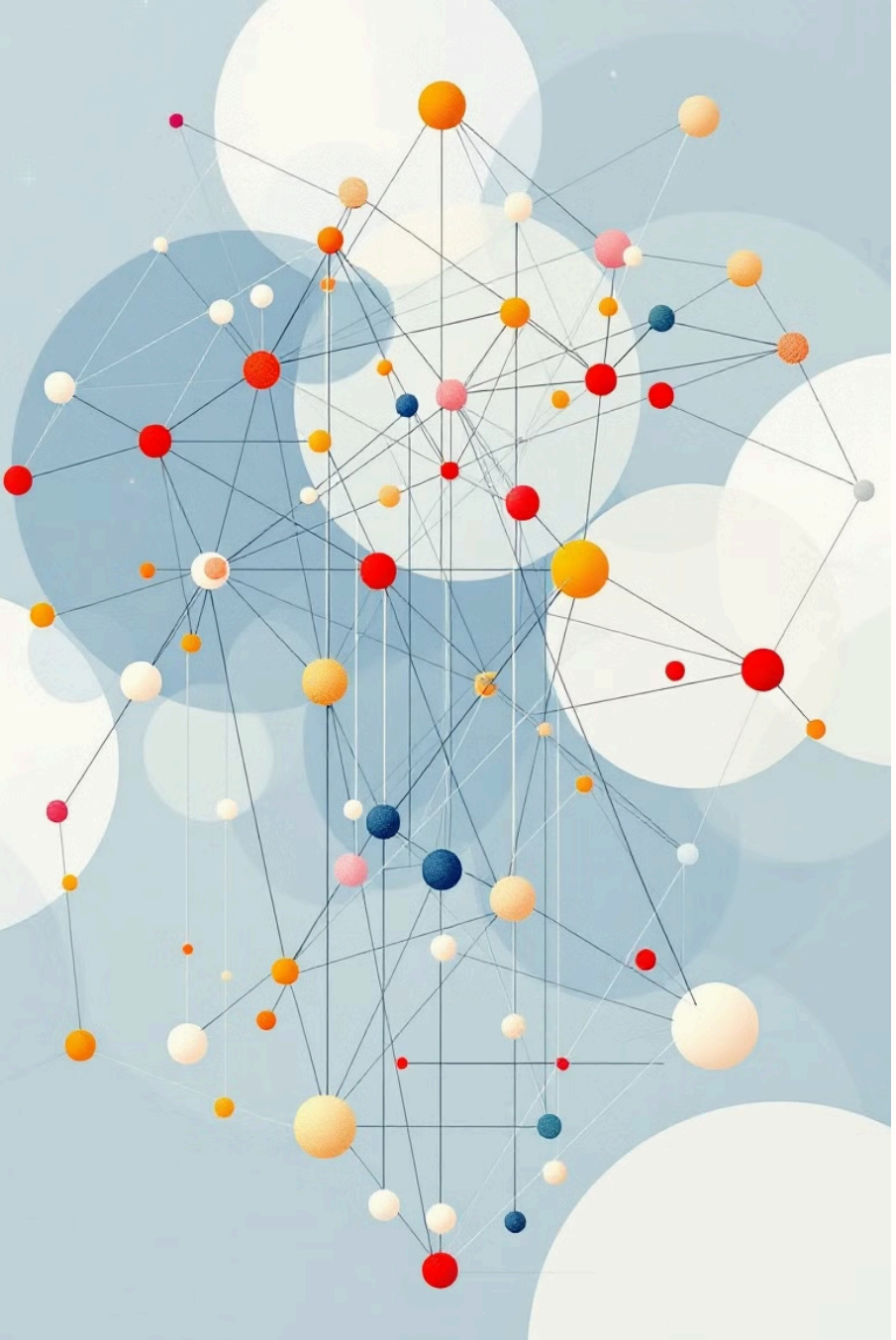


Dinaminis programavimas (DP)

Algoritminė technika sudėtingoms problemoms spręsti

AG **by Andrej Gorbatniov**

Kas yra dinaminis programavimas?

Algoritminė technika

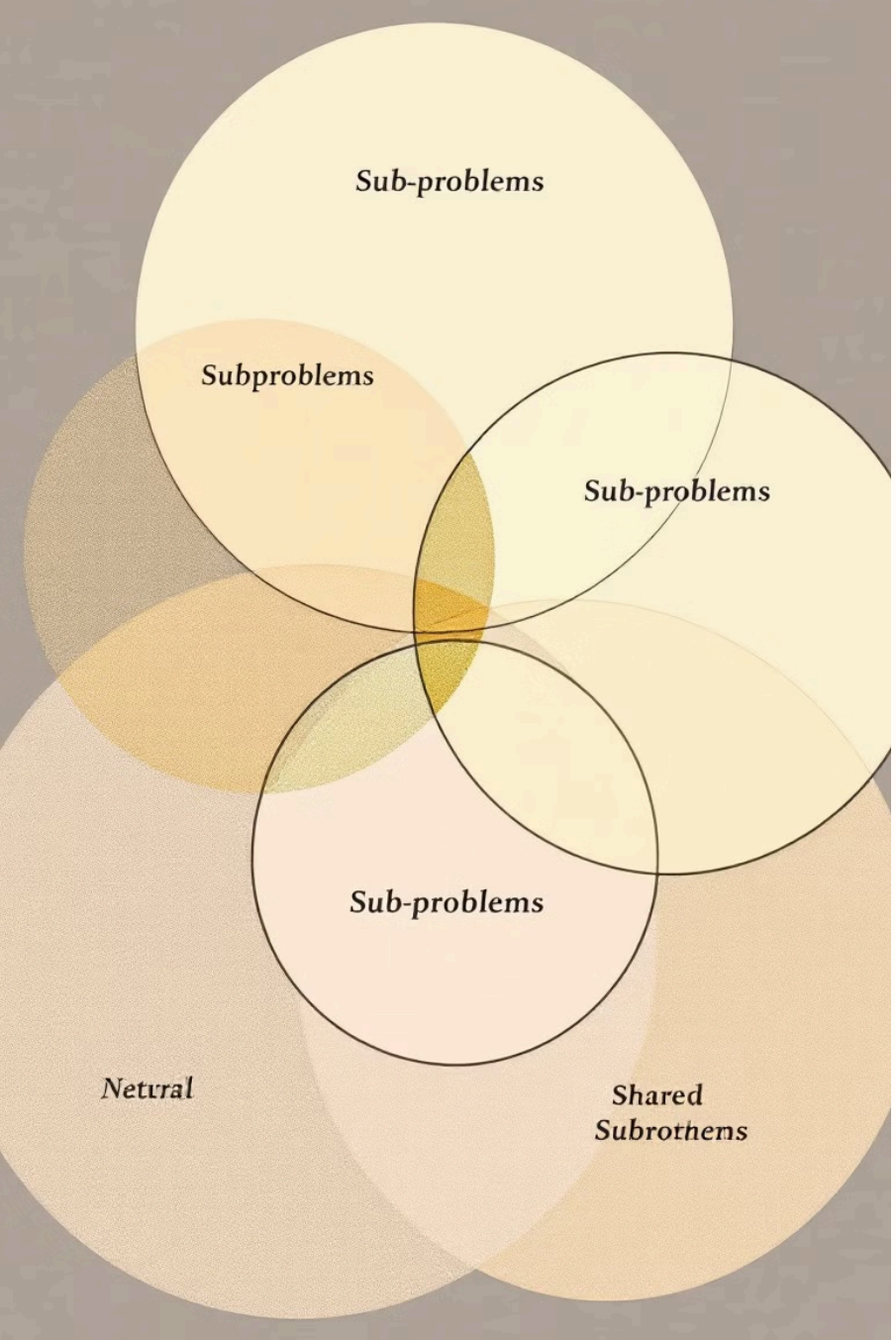
Išskaido sudėtingas problemas į paprastesnes dalis

Pasikartojančios dalys

Sprendimo dalys persidengia ir išsaugomos

Metodai

Memoizacija (iš viršaus) arba bottom-up (iš apačios)



Kada verta naudoti dinaminį programavimą?

Pasikartojančios dalinės problemos (overlapping)

Sprendimas priklauso nuo tų pačių smulkesnių dalių

Optimali sandara

Optimalus sprendimas sudėtas iš optimalių sprendimų dalims

Greedy algoritmai – kas tai?

1

🔍 Kas yra Greedy algoritmas?

Kiekviename žingsnyje renkamės lokaliai geriausią sprendimą.

2

Pasirinkimai

Pasirinkimai daromi vietoje, be žvilgsnio į ateitį.

3

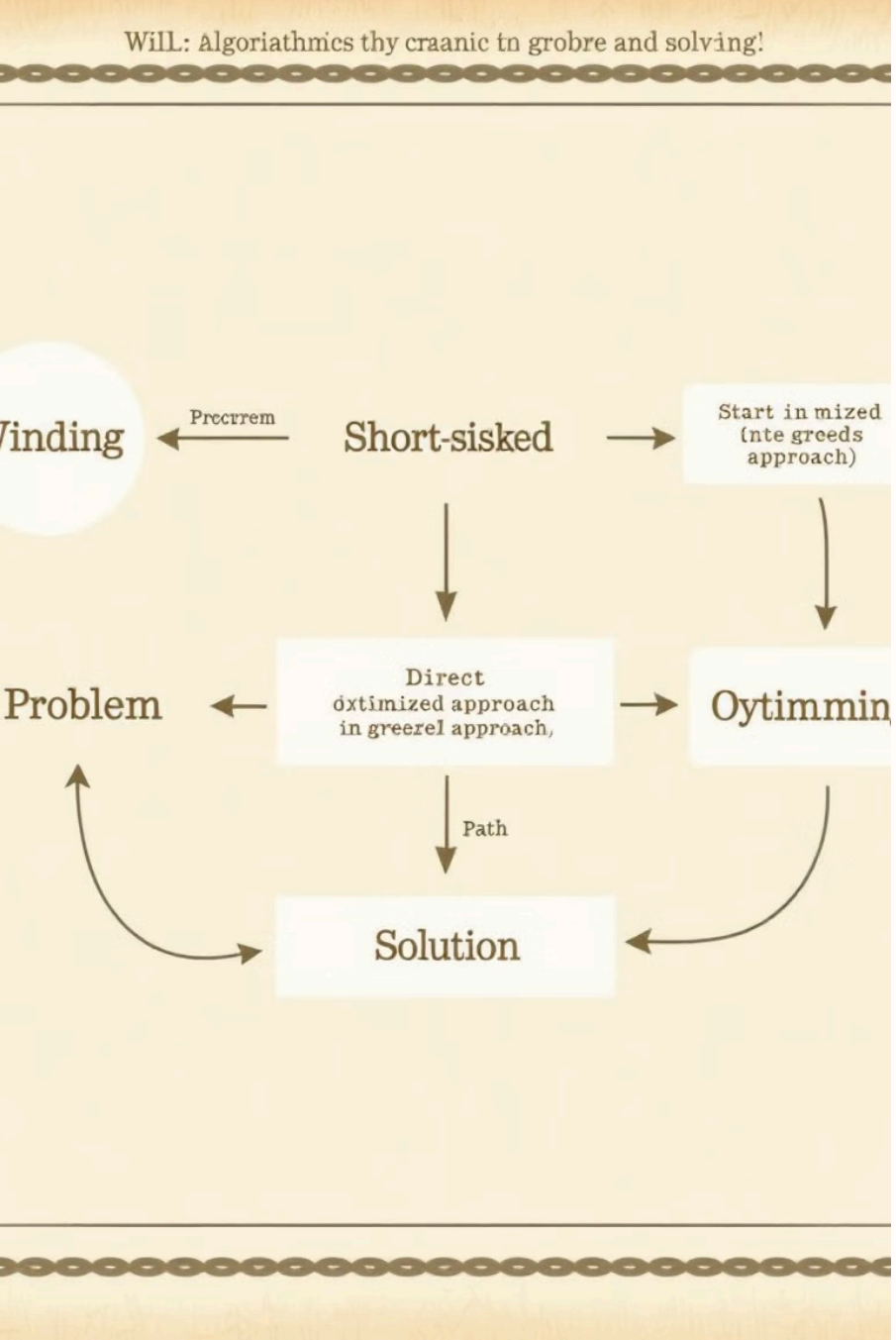
Nėra grįžimo

Nėra grįžimo atgal ar peržiūros, kaip rekursijoje.

4

Sprendimas

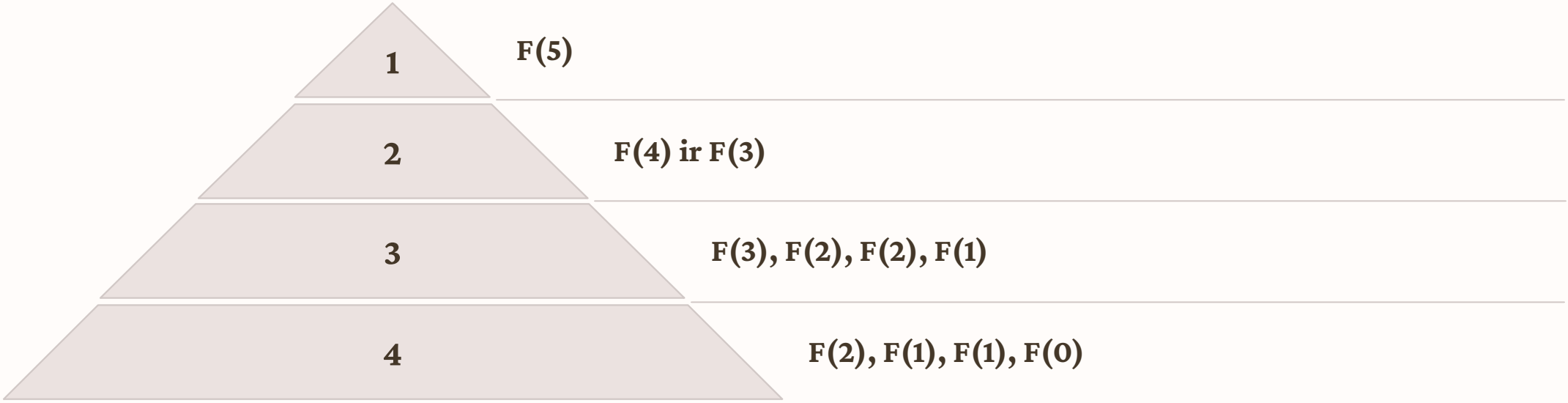
Sprendimas formuojamas palaipsniui.



Rekursija vs. Greedy vs. DP

Metodas	Kaip veikia	Kada naudoti
Rekursija	Skirsto problemą į mažesnes dalis	Kai nėra pasikartojimo
Greedy	Renkasi lokaliai geriausią	Kai užtikrinamas globalus optimalumas
DP	Išsaugo tarpinius rezultatus	Kai yra persidengiančios dalys

Rekursinis Fibonačio medis: koks neefektyvus!



Funkcija: F(n) = F(n-1) + F(n-2)

Pastebėkite, kiek kartų skaičiuojamas tas pats F(2), F(3)

Tas pats Fibonačio uždavinys su DP

Saugome rezultatus

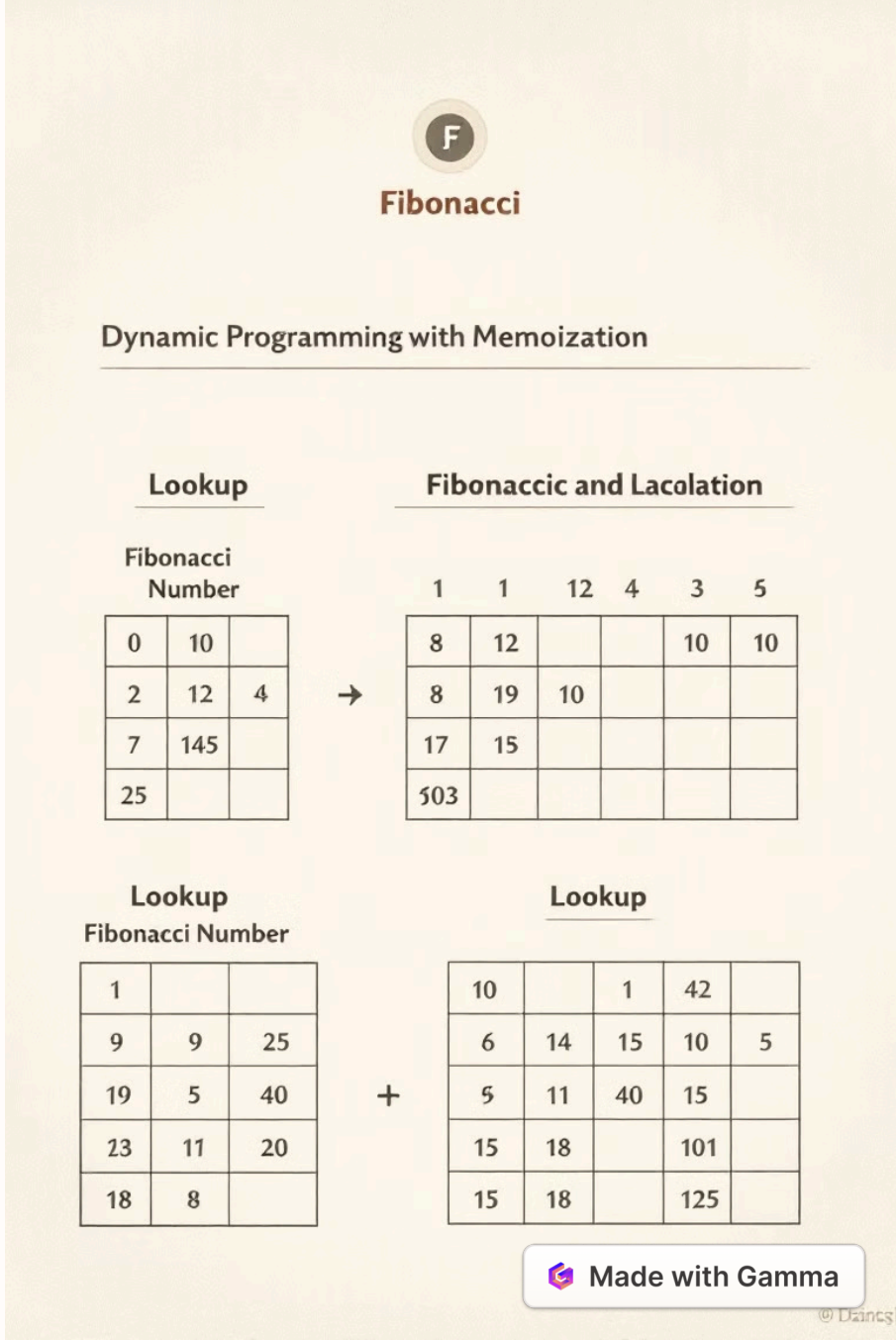
Kiekvienas F(n) skaičiuojamas tik kartą

F(0) → 0, F(1) → 1

Baziniai atvejai

F(2) = F(1) + F(0)

Pildome lentelę nuosekliai



Du DP sprendimo būdai

Memoizacija (rekursinis iš viršaus žemyn)

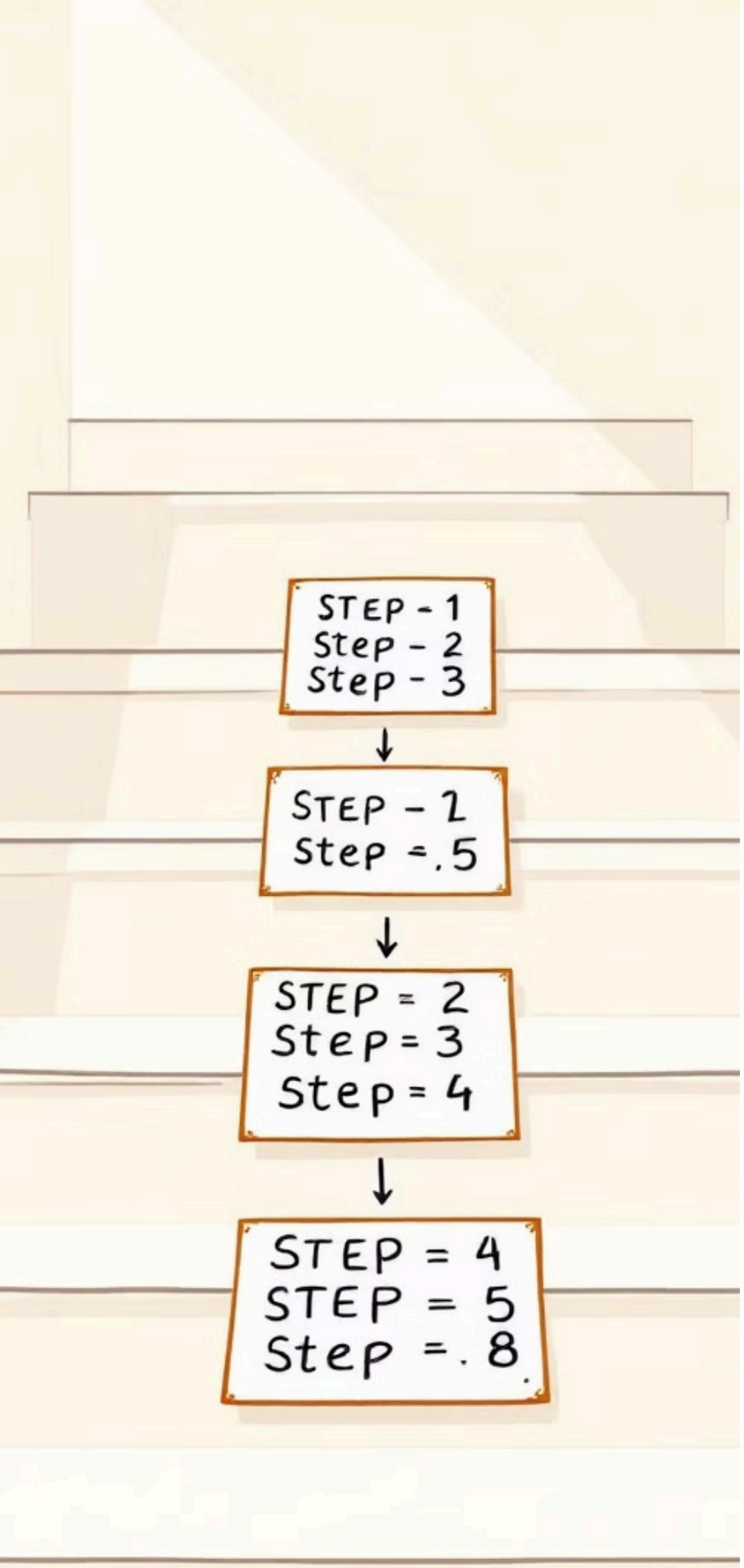
Memoizacija – tarsi **kelias per medį**, kur saugomi jau aplankyti mazgai

- Lengviau rašyti
- Artima rekursijai
- Daug rekursijos gylio

Bottom-Up (iš apačios į viršų)

Bottom-Up – tarsi **plytelių klojimas nuo grindų iki lubų**

- Nenaudoja rekursijos
- Paprasta optimizuoti
- Reikia žinoti eigą iš anksto



DP lentelės pavyzdys: Kiek būdų užlipti laiptais?

1

dp[0]

Vienas būdas niekur nelipti

1

dp[1]

Vienas būdas užlipti ant pirmo laiptelio

2

dp[2]

dp[1] + dp[0] = 2

8

dp[5]

dp[4] + dp[3] = 8

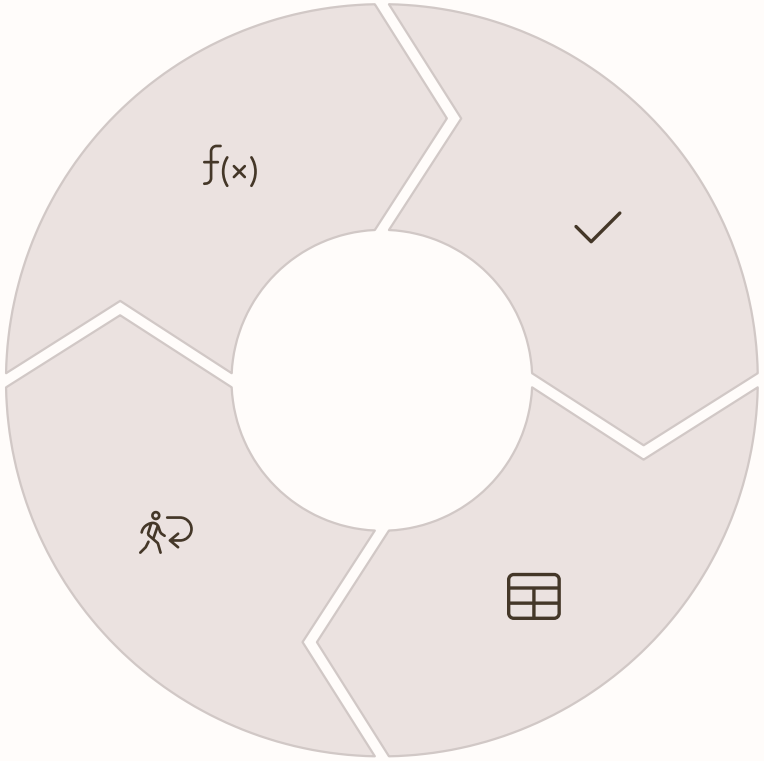
Climbing Stairs – Bottom-Up DP pseudokodas

Funkcija

countWays(n)

Bazinis atvejis

if n == 0 or n == 1: return 1



Ciklas

for i from 2 to n: dp[i] = dp[i-1] + dp[i-2]

Lentelė

dp = array of size n+1