(n)\)?

\[
\text{Avg}_{\text{has}_1}(n) = \frac{1}{\# \text{ of inputs of size } n} \sum_{\text{inputs } L \in \text{size } n} \text{runtime of } \text{has}_1(L)
\]
How many lists of int of length $n$?

00 many.

& will need to constrain the set of inputs considered.

**def** Let $I_{f,n}$ be a finite set of allowable inputs to $f$ of size $n$.

Then compute: \[
\frac{1}{|I_{f,n}|} \sum_{L \in I_{f,n}} \text{runtime}_f(L)
\]

**example** Consider $I_{f,n} = \{L \mid L \text{ is a permutation of } \{1,2,\ldots,n\}\}$

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

e.g. $[1,2,3,\ldots,n]$ or $[2,1,3,\ldots,n]$
**def**: Let $S_n$ represent the set of all permutations of $\{1, 2, 3, \ldots, n\}$

e.g. $S_1 = \{1\}$
$S_2 = \{1, 2\}, \{2, 1\}$
$S_3 = \{1, 2, 3\}, \{1, 3, 2\}, \{2, 1, 3\}, \{2, 3, 1\}, \{3, 1, 2\}, \{3, 2, 1\}$

What is $|S_n|$?

$n$ choices for first item, then $n-1$ choices for second, and so on:

$\quad |S_n| = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1$

$\quad = n!$

Compute $\text{Avg}_{\text{has.1}}(n)$ for $I = S_n$.

$\text{Avg}_{\text{has.1}}(n) = \frac{1}{|I|} \sum_{\text{LE}I} \sum_{\text{runtime db has.1}(u)}$

$\uparrow$

# allowable inputs

$\uparrow$ sum over all allowable inputs.
\[= \frac{1}{n!} \sum_{LCS_n} \text{runtime}_b \text{ has } 1(L)\]

How to quantify \( \text{runtime}_b \text{ has } 1(L) \)?

\[= \# \text{ loop iterations performed} \]

\[= (\text{index of } 1 \text{ in } L) + 1 \quad \uparrow \text{ because Python indexing.} \]

\[\text{Avg}_{\text{has } 1}(n) = \frac{1}{n!} \sum_{i=0}^{n-1} \sum_{LCS_n \ L[i] \equiv 1} \sum_{(\text{index of } 1 \text{ in } L) + 1} \]

\[= \frac{1}{n!} \sum_{i=0}^{n-1} \sum_{LCS_n \ L[i] \equiv 1} (i + 1) \quad \uparrow \text{ sum over possible indices where } 1 \text{ could appear} \]

\[= \frac{1}{n!} \sum_{i=0}^{n-1} \sum_{LCS_n \ L[i] \equiv 1} (i + 1) \quad \uparrow \text{ independent of } L \text{ so can extract it} \]

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

\[= \# \text{ } bL \text{ with a 1 in pos } i \]
\[ L = \begin{bmatrix} 
\hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, \\
\hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, \\
\hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, \\
\hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, \\
\hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, \\
\hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, \\
\hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, \\
\hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, \\
\hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, \\
\hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, \\
\hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, \\
\hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, \\
\hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a}, & \hat{a} 
\end{bmatrix} \]

So \( \# \) of such \( cs + t \) is \((n-1)!
\]

\[
\text{Avg}_{\text{has. }_1}(n) = \frac{1}{n!} \sum_{i=0}^{n-1} [(i+1)(n-1)!]
\]

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

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

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

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

Since this is an exact count of operations

\[
: \text{ Avg}_{\text{has. }_1}(n) \in \Theta(n)
\]

Notes: \( \text{ Avg}_{f}(n) \) depends on what inputs \( f \) allows.

- for this example \( \text{ WC}(n) \in \Theta(n) \)
and \( \text{Avg}(n) \in \Theta(n) \)

\[ \Rightarrow \text{Worst case is not a typical for this algorithm.} \]

Ch 6  Graphs and Trees.

Schematics of a few graphs

\[ \text{def}^*: \text{A (undirected, simple) graph is a pair of finite sets } (V, E), \]
defined as follows:

- \( V \) is a finite set of objects, where each element \( v \in V \) is called a vertex of the graph.
\[ E \] is a set of pairs of objects, where each pair consists of two distinct vertices (i.e., \( u_1, u_2 \in V \)). Each pair, called an edge, order does not matter, so \((u_1, u_2)\) and \((u_2, u_1)\) represent the same edge.

We write \[ G = (V, E) \]

- G is the graph being described
- \( V \) is its vertex set
- \( E \) is its edge set