Think about what was last choice you made, and what options/choices you had.

- **Stairs**
  - Final choice: 1 step or 2 steps to get to final

$$T(n-2) \rightarrow \text{choice}^2 \xrightarrow{\text{Choice 1}} T(n-1)$$

- Use recursive expression to figure out prior # of options
- Use sum rule to combine.
  $$T(n-2) + T(n-1)$$
- Base case (2 cases here!)

- **Strings**
  - Last digit is 0, 1, ..., 9

\[ n \]
Master Method way to solve recurrence

Input: Array A of length n
Output: Max value in array

If A.length = 1, return A[1]
for i = 1 to n, do nothing

max 1 = max (1st half of A)
max 2 = max (2nd half of A)
return maximum [max1, max2]

Time complexity:

T(1) = O(1)
T(n) = 2T(n/2) + O(n)

Level of
Recursion

1
2
3
4
5

In box: amount
of work done at
this call not including
work done by recursive calls

O(n)
O(n)
O(n)
O(n)
O(n)
O(n)
O(n)
O(n)
O(n)

Size of input

n
n/2
n/4

Idea: count all work done in all boxes... that will be all the work.
Master Method

Way to solve certain recurrences

\[
T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d)
\]

If \( T(n) \leq C \) for \( n < n^* \)

\( a, b, d \) don't depend on \( n \)

Q: If \( T(n) \) is runtime of an algorithm,

What are \( a, b, d \) in words?

A: 
- \( a \): \# of recursive calls
- \( b \): factor by which problem shrinks in recursive call
- \( d \): characterizes extra work outside recursive call

Let's Add Up All Work

\( n \leftarrow \text{Problem size} \)

\( \frac{n}{s} \leftarrow \text{Input size} \)

\( \frac{n}{25} \leftarrow \)...

ex:
- \( a = 3 \)
- \( b = 5 \)
- \( d = 4 \)
Proof of Master Method

0. What is $F$ (in terms of $a$, $b$, $d$)?

A) $O(\log_b n)$  B) $O(\log d n)$  C) $O(n^{\log_b d})$  D) $O(b^{\log_d n})$

Because at each level, problem size is divided by $b$. $\log_b n$ is number of times $n$ can be divided by $b$ before reaching a constant.

$$\underbrace{\text{constant} \cdot b \cdot b \cdots b} = n$$

$$F = \frac{F}{\log_b n} \cdot \frac{F}{\log_b n} \cdot \cdots \cdot \frac{F}{\log_b n} = 1$$

Take $\log_b$ of both sides:

$$F = \log_b n + \text{constant}$$
Q. What is the total work done at level $K$ (outside of recursive calls & in terms of $a, b, c$)?

- $a^K$ subproblems at level $K$.
- Level $K$ subproblem size: \( \frac{n}{b^K} \)
- Work outside of recursive call required to solve 1 subproblem

\[ \Rightarrow \text{Total work} \quad a^K \left( \frac{n}{b^K} \right)^d = \left( \frac{a}{b^k} \right)^K n^d \]

Now we add up work done at all levels:

\[ \sum_{k=0}^{\log_b n} \left( \frac{a}{b^d} \right)^K n^d \]

\[ T(n) = n^d \left[ \sum_{k=0}^{\log_b n} \left( \frac{a}{b^d} \right)^K \right] \]

**Multiplicative Distributive Property**
Geometric Series:

\[
\sum_{k=0}^{F} r^k = \begin{cases} 
F+1 & \text{if } r = 1 \\
\frac{1-r^{F+1}}{1-r} & \text{otherwise}
\end{cases}
\]
2 cases:

- \( \frac{a}{b^d} = 1 \) \quad \rightarrow \quad n^d \sum_{k=0}^{\log_b n} \left( \frac{a}{b^d} \right)^k = O \left( n^d \log_b n \right)

- \( \frac{a}{b^d} \neq 1 \) \quad \rightarrow \quad n^d \sum_{k=0}^{\log_b n} \left( \frac{a}{b^d} \right)^k = n^d \frac{1 - \left( \frac{a}{b^d} \right)^{\log_b n}}{1 - \frac{a}{b^d}}

\uparrow \text{constant} = C

Look at:

2 cases

- \( \left( \frac{a}{b^d} \right) < 1 \) \quad \rightarrow \quad 1 - \left( \frac{a}{b^d} \right)^{\log_b n} = O(1)

- \( \left( \frac{a}{b^d} \right) > 1 \) \quad \rightarrow \quad \frac{1 - \left( \frac{a}{b^d} \right)^{\log_b n}}{1 - \frac{a}{b^d}} = O \left( \left( \frac{a}{b^d} \right)^{\log_b n} \right)
\[
\left( \frac{a}{b} \right)^{\log_b n} = \frac{a^{\log_b n}}{b^{\log_b n}} = \frac{a^{\log_b n}}{n^{\log_b b}} = \frac{a^{\log_b n}}{n^d} = \frac{n^{\log_b a}}{n^d} = O\left(n^{\log_b a}\right)
\]

\[
T(n) = aT\left(\frac{n}{b}\right) + O\left(n^d\right)
\]

\[
T(n) = \begin{cases} 
O(n^d \log n) & \text{if } a = b^d \\
O(n^d) & \text{if } a < b^d \\
O\left(n^{\log_b a}\right) & \text{if } a > b^d
\end{cases}
\]

Q: Interpret

- Balance between current work + recursive work.
- Run-time dominated by work outside recursive calls.
- Runtime dominated by work in bottom level of tree.