Sugar packet game:

n x n grid
- n pink token on left border
- n green tokens on top border

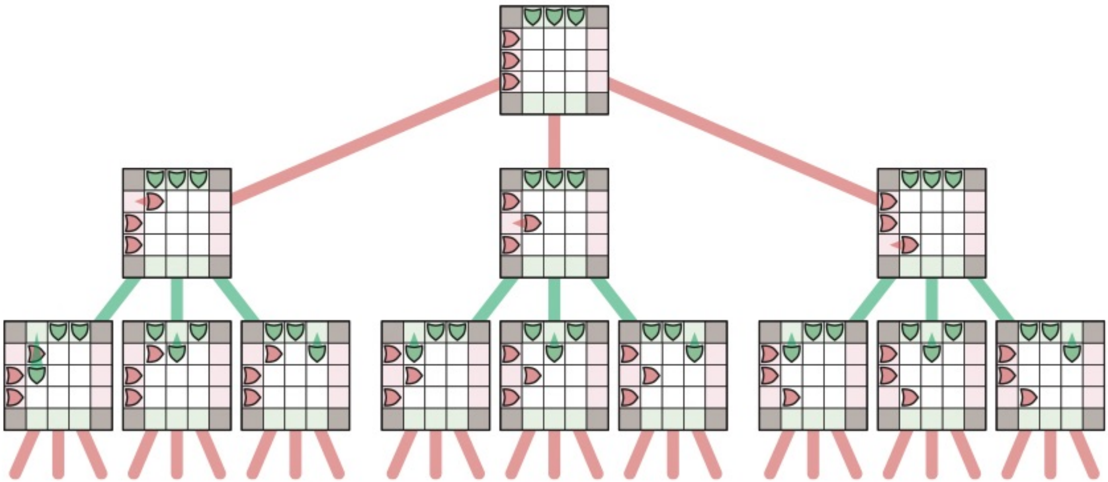
- pink player moves one token one square right or jumps over a green token to go ~~one~~ two squares right
- skip turn if no legal moves
- winner gets all their tokens off the grid first

An algorithm for <u>any</u> 2-player game (almost)

state: location of pieces, current player, etc.

game tree: nodes are states,

edge 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 <u>good</u> if current player has already won or there is <u>some</u> move to put other player in a <u>bad</u> state.

- a state is <u>bad</u> if 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 state

current player

```
PlayAnyGame(X, player):
    if player has already won in state X
        return Good
    if player has already lost in state X
        return Bad
    for all legal moves X ⇝ Y
        if PlayAnyGame(Y, ¬player) = Bad
            return Good        ⟨⟨X ⇝ Y is a good move⟩⟩
    return Bad                 ⟨⟨There are no good moves⟩⟩
```

Backtracking:

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

- make <u>one</u> decision by examining each choice
  - ask Recursion Fairy to consistenly make remaining decisions

- want to tell Recursion Fairy enough to make consistent decisions for each choice
  - try to minimize amount of info passed to R.F.

Rod Cutting

an <u>optimization</u> problem

- many feasible/valid solutions
- each has a <u>value</u>; 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 $\langle i_1, i_2, \ldots, i_k \rangle$ of integer lengths s.t. $\sum_{j=1}^{k} i_j = n$.

Want to maximize total revenue $\sum_{j=1}^{k} p[i_j]$.

Ex: $n = 4$ $p = \langle 1, 5, 8, 9 \rangle$

opt. solution: $\langle 2, 2 \rangle$

value $5 + 5 = 10$

Usually enough to compute optimal value.

Guess one piece size to sell & recursively cut up the length that remains.

← amount remaining

```
RodCutting(P[1 .. n], i):
    if i = 0
        return 0
                    + RodCutting(P, i-1)
    maxRev ← P[1]  ⟨⟨We must sell something.⟩⟩
    for j ← 2 to i
        optionalRev ← P[j] + RodCutting(P[1 .. n], i − j)
        if optionalRev > maxRev
            maxRev ← optionalRev
    return maxRev
```

($O(2^n)$ as written)

Subset Sum

Input: A set X of positive integers & a target integer T,

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

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

$T: 10$ True $(2+8=10)$

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

$T := 11$ False

Easy cases:

- $T = 0$. Answer is True. (empty set sums to 0)
- $T < 0$ or ($T \neq 0$ and X 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 $X \setminus \{x\}$ & they add up to $T - x$.

↑ set subtraction

- without x, the whole subset comes from $X \setminus \{x\}$ & sums to T.

⟨⟨Does any subset of $X[1..i]$ sum to T?⟩⟩

$\underline{\textsc{SubsetSum}(X, i, T)}$:
   if $T = 0$
      return TRUE
   else if $T < 0$ or $i = 0$
      return FALSE
   else
      $with \leftarrow \textsc{SubsetSum}(X, i-1, T - X[i])$ ⟨⟨Recurse!⟩⟩
      $wout \leftarrow \textsc{SubsetSum}(X, i-1, T)$ ⟨⟨Recurse!⟩⟩
      return $(with \vee wout)$

logical or

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 $n = |X|$)