CODELESS CODE LAB

JUMPING STAIRS

#dynamic_programming #recursion

#memoization #tabulation

By solving **J**UMPING **S**TAIRS,

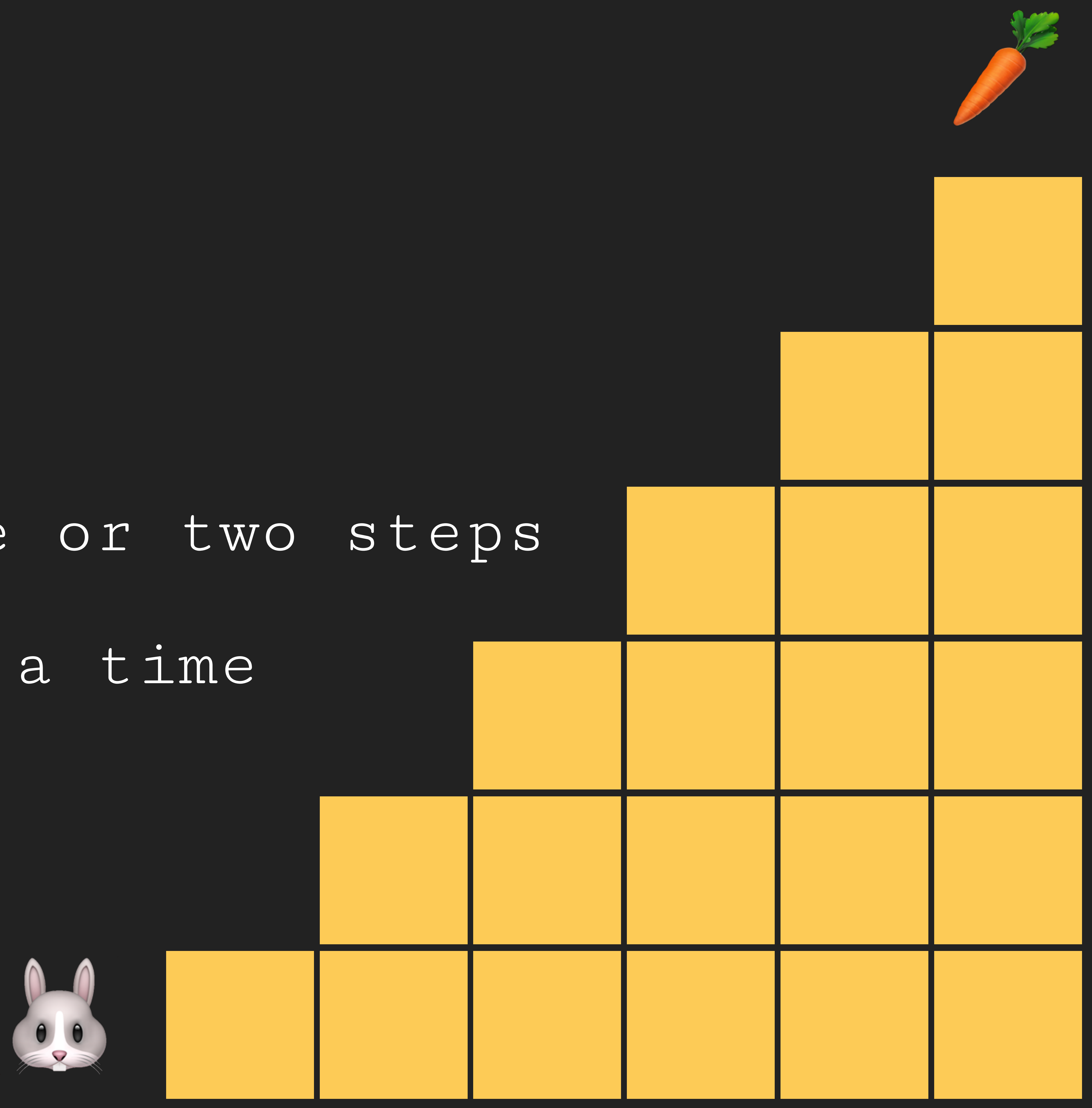
you will understand…

#dynamic_programming #recursion

#memoization #tabulation

이 강의는 한국어로도 제공됩니다.

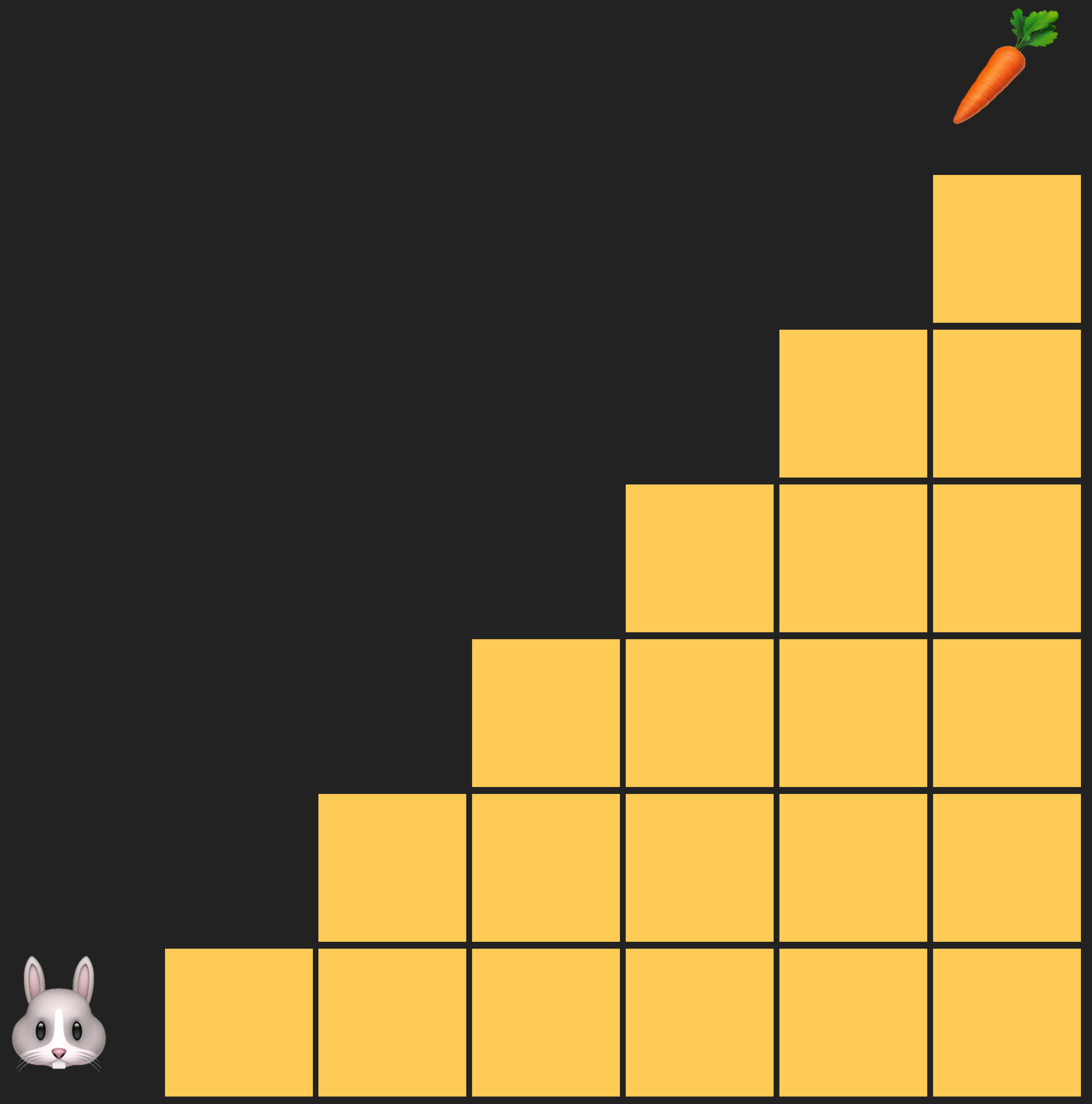
하단 설명란의 링크를 확인해주세요.



one or two steps

at a time

How many distinct ways of 🐰 → 🥕

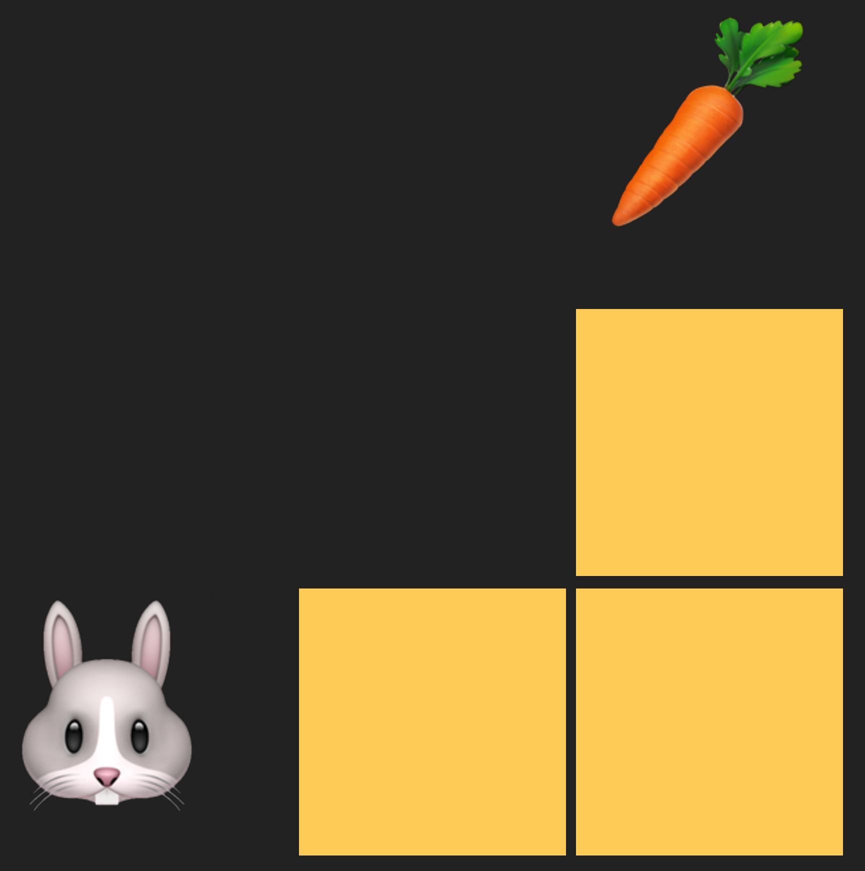


- 1, 1, 1, 1, 1, 1.
- 2, 1, 1, 1, 1.
- 2, 2, 1, 1.
- 2, 1, 2, 1.
- …
- Too complex to solve at once!

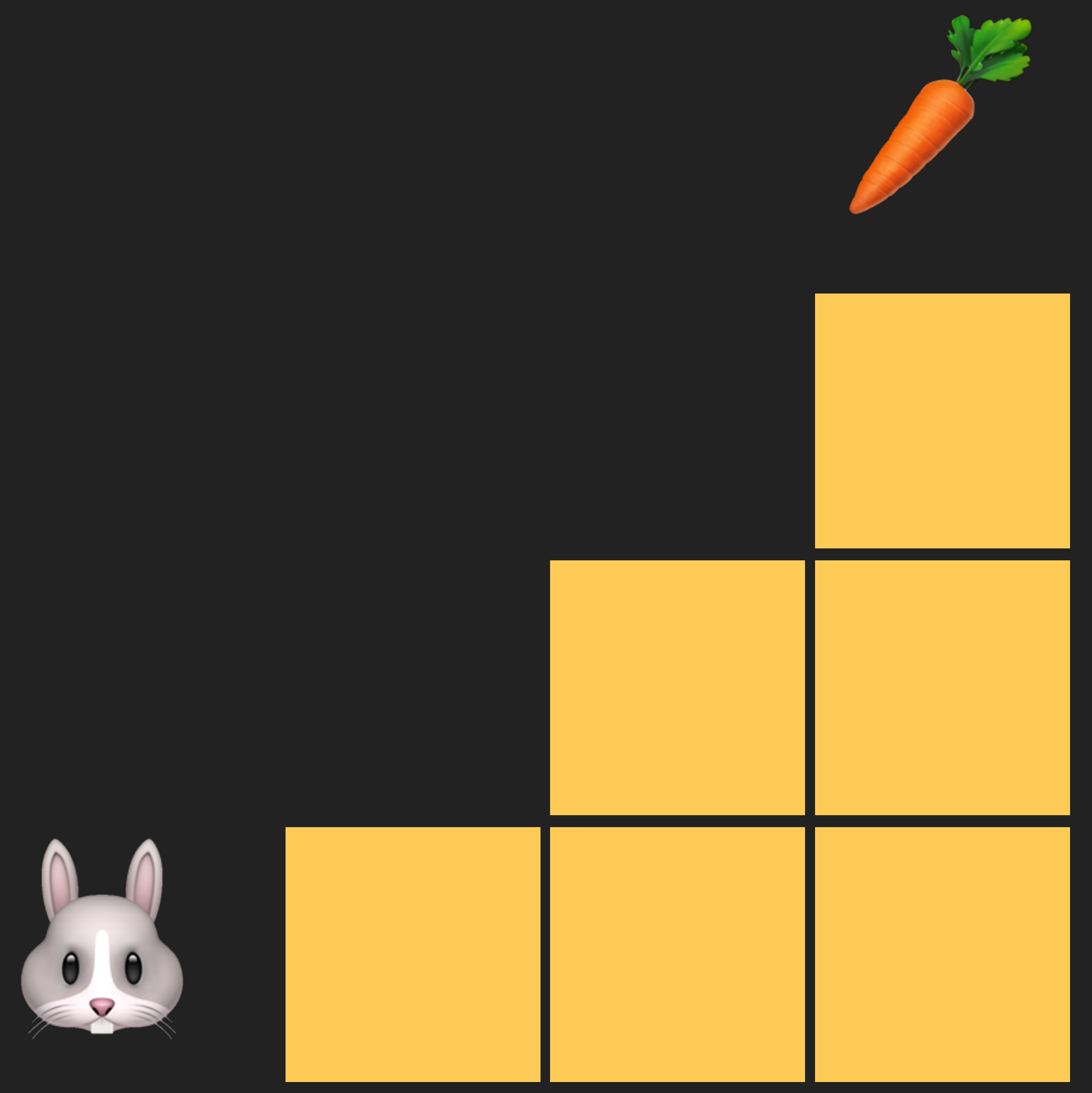
① 1

stair#	1	2	3	4	5	6
total#	1					

① 1, 1
② 2

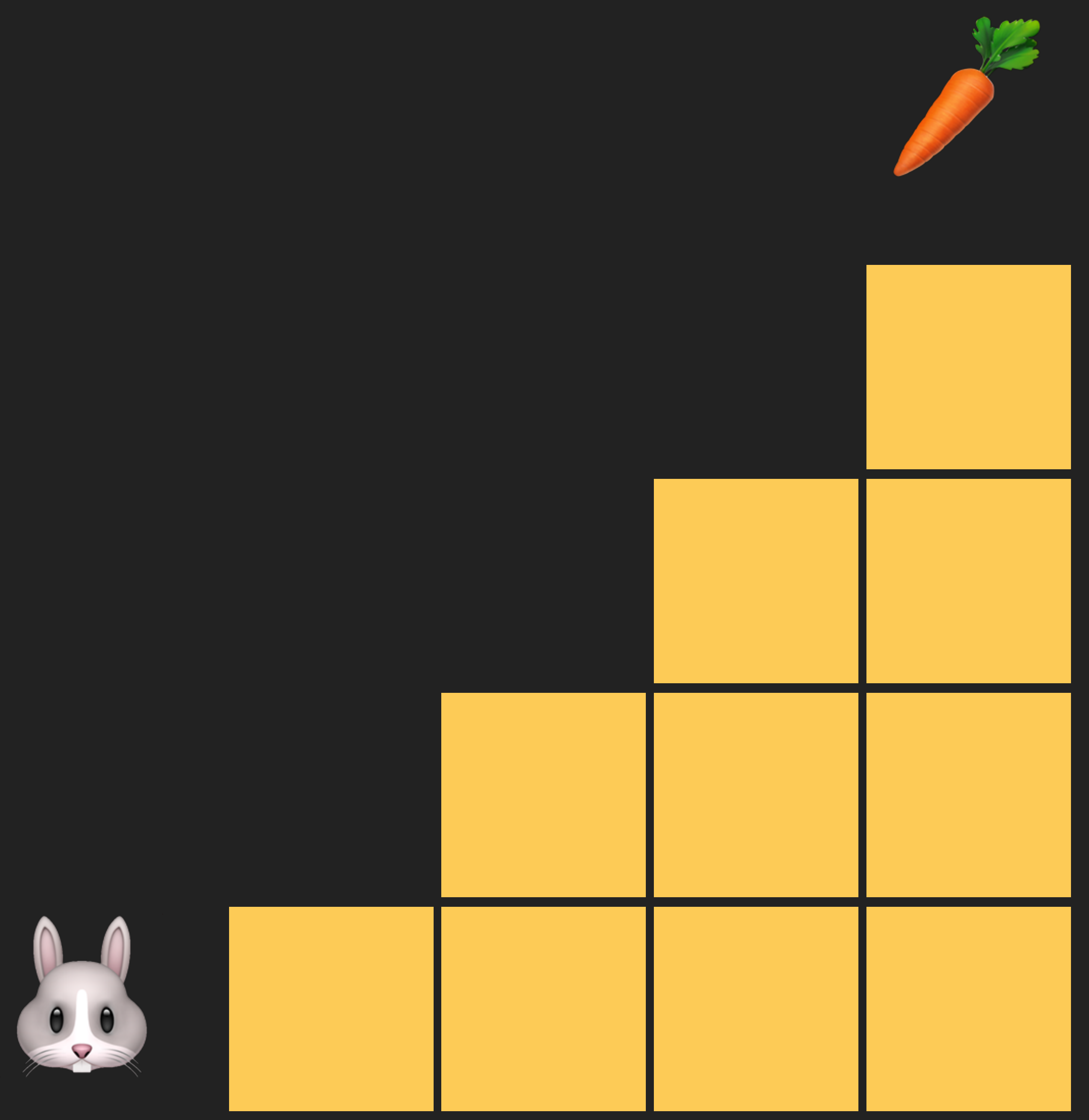


stair#	1	2	3	4	5	6
total#	1	2				



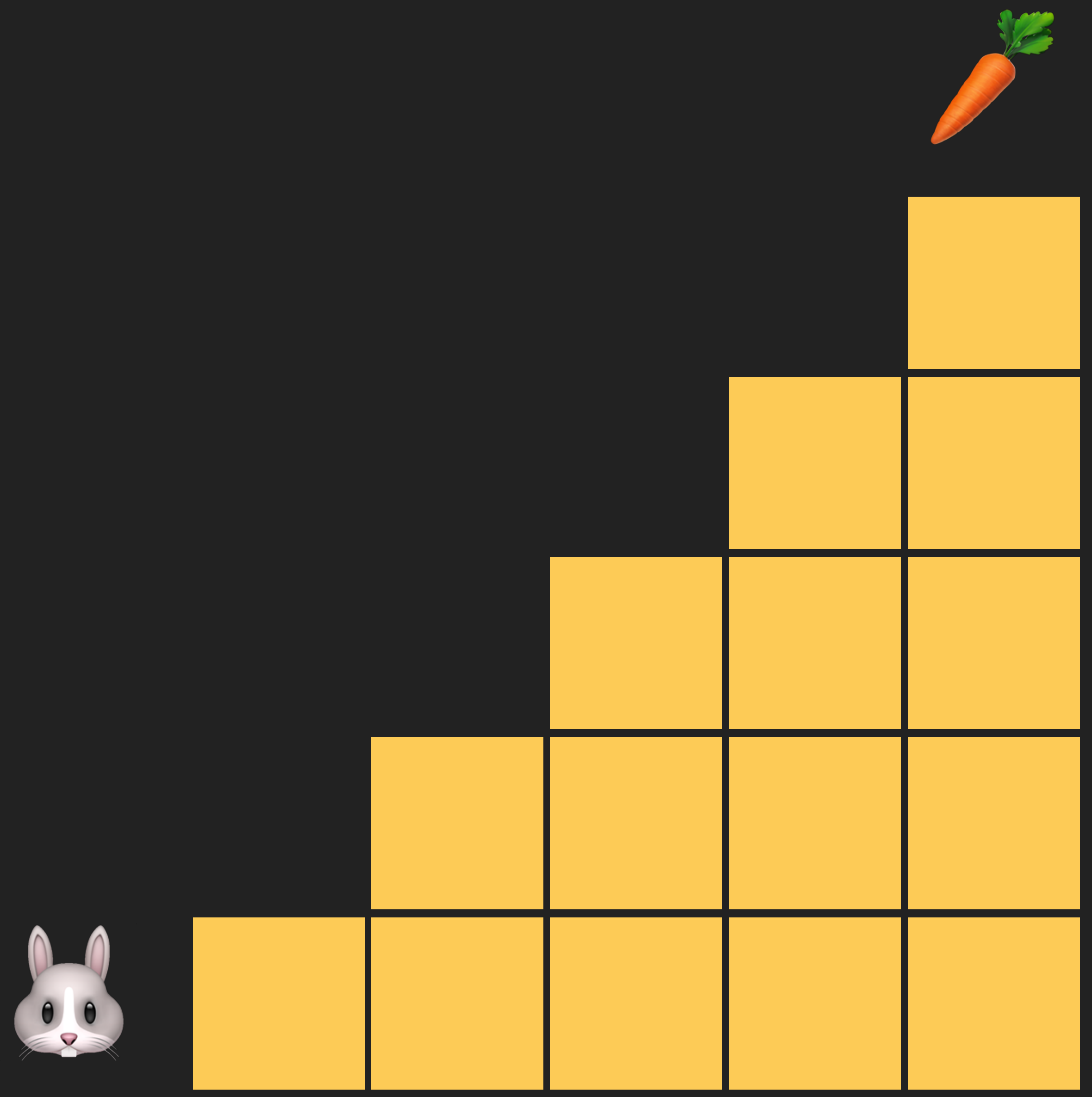
① 1, 1, 1
② 1, 2
③ 2, 1

stair#	1	2	3	4	5	6
total#	1	2	3			



① 1, 1, 1, 1
② 1, 1, 2
③ 1, 2, 1
④ 2, 1, 1
⑤ 2, 2

stair#	1	2	3	4	5	6
total#	1	2	3	5		



① 1, 1, 1, 1, 1
② 1, 1, 1, 2
③ 1, 1, 2, 1
④ 1, 2, 1, 1
⑤ 1, 2, 2
⑥ 2, 1, 1, 1
⑦ 2, 1, 2
⑧ 2, 2, 1

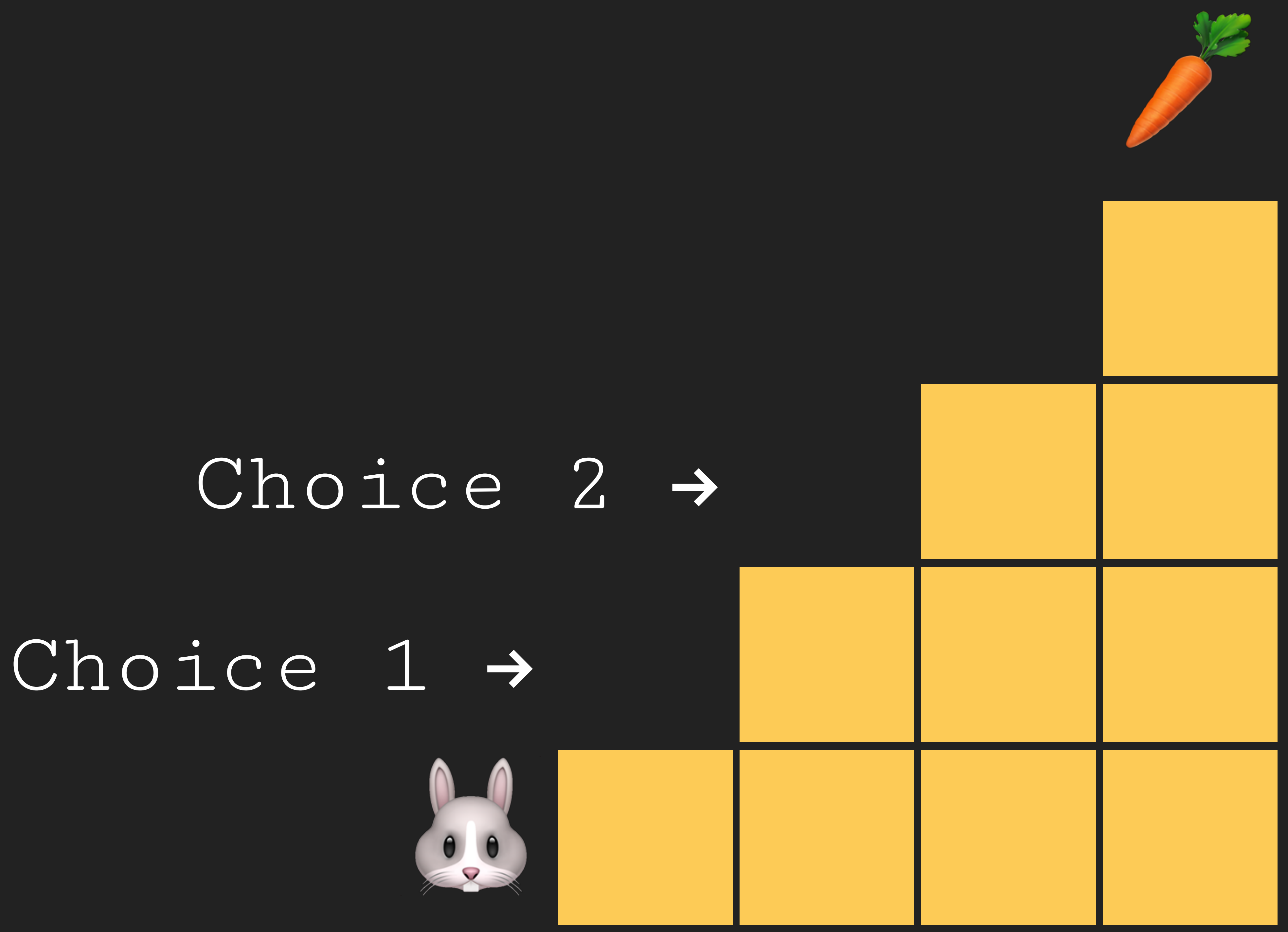
stair#	1	2	3	4	5	6
total#	1	2	3	5	8	

stair#	1	2	3	4	5	6
total#	1	2	3	5	8	

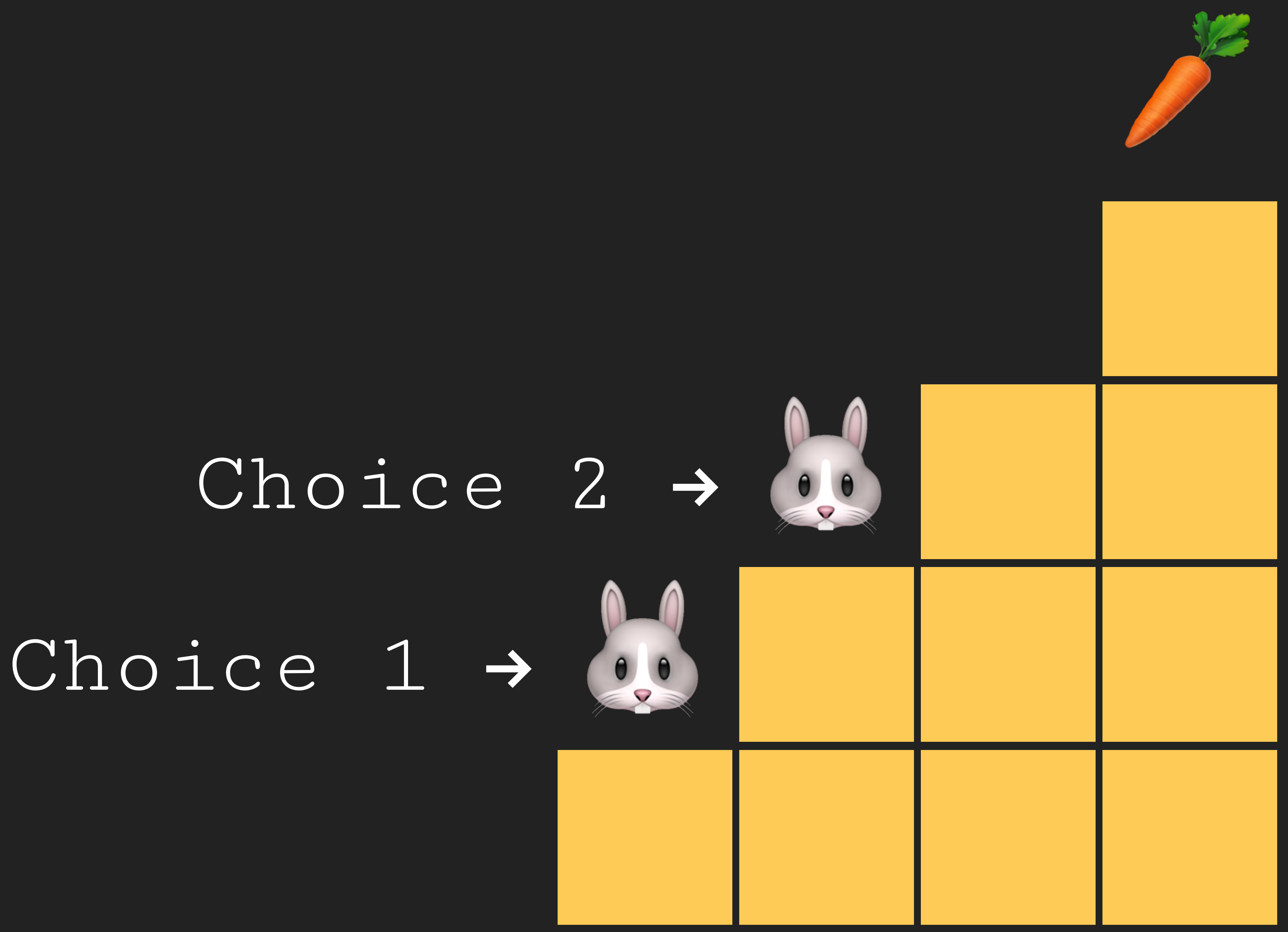
stair#	1	2	3	4	5	6
total#	1	2 +	3	5	8	

stair#	1	2	3	4	5	6
total#	1	2	3	+ 5	8	

stair#	1	2	3	4	5	6
total#	1	2	3	5	8	13



	1	2	3	4	5	6
total	1	2	3	5	8	13



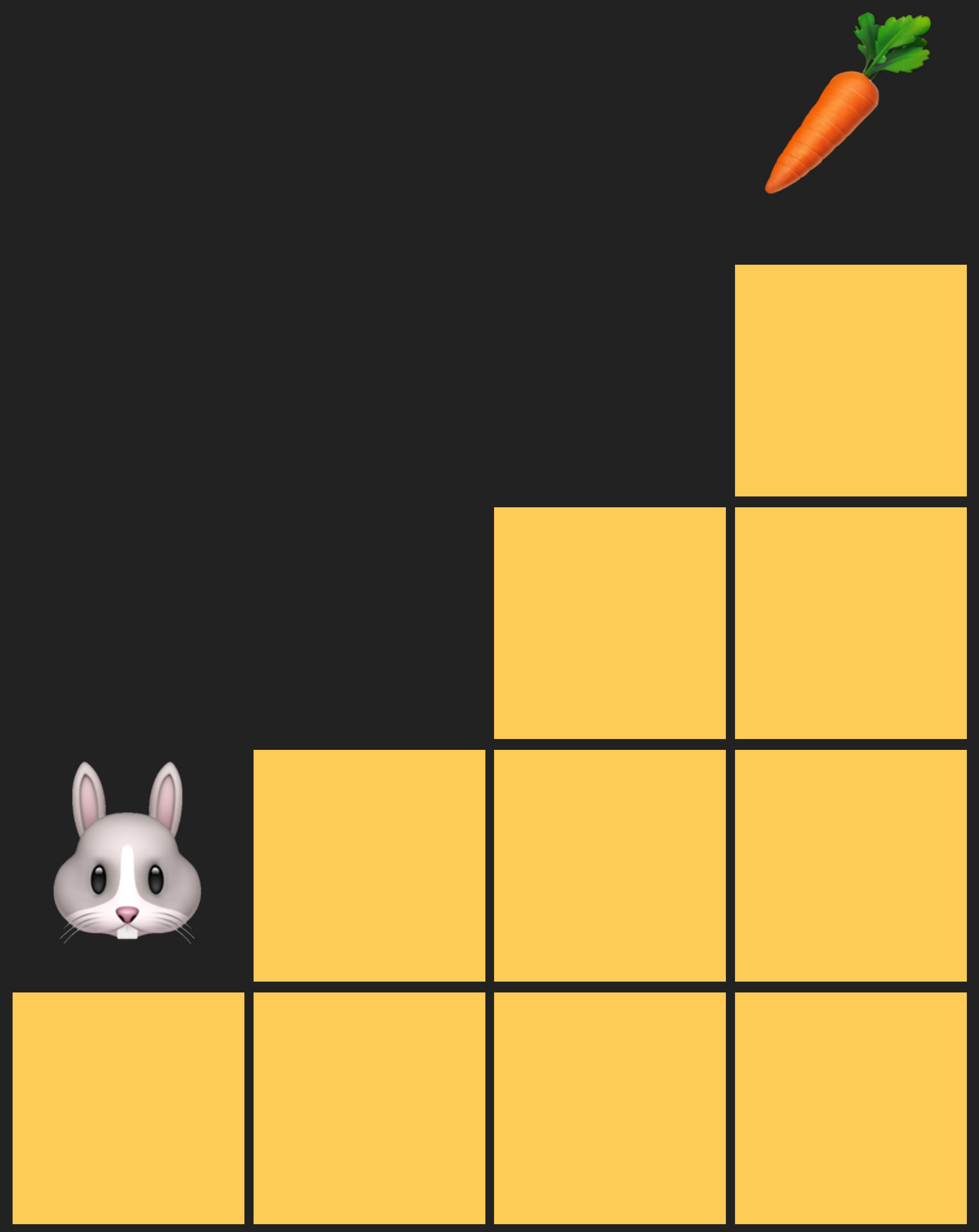
SP(4)

= Case# from Choice 1
+ Case# from Choice 2

Mutually Exclusive
Completely Exhaustive

	1	2	3	4	5	6
total	1	2	3	5	8	13

Choice 1 →

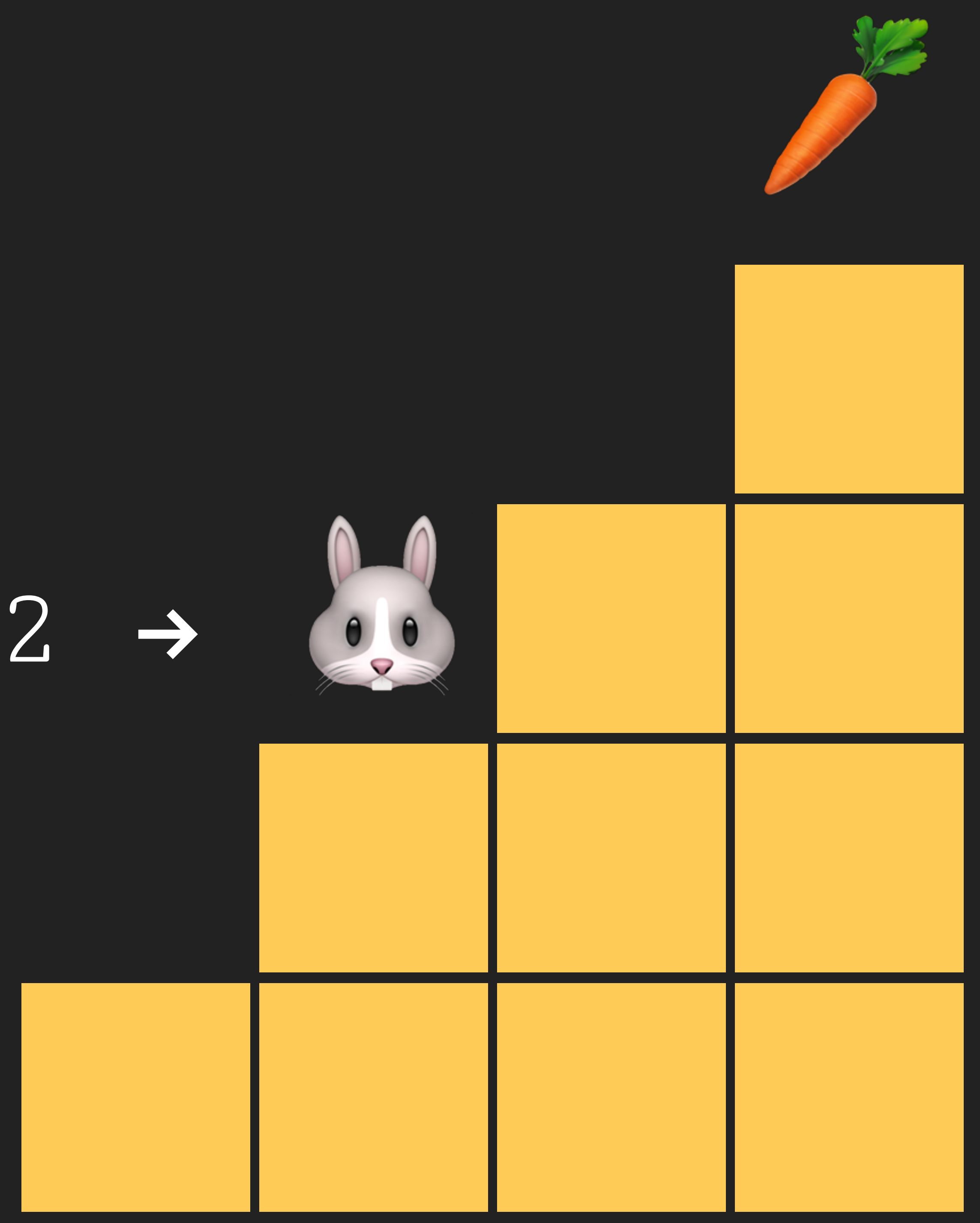


SP(4)

= Case# from Choice 1
+ Case# from Choice 2

	1	2	3	4	5	6
total	1	2	3	5	8	13

Choice 2 →



SP(4)

= SP(3)

+ SP(2) from Choice 2

	1	2	3	4	5	6
total	1	2	3	5	8	13

$$\underbrace{\text{SP}(N)}_{} = \overbrace{\text{SP}(N-1)}^{\text{Choice 1}} + \underbrace{\text{SP}(N-2)}_{\text{Choice 2}}$$

Generalize this to find SP(N)

Stairs	1	2	3	4	…	N
total#	1	2	3	5	…	SP(N)

```
Find SP(N):
     1. Find SP(N-1)
     2. Find SP(N-2)
     3. Add SP(N-1) and SP(N-2)
      → Value of SP(N)
```

```
Fine SP(N):
    1. SP(1) = 1
    2. SP(2) = 2

    If we still don't know SP(N),

    3. Find SP(N-1)
    4. Find SP(N-2)
    5. Add SP(N-1) and SP(N-2)
     → Value of SP(N)
```

$$
\begin{aligned}
SP(N) &= SP(N-1) + SP(N-2) \\
&= SP(N-2) + SP(N-3) + SP(N-3) + SP(N-4) \\
&= SP(N-3) + SP(N-4) + SP(N-4) + SP(N-5) \\
&\quad + SP(N-4) + SP(N-5) + SP(N-5) + SP(N-6)
\end{aligned}
$$

RECURSION

RECURSION

Method of solving a problem
where the solution depends on solutions to
smaller instances of the same problem

Source: Wikipedia

$$SP(N) = SP(N-1) + SP(N-2)$$

$$= SP(N-2) + SP(N-3) + SP(N-3) + SP(N-4)$$

$$= SP(N-3) + SP(N-4) + SP(N-4) + SP(N-5)$$
$$+ SP(N-4) + SP(N-5) + SP(N-5) + SP(N-6)$$

→ Same operation is repeated.

Dear Computer,

Remember the data of every SP(N).

For every request, check if you remember SP(N).

If you remember, do not calculate again.

If you don't, then calculate the value.

Then remember what you've calculated as well.

Sincerely,

Coder.

MEMOIZATION

Optimization method for Recursion

MEMOIZATION

optimization technique used primarily to speed up computer programs by storing the results of expensive function calls and returning the cached result when the same inputs occur again.

Source: Wikipedia

Stairs	1	2	3	4	5	6
total#	1	2	3	5	8	13

How to find the answer of SP(N):

1. IF N = 1, SP(N) = 1
2. IF N = 2, SP(N) = 2

If we still don't know the answer,

3. Find SP(3) with SP(1) and SP(2)
4. Find SP(4) with SP(2) and SP(3)

… Repeat this until we reach SP(N)

→ Value of SP(N)

$$SP(1) = 1$$

$$SP(2) = 2$$

$$SP(3) = SP(1) + SP(2)$$

$$SP(4) = SP(2) + SP(3)$$

$$SP(5) = SP(3) + SP(4)$$

...

$$SP(N) = SP(N-1) + SP(N-2)$$

TABULATION

TABULATION

Method of solving all related sub-problems first, typically by filling up a table.

Based on the results in the table, the solution to the original problem is then computed.

Source: StackOverflow

	Recursion & Memoization	Tabulation
Code	Easy (when implemented correctly)	Difficult
Speed	Slow (results may vary)	Fast (results may vary)
SubP	Better solving subproblems	Better solving the entire problem
Table	filled on demand	filled at once

DYNAMIC PROGRAMMING

DYNAMIC PROGRAMMING

Method of simplifying a complicated problem by breaking it down into simpler sub-problems.

→ Recursion (Optimized with Memoization)

→ Tabulation

Source: Wikipedia

CODELESS CODE LAB

SUNGHYUN CHO

Resources and Credits
are written in the description.

Special thanks to Mr. Park, Yong Sung,
for providing the initial idea.