Sugar packet game:

nxn grid

n pink token on left border

n green tokens on top border
- Pink player moves one token one square right or jumps over a green token to go one-two squares right.
- Skip turn if no legal moves.
- Winner gets all their tokens off the grid first.
An algorithm for any 2-player game (almost)

state: location of pieces, current player, etc.

game tree: nodes are states,
dge from x to y if you can go from x to y in a single move.
Recursive def. For good/bad game states:
- A state is **good** if current player has already won or there is some move to put other player in a **bad** state.
- A state is **bad** if the current player lost or all moves put other player into a good state.

  **Good**: you can always win.

  **Bad**: you cannot win unless opponent makes a mistake.
**PLAY_ANY_GAME**\((X, \text{player})\):

if \(\text{player}\) has already won in state \(X\)
    return GOOD

if \(\text{player}\) has already lost in state \(X\)
    return BAD

for all legal moves \(X \leadsto Y\)
    if \(\text{PLAY_ANY_GAME}(Y, \neg\text{player}) = \text{BAD}\)
        return GOOD \(\langle X \leadsto Y \text{ is a good move} \rangle\)

return BAD \(\langle \text{There are no good moves} \rangle\)
Backtracking!

- you have some problem that requires you to make a sequence of decisions

- make one decision by examining each choice

  - ask Recursion Fairy to consistently make remaining decisions
I want to tell Recursion Fairy enough to make consistent decisions for each choice. I try to minimize the amount of info passed to R.F.
Rod Cutting

An optimization problem with many feasible/valid solutions. Each has a value; find the optimal (max or min value) solution.
Input: $n$, integer length of a rod we need to cut

$P[1..n]$ : $P[i] =$ how much we charge for a rod of length $i$

A solution: a sequence $<i_1, i_2, \ldots, i_k>$ of integer lengths $s.t. \sum_{j=1}^{k} i_j = n.$
Want to maximize total revenue \( \sum_{i,j} p_{i,j} \).

Ex: \( n = 4 \), \( p = \{1, 5, 8, 9\} \)

Opt. solution: \( \{2, 2\} \)

Value: \( 5 + 5 = 10 \)

Usually enough to compute optimal value.
Guess one piece size to sell & recursively cut up the length that remains.

\[ \text{RodCutting}(P[1..n], i): \]
\[
\text{if } i = 0 \\
\quad \text{return } 0
\]
\[
\text{maxRev } \leftarrow P[1] \quad \text{\texttt{\small \textbf{(We must sell something.)}}} \\
\text{for } j \leftarrow 2 \text{ to } i \\
\quad \text{optionalRev } \leftarrow P[j] + \text{RodCutting}(P[1..n], i - j) \\
\quad \text{if } \text{optionalRev} > \text{maxRev} \\
\quad \text{maxRev } \leftarrow \text{optionalRev}
\]
\text{return maxRev}

\( O(2^n) \) as written
Subset Sum

Input: A set $X$ of positive integers and a target integer $T$.

Output: Is there some subset of $X$ that sums to $T$.

Ex: $X = \{2, 5, 8\}$

$T = 10$ True ($2 + 8 = 10$)
\[ x = 2, 5, 83 \]
\[ T = 11 \quad \text{False} \]
Easy cases:

• \( T = 0 \), Answer is True. (Empty set sums to 0)
• \( T < 0 \) or \((T \neq 0 \text{ and } X \text{ is empty})\)
  Answer is False.

Otherwise, let \( x \) be any member of \( X \).
If there is a good subset summing to $T$... with $x$, everything else is a subset of $\mathbb{I} \setminus \exists x \exists 3 + \text{they add set subtraction up to } T - x$.

- without $x$, the whole subset comes from $\mathbb{I} \setminus \exists x \exists 3 \pm \text{sums to } T$. 
\[
\text{(Does any subset of } X[1..i] \text{ sum to } T?)
\]

\[
\text{SubsetSum}(X, i, T):
\]

- if \( T = 0 \)
  - return True
- else if \( T < 0 \) or \( i = 0 \)
  - return False
- else
  - \( \text{with} \leftarrow \text{SubsetSum}(X, i - 1, T - X[i]) \) \( \text{\{Recurse!\}} \)
  - \( \text{wout} \leftarrow \text{SubsetSum}(X, i - 1, T) \) \( \text{\{Recurse!\}} \)
  - return \((\text{with} \lor \text{wout})\)

---

- do we include \( X[i] \) in our subset?
- Is the a good subset with \( X[i] \) or
- without \( X[i] \)?

\( O(2^n) \) time (where \( n = 1 \times 1 \))