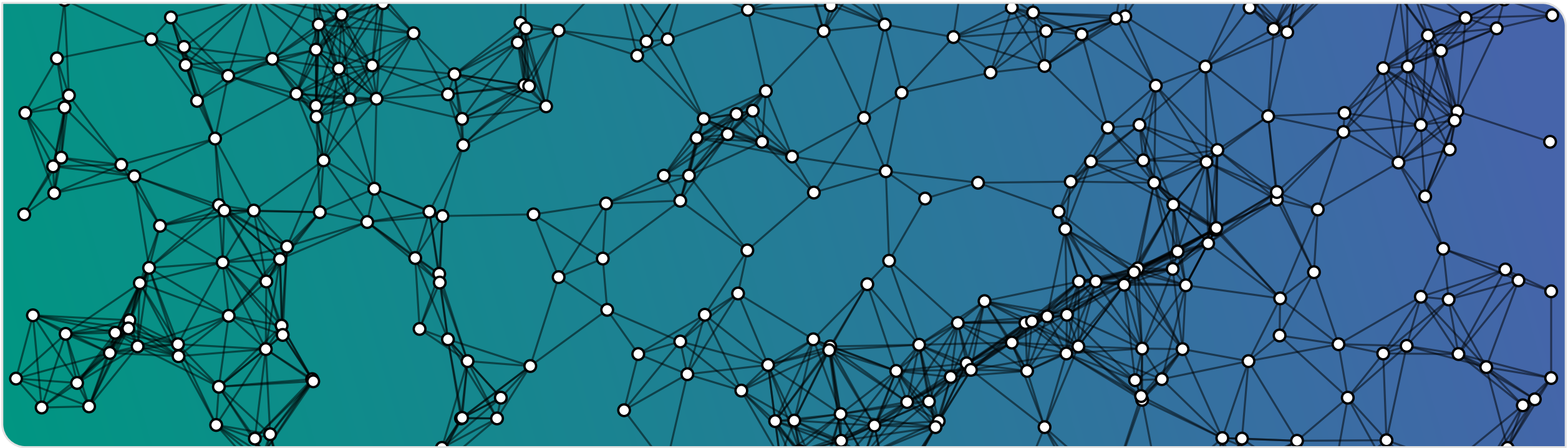
Karlsruher Institut für Technologie

Algorithmen 1

Übung 7
Dynamische Programmierung und Amortisierte Analyse



Dynamische Programmierung

- Methode zum Lösen von Optimierungsproblemen

Dynamische Programmierung

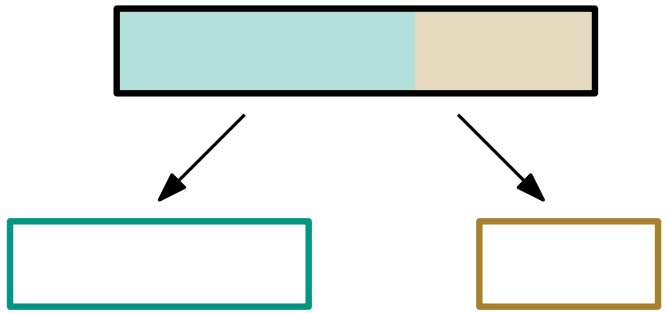
- Methode zum Lösen von Optimierungsproblemen
- optimale Lösung

Dynamische Programmierung

- Methode zum Lösen von Optimierungsproblemen
- optimale Lösung $\leftarrow$ optimale Teillösung + optimale Teillösung

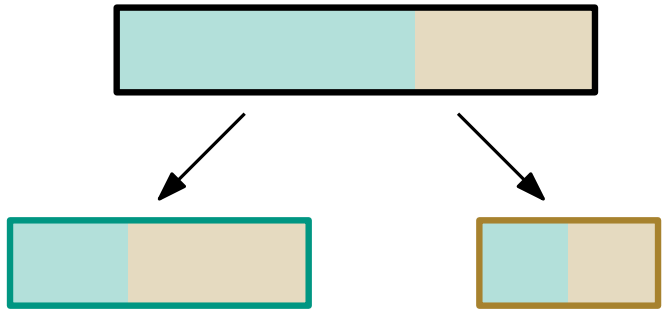
Dynamische Programmierung

- Methode zum Lösen von Optimierungsproblemen
- optimale Lösung $\leftarrow$ optimale Teillösung + optimale Teillösung



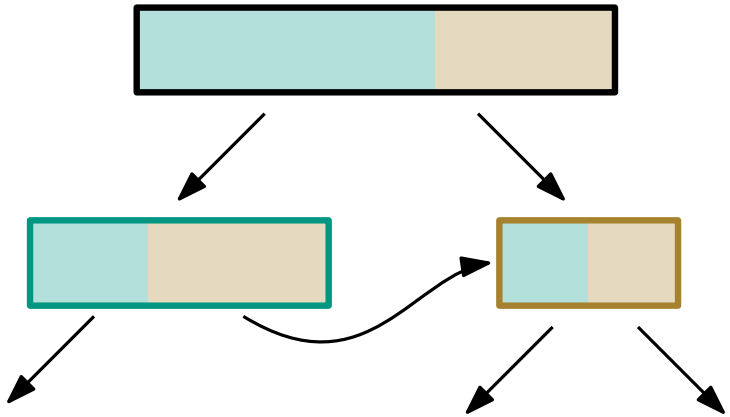
Dynamische Programmierung

- Methode zum Lösen von Optimierungsproblemen
- optimale Lösung $\leftarrow$ optimale Teillösung + optimale Teillösung
- bestimme Rekurrenz



Dynamische Programmierung

- Methode zum Lösen von Optimierungsproblemen
- optimale Lösung $\leftarrow$ optimale Teillösung + optimale Teillösung
- bestimme Rekurrenz
- verwende optimale Teillösungen wieder



Paletten-Hopping

Problemstellung

Paletten-Hopping

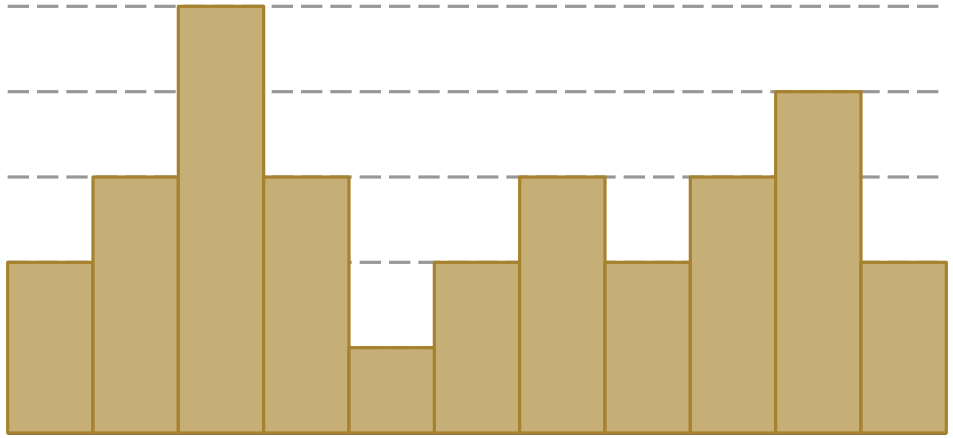
Problemstellung

- Biber hüpfen gerne auf Palettentürmen

Paletten-Hopping

Problemstellung

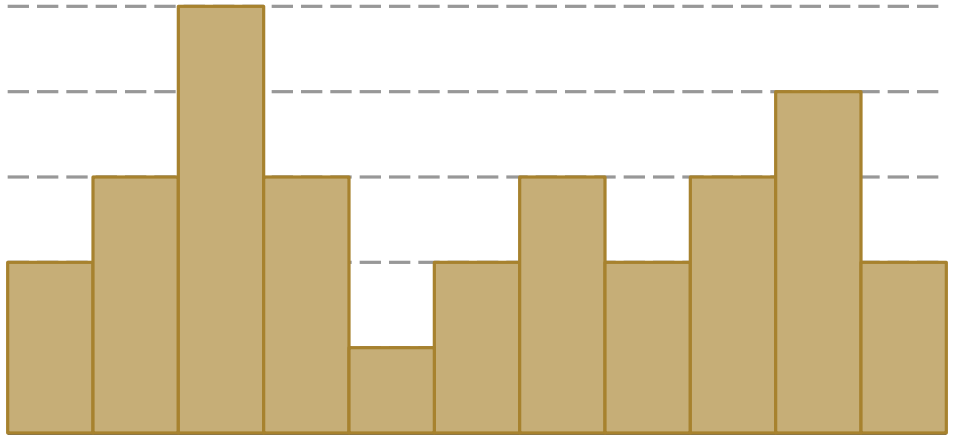
- Biber hüpfen gerne auf Palettentürmen
- Palettentürme haben eine unterschiedliche Höhe



Paletten-Hopping

Problemstellung

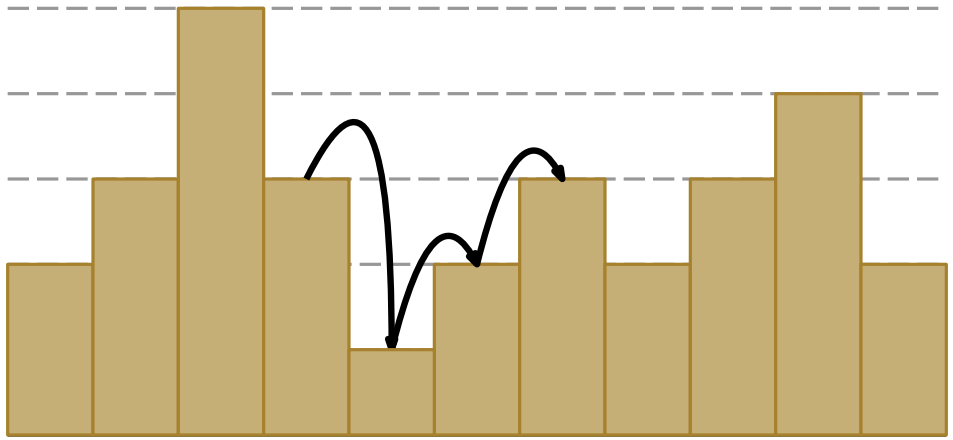
- Biber hüpfen gerne auf Palettentürmen
- Palettentürme haben eine unterschiedliche Höhe
- Hopping: nach $\rightarrow$ zu benachbarter Palette



Paletten-Hopping

Problemstellung

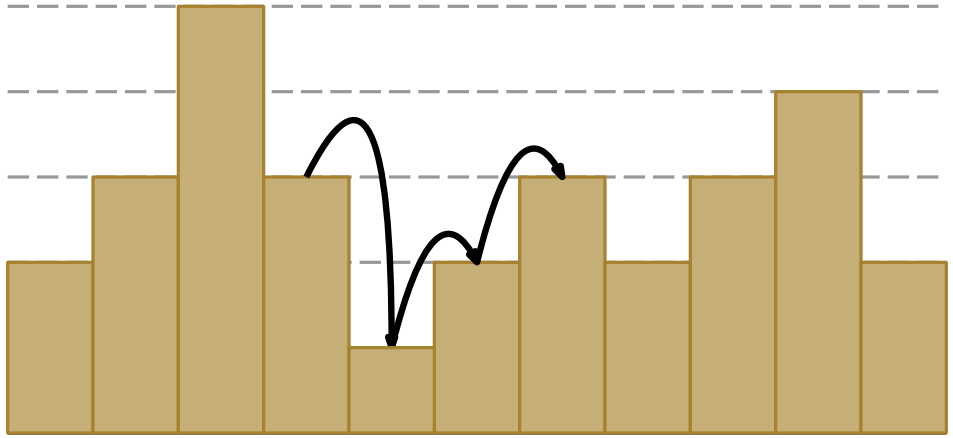
- Biber hüpfen gerne auf Palettentürmen
- Palettentürme haben eine unterschiedliche Höhe
- Hopping: nach $\rightarrow$ zu benachbarter Palette



Paletten-Hopping

Problemstellung

- Biber hüpfen gerne auf Palettentürmen
- Palettentürme haben eine unterschiedliche Höhe
- Hopping: nach $\rightarrow$ zu benachbarter Palette
- Biber-Hopping:
  Höhendifferenz zwischen Hops ≤ 1



Paletten-Hopping

Problemstellung

- Biber hüpfen gerne auf Palettentürmen
- Palettentürme haben eine unterschiedliche Höhe
- Hopping: nach $\rightarrow$ zu benachbarter Palette
- Biber-Hopping:
  Höhendifferenz zwischen Hops ≤ 1

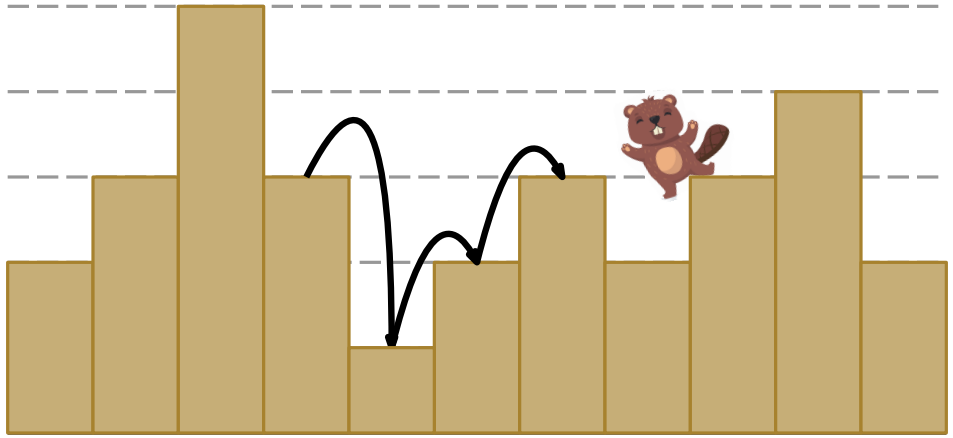


Bild: Funny Beaver Set von pch.vector, Freepik

Paletten-Hopping

Problemstellung

- Biber hüpfen gerne auf Palettentürmen
- Palettentürme haben eine unterschiedliche Höhe
- Hopping: nach $\rightarrow$ zu benachbarter Palette
- Biber-Hopping:
  Höhendifferenz zwischen Hops ≤ 1

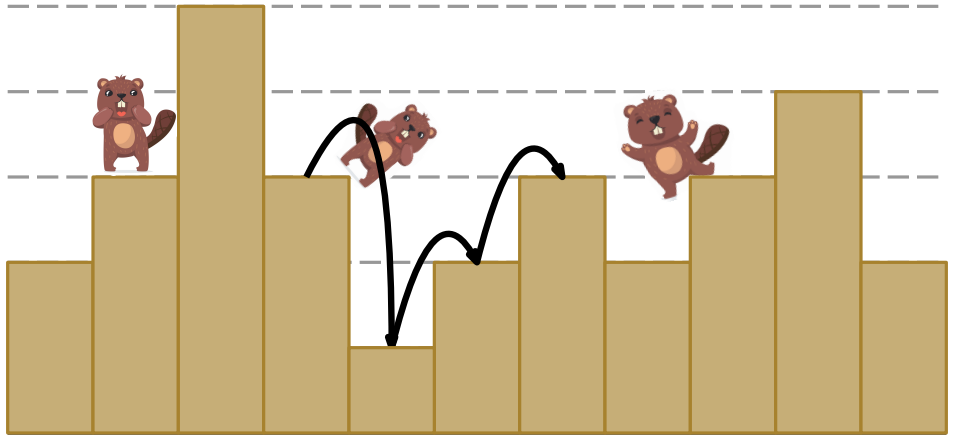


Bild: Funny Beaver Set von pch.vector, Freepik

Paletten-Hopping

Problemstellung

- Biber hüpfen gerne auf Palettentürmen
- Palettentürme haben eine unterschiedliche Höhe
- Hopping: nach $\rightarrow$ zu benachbarter Palette
- Biber-Hopping:
  Höhendifferenz zwischen Hops ≤ 1
- Maximales Biber-Hopping?

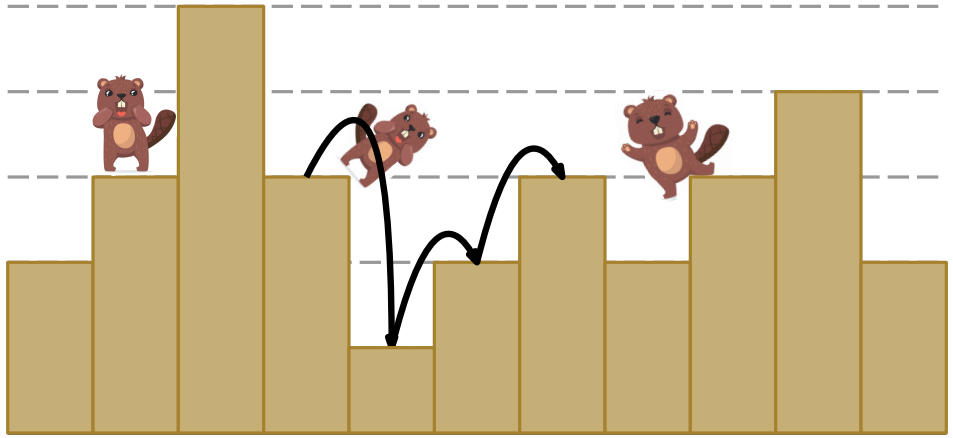


Bild: Funny Beaver Set von pch.vector, Freepik

Paletten-Hopping

Problemstellung

- Biber hüpfen gerne auf Palettentürmen
- Palettentürme haben eine unterschiedliche Höhe
- Hopping: nach $\rightarrow$ zu benachbarter Palette
- Biber-Hopping:
  Höhendifferenz zwischen Hops ≤ 1
- Maximales Biber-Hopping?

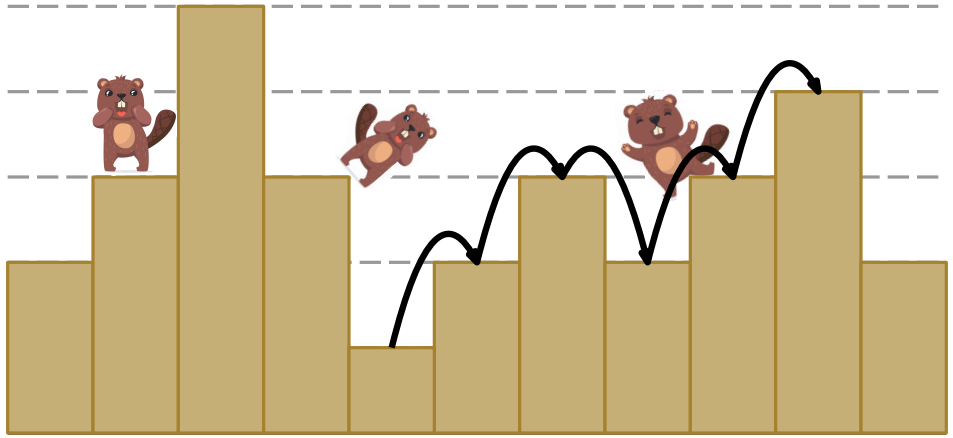


Bild: Funny Beaver Set von pch.vector, Freepik

Paletten-Hopping

Problemstellung

- Biber hüpfen gerne auf Palettentürmen
- Palettentürme haben eine unterschiedliche Höhe
- Hopping: nach $\rightarrow$ zu benachbarter Palette
- Biber-Hopping:
  Höhendifferenz zwischen Hops ≤ 1
- Maximales Biber-Hopping?

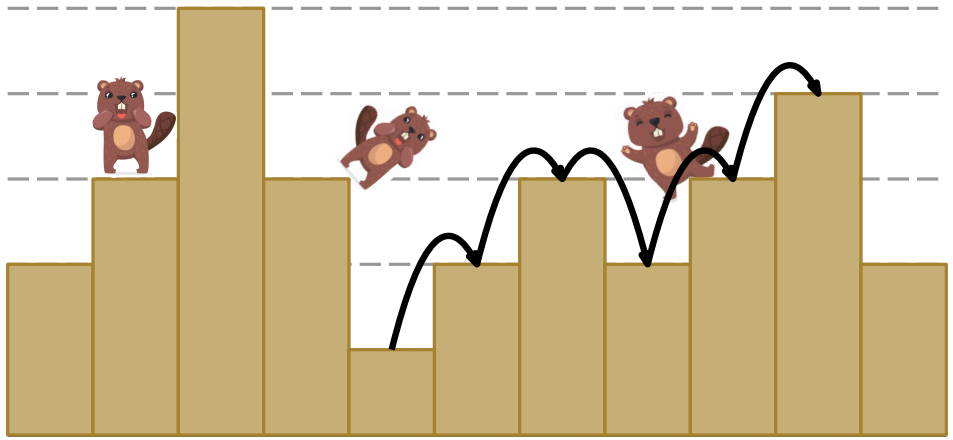


Bild: Funny Beaver Set von pch.vector, Freepik

2D

2	3	5
4	5	4
2	6	3

Paletten-Hopping

Problemstellung

- Biber hüpfen gerne auf Palettentürmen
- Palettentürme haben eine unterschiedliche Höhe
- Hopping: nach $\rightarrow$ zu benachbarter Palette
- Biber-Hopping:
  Höhendifferenz zwischen Hops ≤ 1
- Maximales Biber-Hopping?

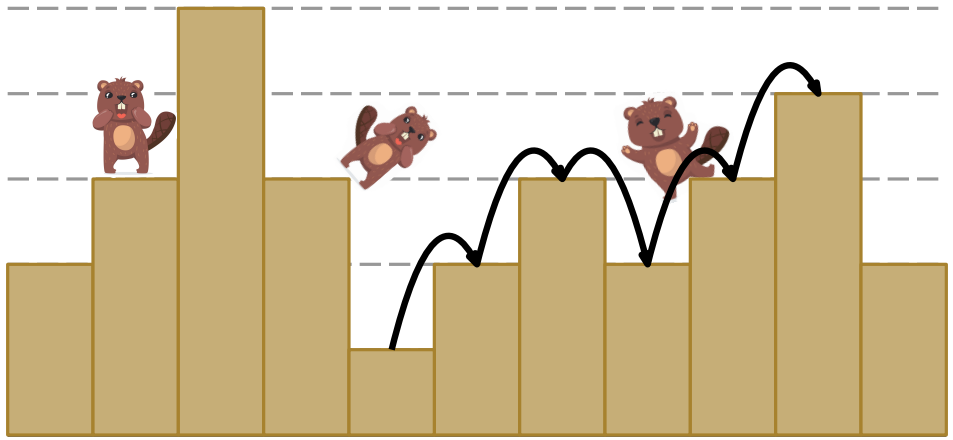
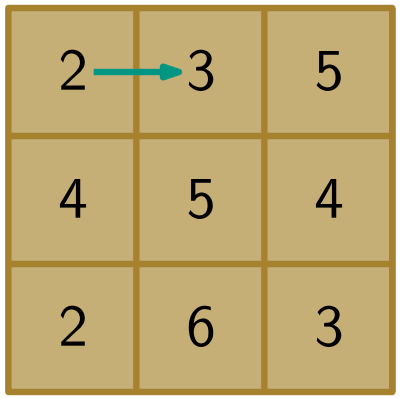


Bild: Funny Beaver Set von pch.vector, Freepik

2D



Paletten-Hopping

Problemstellung

- Biber hüpfen gerne auf Palettentürmen
- Palettentürme haben eine unterschiedliche Höhe
- Hopping: nach $\rightarrow$ zu benachbarter Palette
- Biber-Hopping:
  Höhendifferenz zwischen Hops ≤ 1
- Maximales Biber-Hopping?

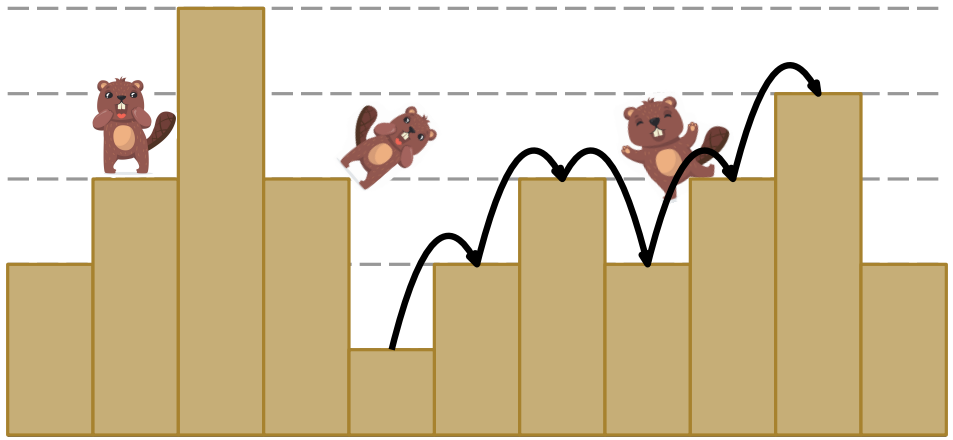
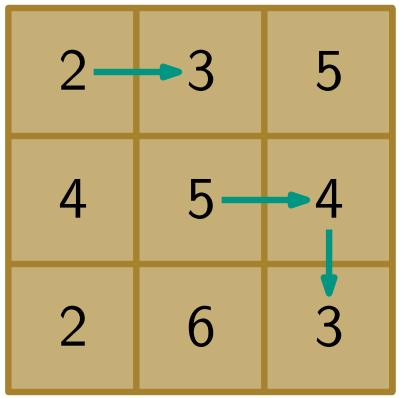


Bild: Funny Beaver Set von pch.vector, Freepik



Paletten-Hopping

Problemstellung

- Biber hüpfen gerne auf Palettentürmen
- Palettentürme haben eine unterschiedliche Höhe
- Hopping: nach $\rightarrow$ zu benachbarter Palette
- Biber-Hopping:
  Höhendifferenz zwischen Hops ≤ 1
- Maximales Biber-Hopping?

Formal (2D):

- n Zeilen, n Spalten

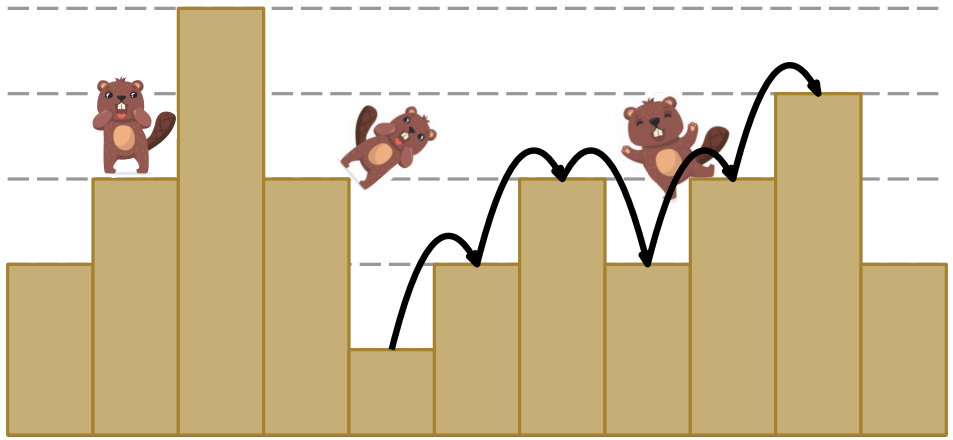
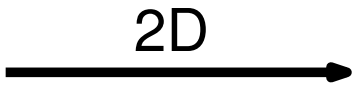
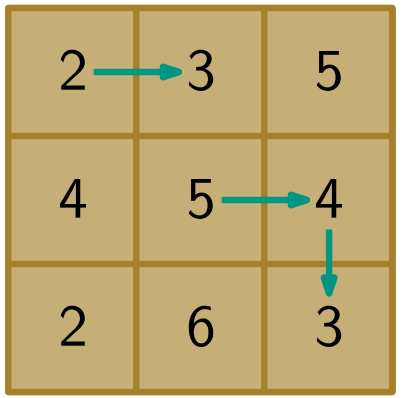


Bild: Funny Beaver Set von pch.vector, Freepik





Paletten-Hopping

Problemstellung

- Biber hüpfen gerne auf Palettentürmen
- Palettentürme haben eine unterschiedliche Höhe
- Hopping: nach $\rightarrow$ zu benachbarter Palette
- Biber-Hopping:
  Höhendifferenz zwischen Hops ≤ 1
- Maximales Biber-Hopping?

Formal (2D):

- n Zeilen, n Spalten
- Höhe $h(i, j) \in \mathbb{N}$

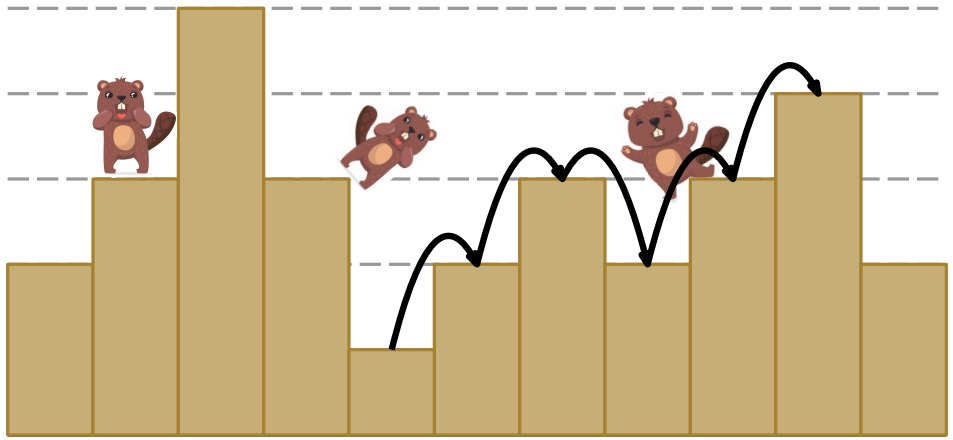
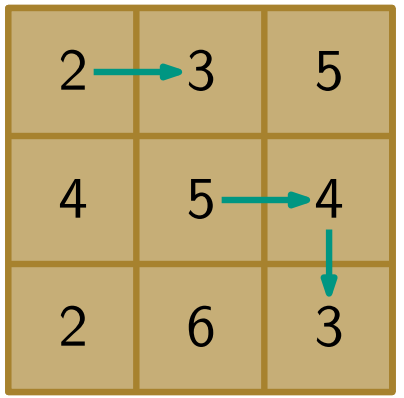


Bild: Funny Beaver Set von pch.vector, Freepik

2D



Paletten-Hopping

Problemstellung

- Biber hüpfen gerne auf Palettentürmen
- Palettentürme haben eine unterschiedliche Höhe
- Hopping: nach $\rightarrow$ zu benachbarter Palette
- Biber-Hopping:
  Höhendifferenz zwischen Hops ≤ 1
- Maximales Biber-Hopping?

Formal (2D):

- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \rightarrow (i+1,j)$ und $(i,j) \rightarrow (i,j+1)$

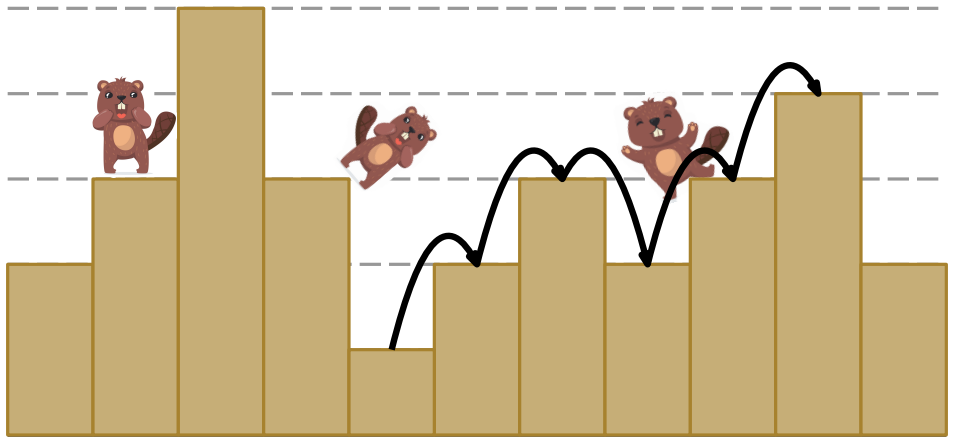
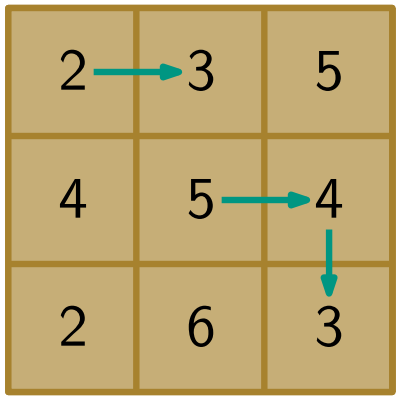


Bild: Funny Beaver Set von pch.vector, Freepik



Paletten-Hopping

Problemstellung

- Biber hüpfen gerne auf Palettentürmen
- Palettentürme haben eine unterschiedliche Höhe
- Hopping: nach $\rightarrow$ zu benachbarter Palette
- Biber-Hopping:
  Höhendifferenz zwischen Hops ≤ 1
- Maximales Biber-Hopping?

Formal (2D):

- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \rightarrow (i+1,j)$ und $(i,j) \rightarrow (i,j+1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

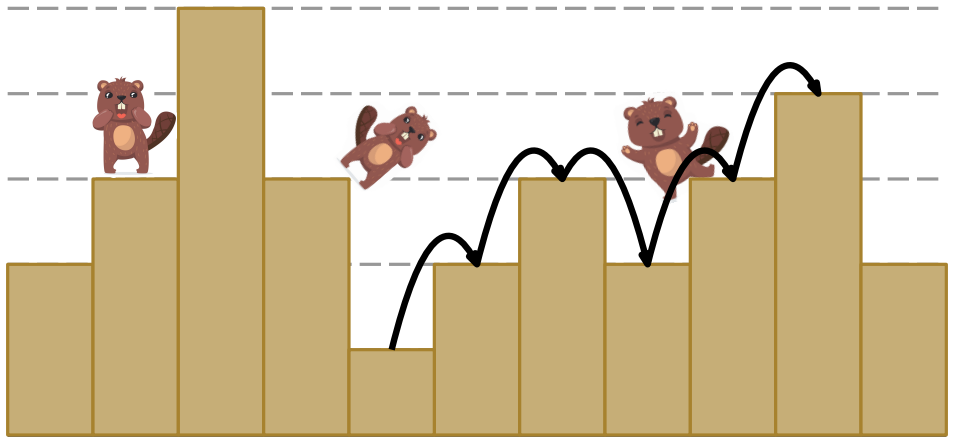
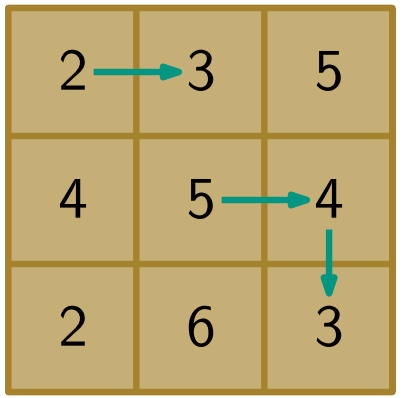


Bild: Funny Beaver Set von pch.vector, Freepik

2D



Paletten-Hopping – Teilprobleme

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

Formal (2D):

- n Zeilen, n Spalten
- Höhe $h(i, j) \in \mathbb{N}$
- Hops $(i, j) \to (i + 1, j)$ und $(i, j) \to (i, j + 1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Paletten-Hopping – Teilprobleme

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

Wie viele Biber-Hoppings gibt es?

Formal (2D):

- n Zeilen, n Spalten
- Höhe $h(i, j) \in \mathbb{N}$
- Hops $(i, j) \to (i + 1, j)$ und $(i, j) \to (i, j + 1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

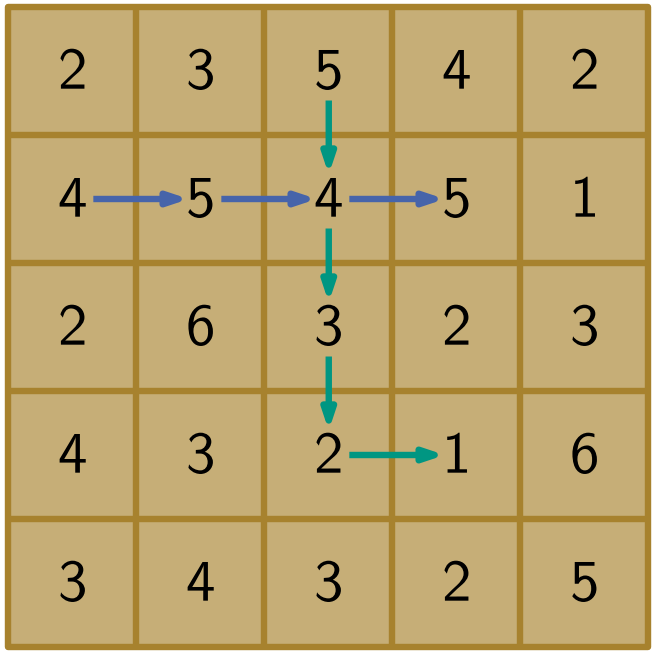
Paletten-Hopping – Teilprobleme

Wie viele Biber-Hoppings gibt es?

Formal (2D):

- n Zeilen, n Spalten
- Höhe $h(i, j) \in \mathbb{N}$
- Hops $(i, j) \to (i + 1, j)$ und $(i, j) \to (i, j + 1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Paletten-Hopping – Teilprobleme



Wie viele Biber-Hoppings gibt es?

Formal (2D):

- n Zeilen, n Spalten
- Höhe $h(i, j) \in \mathbb{N}$
- Hops $(i, j) \to (i + 1, j)$ und $(i, j) \to (i, j + 1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

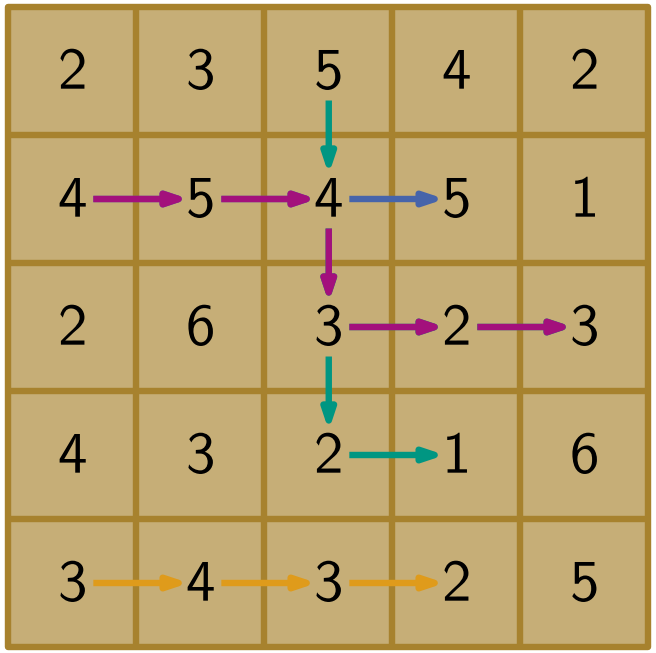
Paletten-Hopping – Teilprobleme

Wie viele Biber-Hoppings gibt es?

Formal (2D):

- n Zeilen, n Spalten
- Höhe $h(i, j) \in \mathbb{N}$
- Hops $(i, j) \to (i + 1, j)$ und $(i, j) \to (i, j + 1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Paletten-Hopping – Teilprobleme

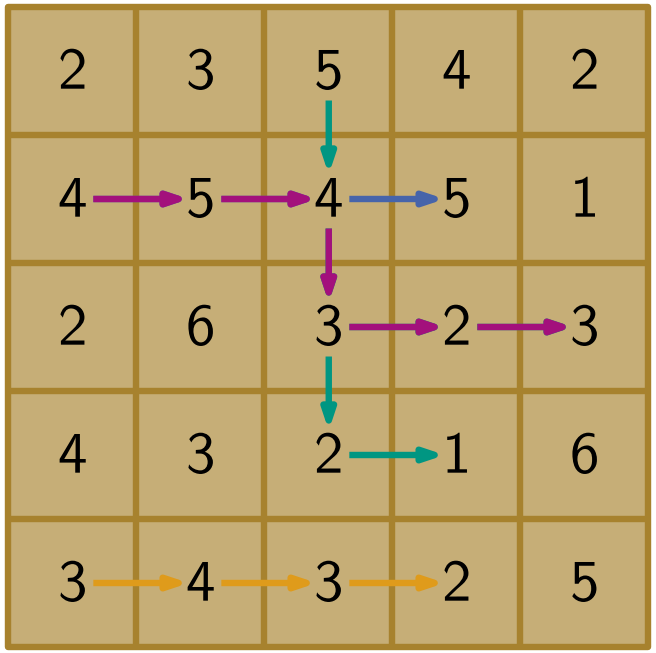


Wie viele Biber-Hoppings gibt es?

Formal (2D):

- n Zeilen, n Spalten
- Höhe $h(i, j) \in \mathbb{N}$
- Hops $(i, j) \to (i + 1, j)$ und $(i, j) \to (i, j + 1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Paletten-Hopping – Teilprobleme



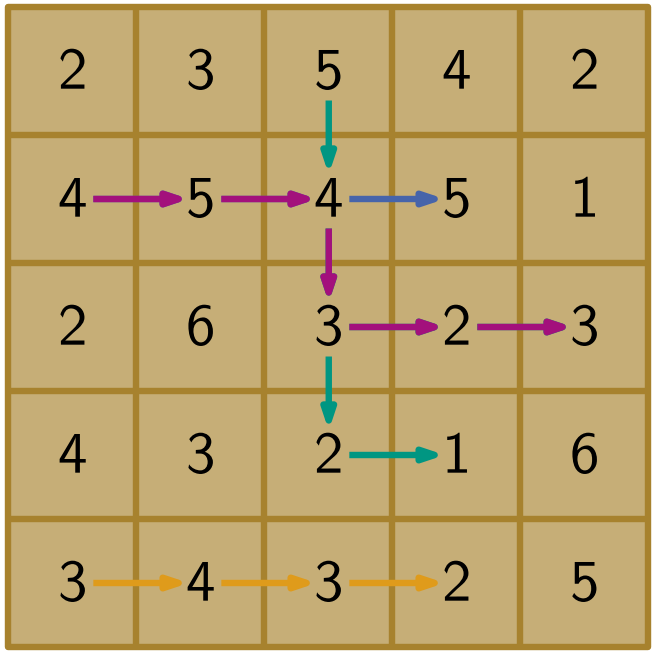
Formal (2D):

- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \to (i+1,j)$ und $(i,j) \to (i,j+1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Wie viele Biber-Hoppings gibt es?

- sehr sehr viele

Paletten-Hopping – Teilprobleme



Formal (2D):
- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \to (i+1,j)$ und $(i,j) \to (i,j+1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Wie viele Biber-Hoppings gibt es?
- sehr sehr viele
- können wir nicht alle aufzählen

Paletten-Hopping – Teilprobleme

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

Formal (2D):

- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \to (i+1,j)$ und $(i,j) \to (i,j+1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Teilprobleme

- $B(i,j) :=$ max. Länge Biber-Hopping mit Ende (i,j)

Paletten-Hopping – Teilprobleme

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

Formal (2D):
- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \to (i+1,j)$ und $(i,j) \to (i,j+1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Teilprobleme

- $B(i,j) :=$ max. Länge Biber-Hopping mit Ende (i,j)

Welche Infos sind nötig, um $B(i,j)$ zu bestimmen?

Paletten-Hopping – Teilprobleme

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

Formal (2D):
- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \to (i+1,j)$ und $(i,j) \to (i,j+1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Teilprobleme

- $B(i,j) :=$ max. Länge Biber-Hopping mit Ende (i,j)
- drei Möglichkeiten:

Paletten-Hopping – Teilprobleme

Formal (2D):
- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \to (i+1,j)$ und $(i,j) \to (i,j+1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Teilprobleme

- $B(i,j) :=$ max. Länge Biber-Hopping mit Ende (i,j)
- drei Möglichkeiten:
  - von oben erweiterbar

Paletten-Hopping – Teilprobleme

Formal (2D):
- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \to (i+1,j)$ und $(i,j) \to (i,j+1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Teilprobleme

- $B(i,j) :=$ max. Länge Biber-Hopping mit Ende (i,j)
- drei Möglichkeiten:
  - von oben erweiterbar

$$B(i,j) = B(i-1,j) + 1$$

Paletten-Hopping – Teilprobleme

Formal (2D):
- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \to (i+1,j)$ und $(i,j) \to (i,j+1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Teilprobleme

- $B(i,j) :=$ max. Länge Biber-Hopping mit Ende (i,j)
- drei Möglichkeiten:
  - von oben erweiterbar
  - von links erweiterbar

$$B(i,j) = B(i-1,j) + 1$$

Paletten-Hopping – Teilprobleme

Formal (2D):
- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \to (i+1,j)$ und $(i,j) \to (i,j+1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Teilprobleme

- $B(i,j) :=$ max. Länge Biber-Hopping mit Ende (i,j)
- drei Möglichkeiten:
  - von oben erweiterbar
  - von links erweiterbar

$$B(i,j) = B(i-1,j) + 1$$

$$B(i,j) = B(i,j-1) + 1$$

Paletten-Hopping – Teilprobleme

Formal (2D):
- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \to (i+1,j)$ und $(i,j) \to (i,j+1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Teilprobleme

- $B(i,j) :=$ max. Länge Biber-Hopping mit Ende (i,j)
- drei Möglichkeiten:
  - von oben erweiterbar
  - von links erweiterbar
  - nicht erweiterbar

$$B(i,j) = B(i-1,j) + 1$$

$$B(i,j) = B(i,j-1) + 1$$

Paletten-Hopping – Teilprobleme

Formal (2D):
- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \to (i+1,j)$ und $(i,j) \to (i,j+1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Teilprobleme

- $B(i,j) :=$ max. Länge Biber-Hopping mit Ende (i,j)
- drei Möglichkeiten:
  - von oben erweiterbar
  - von links erweiterbar
  - nicht erweiterbar

$$B(i,j) = B(i-1,j) + 1$$

$$B(i,j) = B(i,j-1) + 1$$

$$B(i,j) = 1$$

Paletten-Hopping – Teilprobleme

Formal (2D):
- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \to (i+1,j)$ und $(i,j) \to (i,j+1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Teilprobleme

- $B(i,j) :=$ max. Länge Biber-Hopping mit Ende (i,j)
- drei Möglichkeiten:
  - von oben erweiterbar
  - von links erweiterbar
  - nicht erweiterbar

$$B(i,j) = B(i-1,j) + 1$$

$$B(i,j) = B(i,j-1) + 1$$

$$B(i,j) = 1$$

Rekurrenz

$B(i,0) := 0$ und $B(0,j) := 0$

$$B(i,j) = \max \begin{cases} B(i-1,j)+1, \text{ falls } |h(i-1,j) - h(i,j)| \leq 1 \\ B(i,j-1)+1, \text{ falls } |h(i,j) - h(i,j-1)| \leq 1 \\ 1 \end{cases}$$

Paletten-Hopping – Teilprobleme

Formal (2D):
- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \to (i+1,j)$ und $(i,j) \to (i,j+1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Teilprobleme

- $B(i,j) :=$ max. Länge Biber-Hopping mit Ende (i,j)
- drei Möglichkeiten:
  - von oben erweiterbar
  - von links erweiterbar
  - nicht erweiterbar

$B(i,j) = B(i-1,j) + 1$

$B(i,j) = B(i,j-1) + 1$

$B(i,j) = 1$

Rekurrenz

$B(i,0) := 0$ und $B(0,j) := 0$

$$B(i,j) = \max \begin{cases} B(i-1,j)+1, \text{ falls } |h(i-1,j) - h(i,j)| \leq 1 \\ B(i,j-1)+1, \text{ falls } |h(i,j) - h(i,j-1)| \leq 1 \\ 1 \end{cases}$$

Laufzeit von $B(i,j)$ bei rekursiver Implementierung?
- $O(n)$
- $O(n^2)$
- $O(2^n)$
- $O(n!)$

Paletten-Hopping – Teilprobleme

Formal (2D):
- n Zeilen, n Spalten
- Höhe $h(i,j) \in \mathbb{N}$
- Hops $(i,j) \to (i+1,j)$ und $(i,j) \to (i,j+1)$
- maximiere Länge Biber-Hopping $(b_1, \ldots, b_\ell)$: $|h(b_k) - h(b_{k+1})| \leq 1$

Teilprobleme

- $B(i,j) :=$ max. Länge Biber-Hopping mit Ende (i,j)
- drei Möglichkeiten:
  - von oben erweiterbar
  - von links erweiterbar
  - nicht erweiterbar

$$B(i,j) = B(i-1,j) + 1$$

$$B(i,j) = B(i,j-1) + 1$$

$$B(i,j) = 1$$

Rekurrenz

$B(i,0) := 0$ und $B(0,j) := 0$

$$B(i,j) = \max \begin{cases} B(i-1,j)+1, \text{ falls } |h(i-1,j) - h(i,j)| \leq 1 \\ B(i,j-1)+1, \text{ falls } |h(i,j) - h(i,j-1)| \leq 1 \\ 1 \end{cases}$$

- Werte zwischenspeichern

Iterative Implementierung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

Rekurrenz

$B(i,0) := 0$ und $B(0,j) := 0$

$$B(i,j) = \max \begin{cases} B(i-1,j)+1, \text{ falls } |h(i-1,j)-h(i,j)| \leq 1 \\ B(i,j-1)+1, \text{ falls } |h(i,j)-h(i,j-1)| \leq 1 \\ 1 \end{cases}$$

Iterative Implementierung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

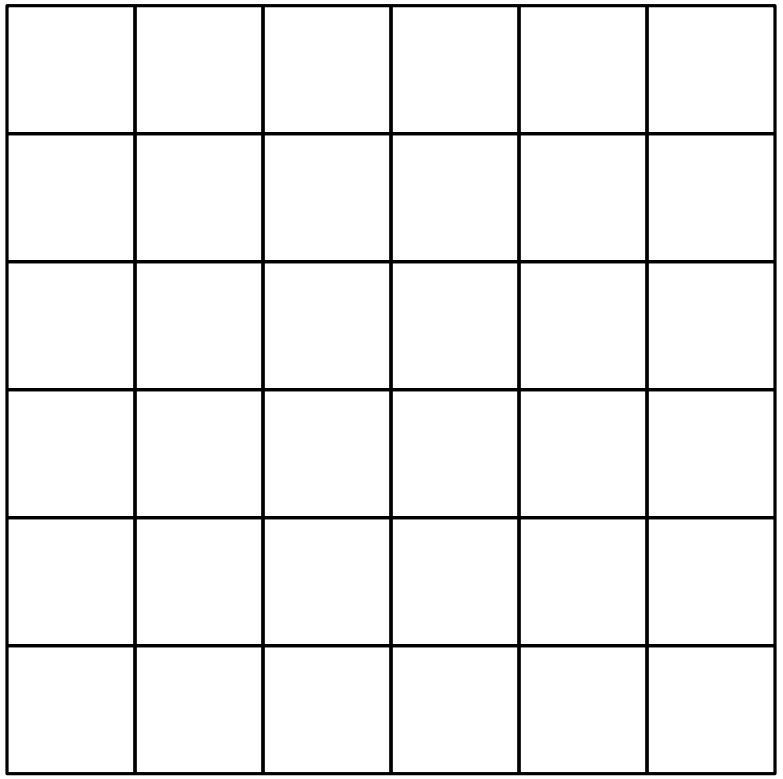
~~Rekurrenz~~ **Iterativ**

$B[i][0] := 0$ und $B[0][j] := 0$

$$B[i][j] = \max \begin{cases} B[i-1][j]+1, \text{ falls } |h(i-1,j)-h(i,j)| \leq 1 \\ B[i][j-1]+1, \text{ falls } |h(i,j)-h(i,j-1)| \leq 1 \\ 1 \end{cases}$$

Iterative Implementierung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5



~~Rekurrenz~~ Iterativ

$B[i][0] := 0$ und $B[0][j] := 0$

$$B[i][j] = \max \begin{cases} B[i-1][j]+1, \text{ falls } |h(i-1,j)-h(i,j)| \leq 1 \\ B[i][j-1]+1, \text{ falls } |h(i,j)-h(i,j-1)| \leq 1 \\ 1 \end{cases}$$

Iterative Implementierung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

0	0	0	0	0	0
0					
0					
0					
0					
0					

Rekurrenz **Iterativ**

$B[i][0] := 0$ und $B[0][j] := 0$

$$B[i][j] = \max \begin{cases} B[i-1][j]+1, \text{ falls } |h(i-1,j)-h(i,j)| \leq 1 \\ B[i][j-1]+1, \text{ falls } |h(i,j)-h(i,j-1)| \leq 1 \\ 1 \end{cases}$$

Iterative Implementierung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

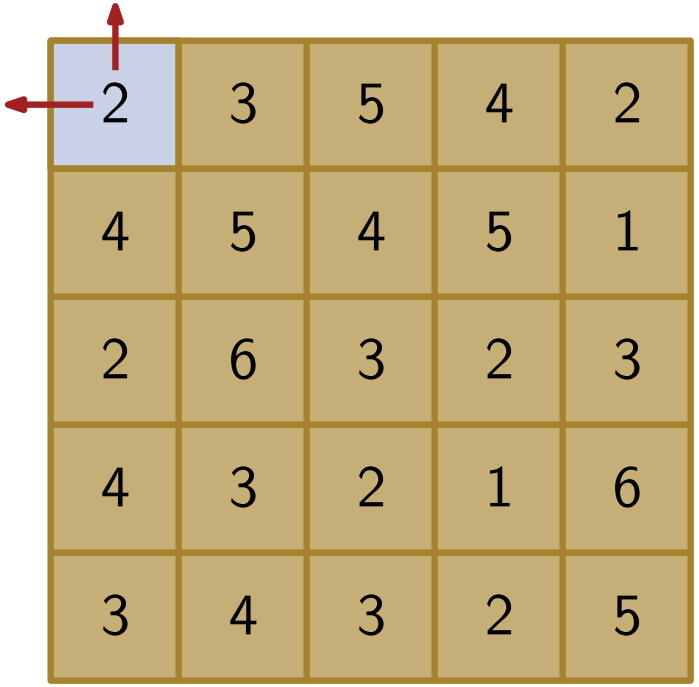
0	0	0	0	0	0
0					
0					
0					
0					
0					

~~Rekurrenz~~ **Iterativ**

$B[i][0] := 0$ und $B[0][j] := 0$

$$B[i][j] = \max \begin{cases} B[i-1][j]+1, \text{ falls } |h(i-1,j)-h(i,j)| \leq 1 \\ B[i][j-1]+1, \text{ falls } |h(i,j)-h(i,j-1)| \leq 1 \\ 1 \end{cases}$$

Iterative Implementierung



2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

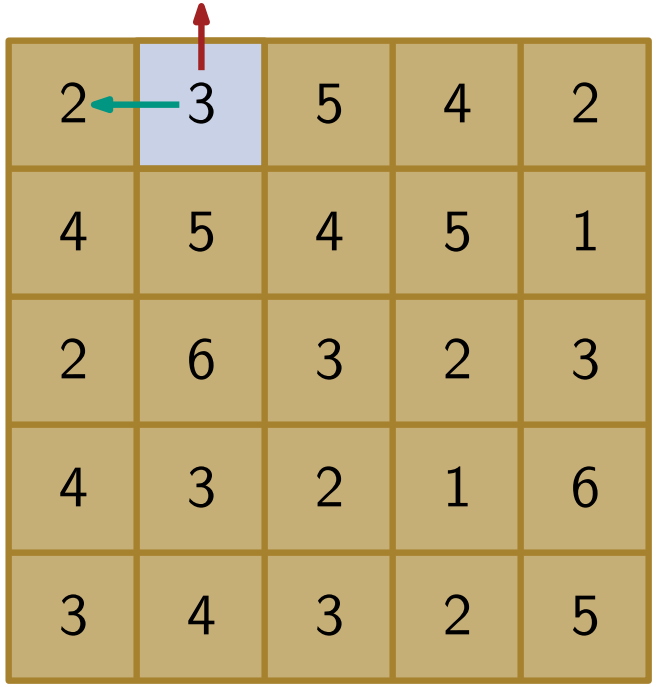
0	0	0	0	0	0
0	1				
0					
0					
0					
0					

~~Rekurrenz~~ **Iterativ**

$B[i][0] := 0$ und $B[0][j] := 0$

$$B[i][j] = \max \begin{cases} B[i-1][j]+1, \text{ falls } |h(i-1,j)-h(i,j)| \leq 1 \\ B[i][j-1]+1, \text{ falls } |h(i,j)-h(i,j-1)| \leq 1 \\ 1 \end{cases}$$

Iterative Implementierung



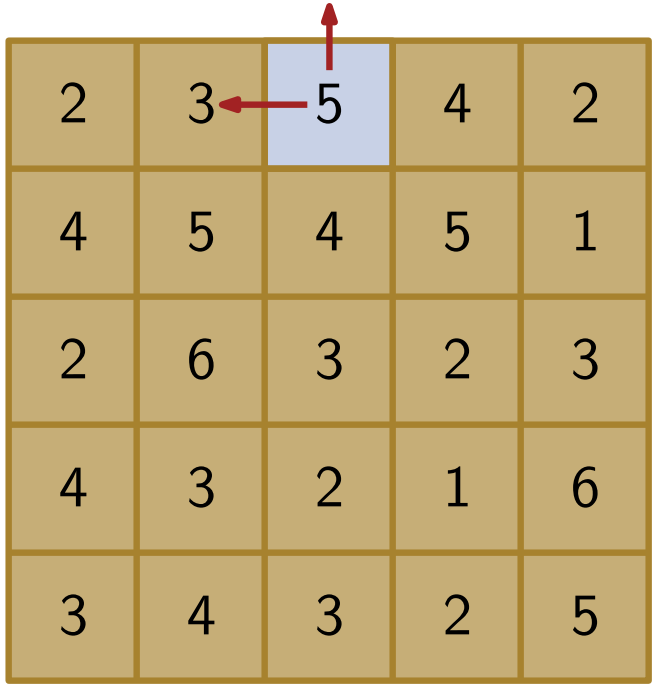
0	0	0	0	0	0
0	1	2			
0					
0					
0					
0					

Rekurrenz **Iterativ**

$B[i][0] := 0$ und $B[0][j] := 0$

$$B[i][j] = \max \begin{cases} B[i-1][j]+1, \text{ falls } |h(i-1,j)-h(i,j)| \leq 1 \\ B[i][j-1]+1, \text{ falls } |h(i,j)-h(i,j-1)| \leq 1 \\ 1 \end{cases}$$

Iterative Implementierung



0	0	0	0	0	0
0	1	2	1		
0					
0					
0					
0					

Rekurrenz **Iterativ**

$B[i][0] := 0$ und $B[0][j] := 0$

$$B[i][j] = \max \begin{cases} B[i-1][j]+1, \text{ falls } |h(i-1,j)-h(i,j)| \leq 1 \\ B[i][j-1]+1, \text{ falls } |h(i,j)-h(i,j-1)| \leq 1 \\ 1 \end{cases}$$

Iterative Implementierung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

0	0	0	0	0	0
0	1	2	1	2	1
0	1	2			
0					
0					
0					

~~Rekurrenz~~ **Iterativ**

$B[i][0] := 0$ und $B[0][j] := 0$

$$B[i][j] = \max \begin{cases} B[i-1][j]+1, \text{ falls } |h(i-1,j)-h(i,j)| \leq 1 \\ B[i][j-1]+1, \text{ falls } |h(i,j)-h(i,j-1)| \leq 1 \\ 1 \end{cases}$$

Iterative Implementierung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

0	0	0	0	0	0
0	1	2	1	2	1
0	1	2	3		
0					
0					
0					

Rekurrenz **Iterativ**

$B[i][0] := 0$ und $B[0][j] := 0$

$$B[i][j] = \max \begin{cases} B[i-1][j]+1, \text{ falls } |h(i-1,j)-h(i,j)| \leq 1 \\ B[i][j-1]+1, \text{ falls } |h(i,j)-h(i,j-1)| \leq 1 \\ 1 \end{cases}$$

Iterative Implementierung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

0	0	0	0	0	0
0	1	2	1	2	1
0	1	2	3	4	2
0	1	3	4	5	6
0	1	2	5	6	1
0	2	3	6	7	2

~~Rekurrenz~~ Iterativ

$B[i][0] := 0$ und $B[0][j] := 0$

$$B[i][j] = \max \begin{cases} B[i-1][j]+1, \text{ falls } |h(i-1,j)-h(i,j)| \leq 1 \\ B[i][j-1]+1, \text{ falls } |h(i,j)-h(i,j-1)| \leq 1 \\ 1 \end{cases}$$

Iterative Implementierung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

0	0	0	0	0	0
0	1	2	1	2	1
0	1	2	3	4	2
0	1	3	4	5	6
0	1	2	5	6	1
0	2	3	6	7	2

Laufzeit von $B(i, j)$ bei iterativer Implementierung?

- $O(n)$
- $O(n^2)$
- $O(2^n)$
- $O(n!)$

Iterative Implementierung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

0	0	0	0	0	0
0	1	2	1	2	1
0	1	2	3	4	2
0	1	3	4	5	6
0	1	2	5	6	1
0	2	3	6	7	2

Was ist die maximale Länge eines Biber-Hoppings?

Iterative Implementierung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

0	0	0	0	0	0
0	1	2	1	2	1
0	1	2	3	4	2
0	1	3	4	5	6
0	1	2	5	6	1
0	2	3	6	7	2

Was ist die maximale Länge eines Biber-Hoppings?

Iterative Implementierung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

0	0	0	0	0	0
0	1	2	1	2	1
0	1	2	3	4	2
0	1	3	4	5	6
0	1	2	5	6	1
0	2	3	6	7	2

Was ist die maximale Länge eines Biber-Hoppings?

Wie rekonstruiert man eine Lösung?

Rekonstruktion der Lösung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

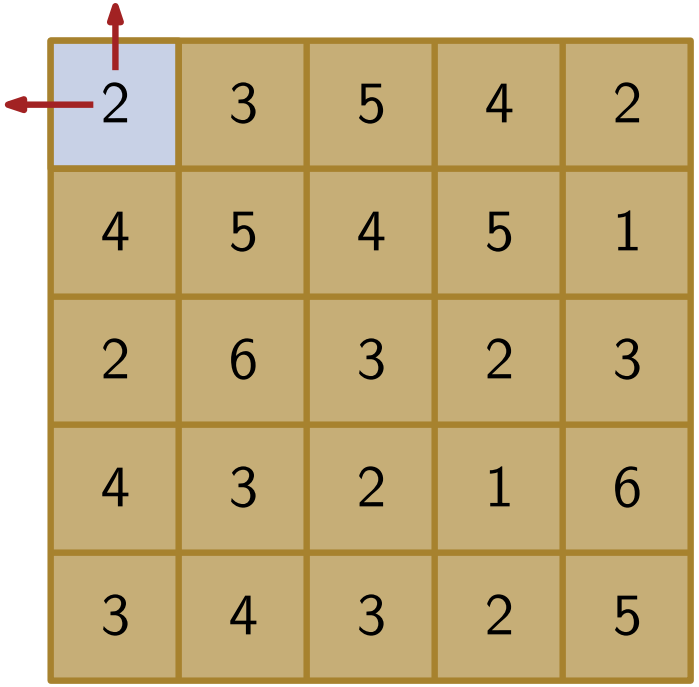
Rekonstruktion der Lösung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

0	0	0	0	0	0
0					
0					
0					
0					
0					

- speichere den passenden Vorgänger

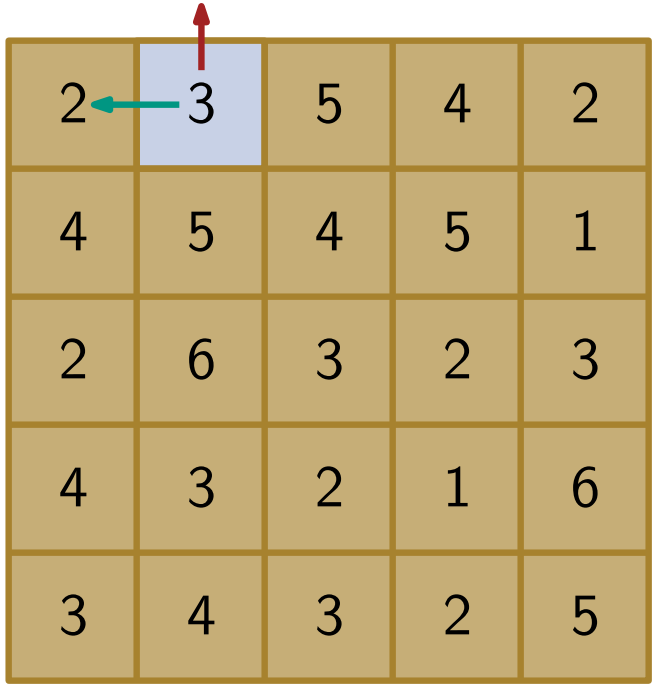
Rekonstruktion der Lösung



0	0	0	0	0	0
0	1				
0					
0					
0					
0					

- speichere den passenden Vorgänger

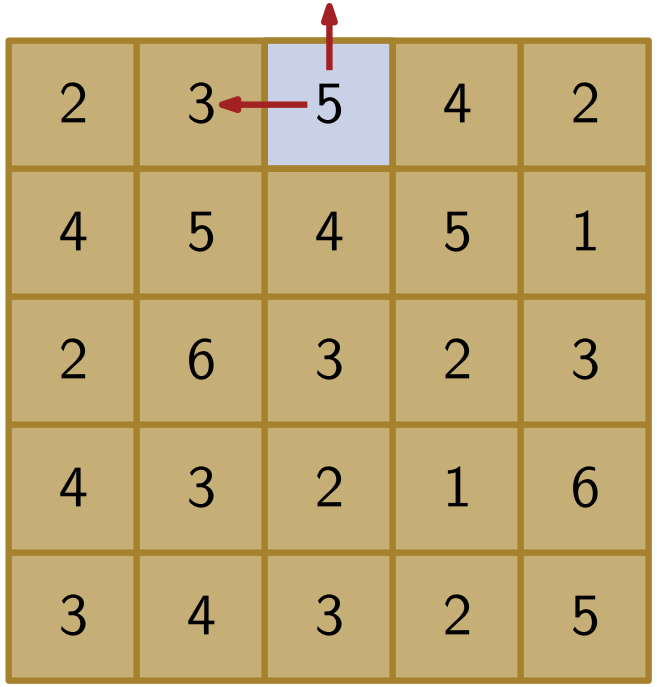
Rekonstruktion der Lösung



0	0	0	0	0	0
0	1	2			
0					
0					
0					
0					

- speichere den passenden Vorgänger

Rekonstruktion der Lösung

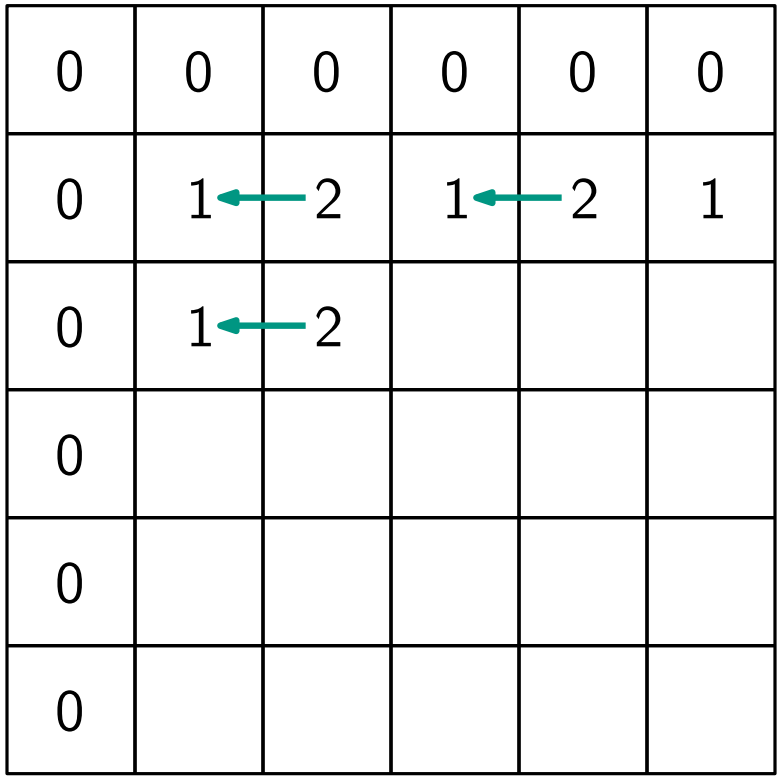


0	0	0	0	0	0
0	1	2	1		
0					
0					
0					
0					

- speichere den passenden Vorgänger

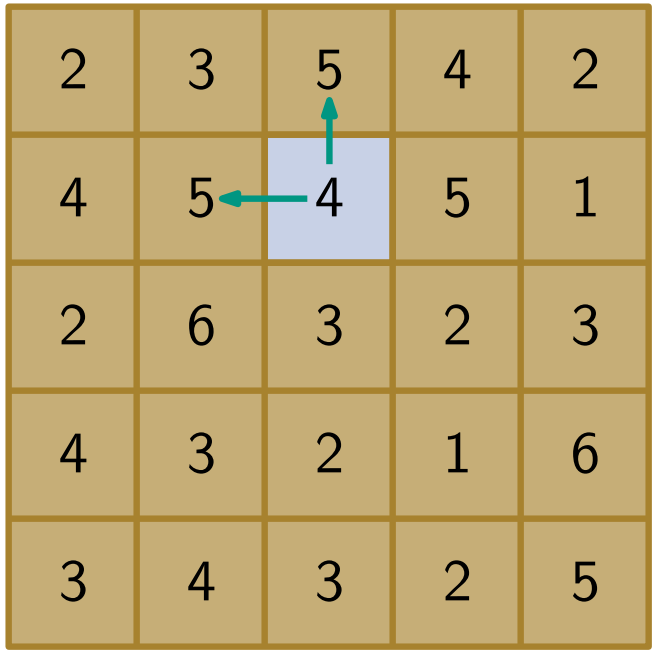
Rekonstruktion der Lösung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5



- speichere den passenden Vorgänger

Rekonstruktion der Lösung

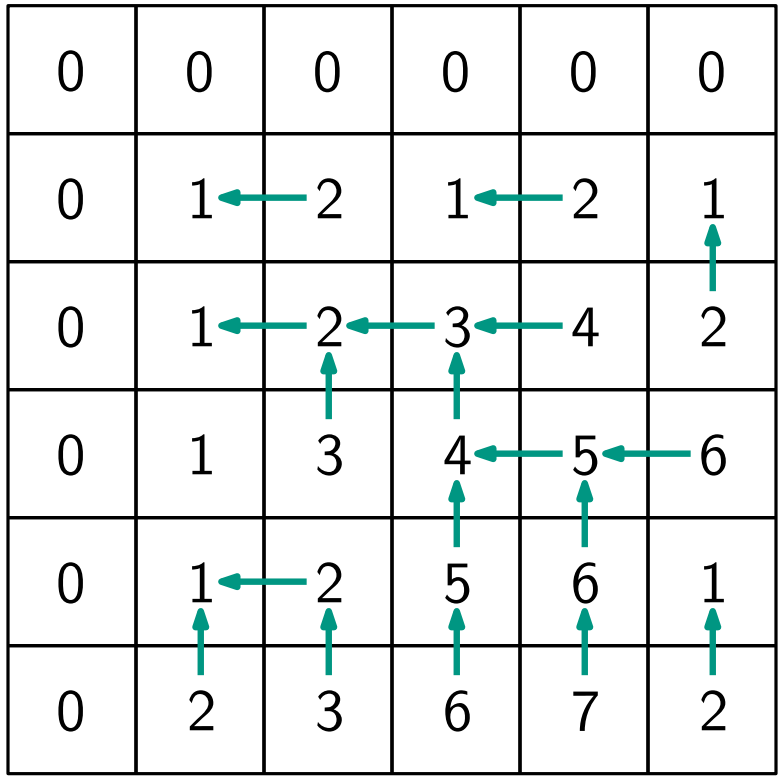


0	0	0	0	0	0
0	1	2	1	2	1
0	1	2	3		
0					
0					
0					

- speichere den passenden Vorgänger

Rekonstruktion der Lösung

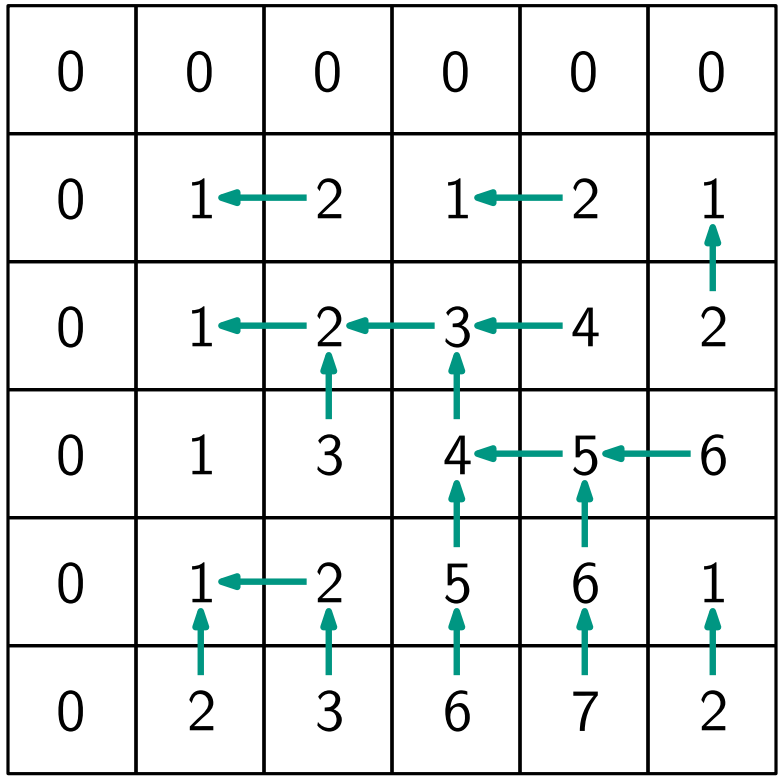
2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5



- speichere den passenden Vorgänger

Rekonstruktion der Lösung

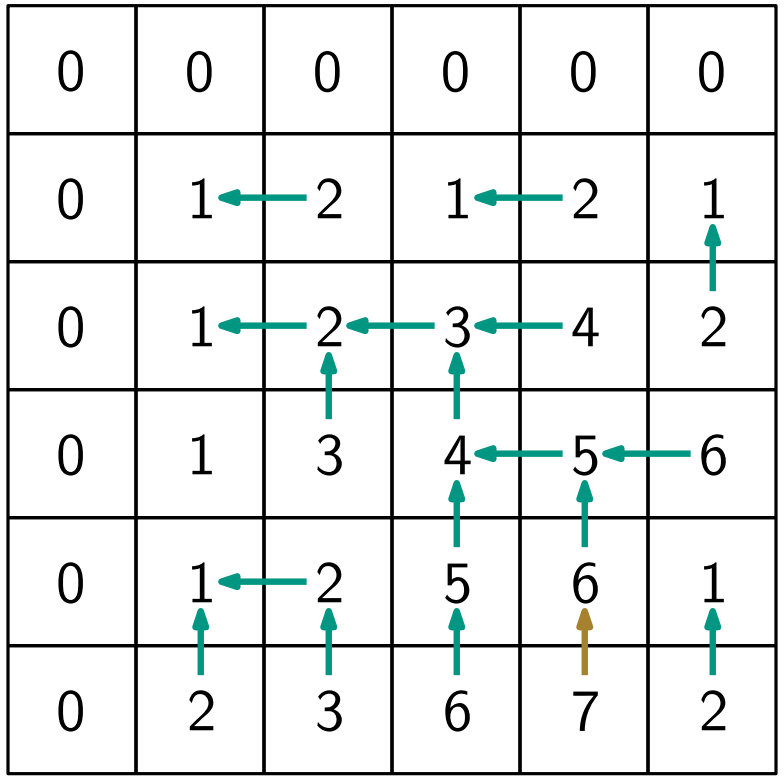
2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5



- speichere den passenden Vorgänger
- verfolge Zeiger zurück

Rekonstruktion der Lösung

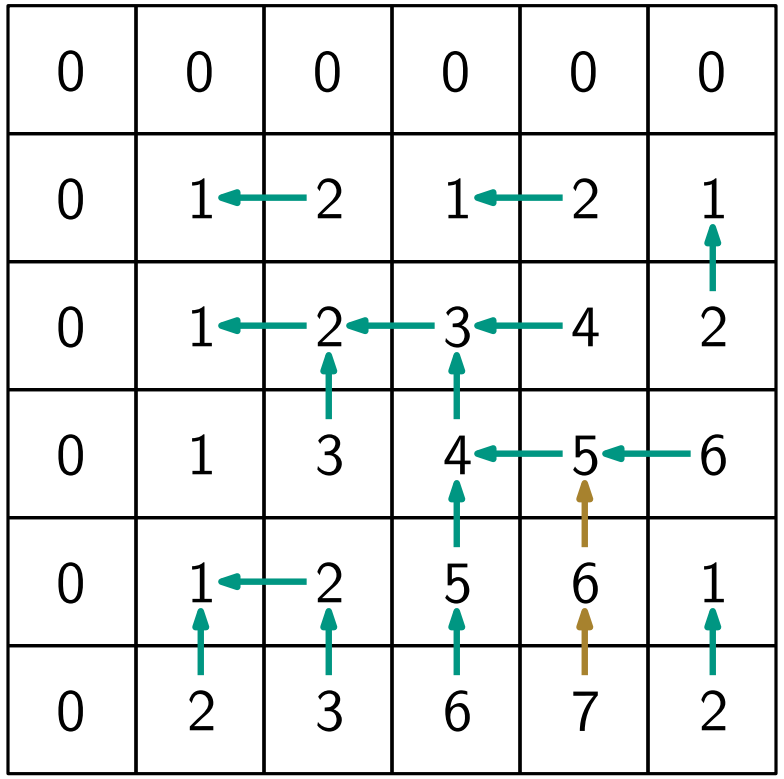
2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5



- speichere den passenden Vorgänger
- verfolge Zeiger zurück

Rekonstruktion der Lösung

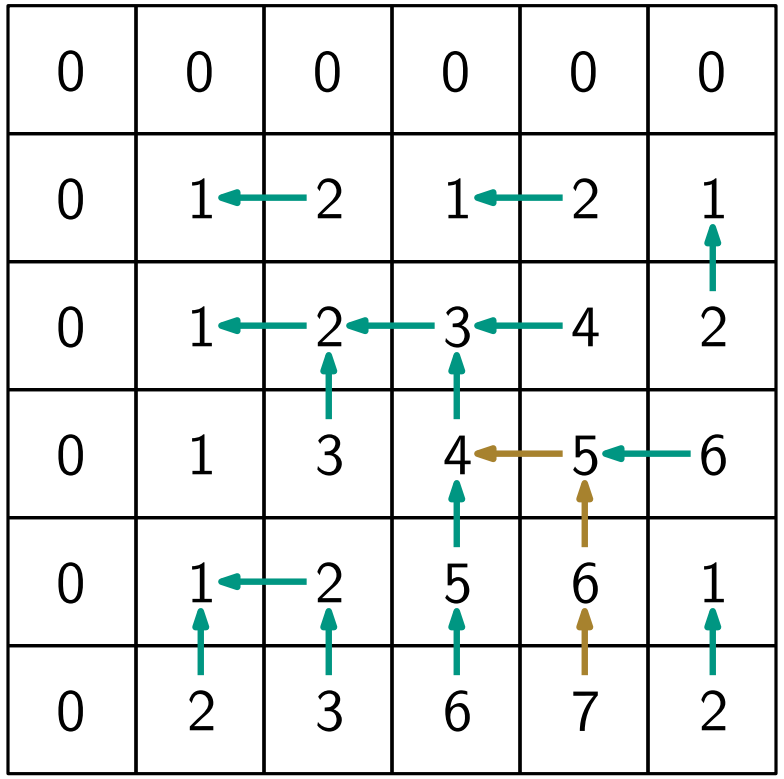
2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5



- speichere den passenden Vorgänger
- verfolge Zeiger zurück

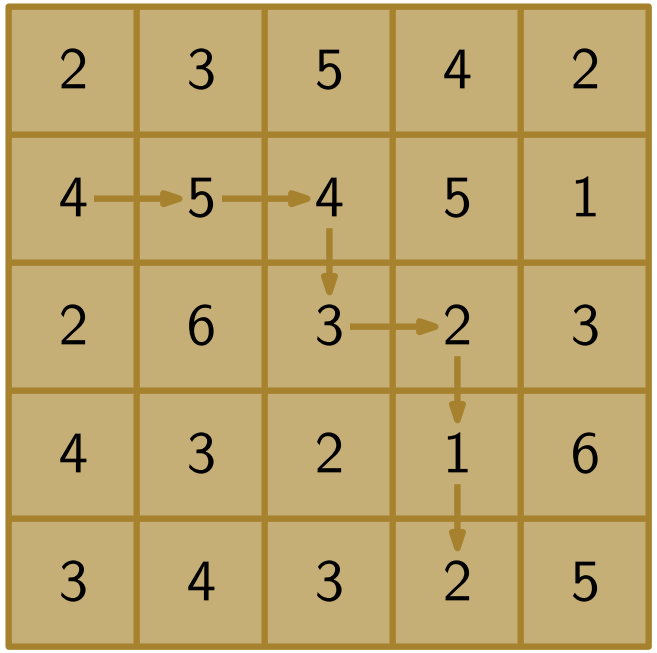
Rekonstruktion der Lösung

2	3	5	4	2
4	5	4	5	1
2	6	3 →	2	3
4	3	2	1 ↓	6
3	4	3	2	5



- speichere den passenden Vorgänger
- verfolge Zeiger zurück

Rekonstruktion der Lösung



0	0	0	0	0	0
0	1	2	1	2	1
0	1	2	3	4	2
0	1	3	4	5	6
0	1	2	5	6	1
0	2	3	6	7	2

- speichere den passenden Vorgänger
- verfolge Zeiger zurück

Rekonstruktion der Lösung

2	3	5	4	2
4	5	4	5	1
2	6	3	2	3
4	3	2	1	6
3	4	3	2	5

0	0	0	0	0	0
0	1	2	1	2	1
0	1	2	3	4	2
0	1	3	4	5	6
0	1	2	5	6	1
0	2	3	6	7	2

Laufzeit für Lösung rekonstruieren (optimales Ende bekannt)?

- $O(n)$
- $O(n^2)$
- $O(2^n)$
- $O(n!)$

Amortisierte Analyse

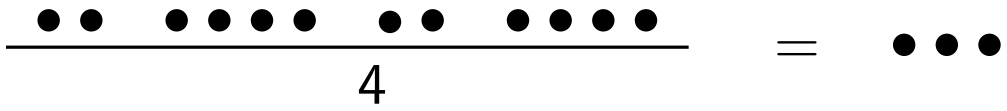
Aggregatmethode

- berechne Gesamtkosten
- teile durch Anzahl von Operationen

Amortisierte Analyse

Aggregatmethode

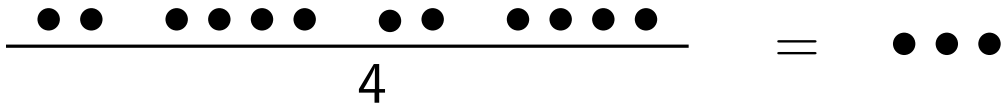
- berechne Gesamtkosten
- teile durch Anzahl von Operationen



Amortisierte Analyse

Aggregatmethode

- berechne Gesamtkosten
- teile durch Anzahl von Operationen



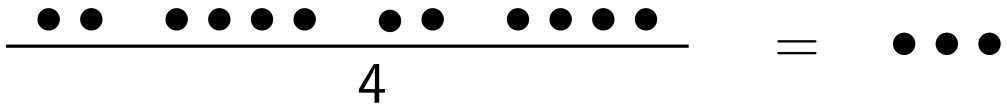
Charging

- verteile Kosten-Tokens von teuren zu günstigen Ops

Amortisierte Analyse

Aggregatmethode

- berechne Gesamtkosten
- teile durch Anzahl von Operationen

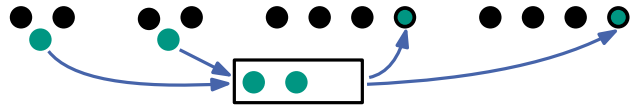


Charging

- verteile Kosten-Tokens von teuren zu günstigen Ops

Kontomethode

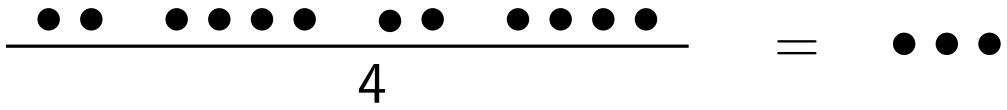
- günstige Op zahlt mehr ins Konto ein, teure Op hebt ab
- Achtung: Kontostand ≥ 0



Amortisierte Analyse

Aggregatmethode

- berechne Gesamtkosten
- teile durch Anzahl von Operationen

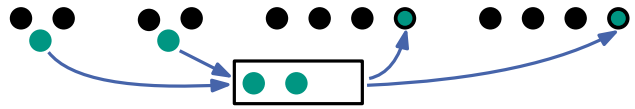


Charging

- verteile Kosten-Tokens von teuren zu günstigen Ops

Kontomethode

- günstige Op zahlt mehr ins Konto ein, teure Op hebt ab
- Achtung: Kontostand ≥ 0



Potentialmethode

- Potential $\Phi(A) \geq 0$ abhängig vom Zustand von A
- amortisierte Kosten = tatsächliche Kosten + $\Phi(A_{\text{nach}}) - \Phi(A_{\text{vor}})$

Amortisierte Analyse

Aufgabe

Aggregatmethode
- Gesamtkosten / Anzahl Ops

Charging
- Kostentoken: teuere Op $\rightarrow$ günstige Op

Kontomethode
- günstige Op $\rightarrow$ Konto, Konto $\rightarrow$ teure Op

Potentialmethode
- Potential $\Phi(A) \geq 0$ abhängig vom Zustand von A
- amort. Kosten = tats. Kosten + $\Phi(A_{\text{nach}}) - \Phi(A_{\text{vor}})$

Amortisierte Analyse

Aufgabe

- Kugeln (drei Farben) in einer Reihe

Aggregatmethode

- Gesamtkosten / Anzahl Ops

Charging

- Kostentoken: teuere Op $\rightarrow$ günstige Op

Kontomethode

- günstige Op $\rightarrow$ Konto, Konto $\rightarrow$ teure Op

Potentialmethode

- Potential $\Phi(A) \geq 0$ abhängig vom Zustand von A
- amort. Kosten = tats. Kosten + $\Phi(A_{\text{nach}}) - \Phi(A_{\text{vor}})$

Amortisierte Analyse

Aufgabe

- Kugeln (drei Farben) in einer Reihe
- Operationen
  - `insert`(pos, color) in $\Theta(1)$

Aggregatmethode

- Gesamtkosten / Anzahl Ops

Charging

- Kostentoken: teuere Op $\rightarrow$ günstige Op

Kontomethode

- günstige Op $\rightarrow$ Konto, Konto $\rightarrow$ teure Op

Potentialmethode

- Potential $\Phi(A) \geq 0$ abhängig vom Zustand von A
- amort. Kosten = tats. Kosten + $\Phi(A_{\text{nach}}) - \Phi(A_{\text{vor}})$

Amortisierte Analyse

Aufgabe

- Kugeln (drei Farben) in einer Reihe
- Operationen
  - `insert`(pos, color) in $\Theta(1)$

Aggregatmethode

- Gesamtkosten / Anzahl Ops

Charging

- Kostentoken: teuere Op $\rightarrow$ günstige Op

Kontomethode

- günstige Op $\rightarrow$ Konto, Konto $\rightarrow$ teure Op

Potentialmethode

- Potential $\Phi(A) \geq 0$ abhängig vom Zustand von A
- amort. Kosten = tats. Kosten + $\Phi(A_{\text{nach}}) - \Phi(A_{\text{vor}})$

Amortisierte Analyse

Aufgabe

- Kugeln (drei Farben) in einer Reihe
- Operationen
  - `insert`(pos, color) in $\Theta(1)$

Aggregatmethode

- Gesamtkosten / Anzahl Ops

Charging

- Kostentoken: teuere Op $\rightarrow$ günstige Op

Kontomethode

- günstige Op $\rightarrow$ Konto, Konto $\rightarrow$ teure Op

Potentialmethode

- Potential $\Phi(A) \geq 0$ abhängig vom Zustand von A
- amort. Kosten = tats. Kosten + $\Phi(A_{\text{nach}}) - \Phi(A_{\text{vor}})$

Amortisierte Analyse

Aufgabe

- Kugeln (drei Farben) in einer Reihe
- Operationen
  - `insert`(pos, color) in $\Theta(1)$

Aggregatmethode
- Gesamtkosten / Anzahl Ops

Charging
- Kostentoken: teuere Op $\rightarrow$ günstige Op

Kontomethode
- günstige Op $\rightarrow$ Konto, Konto $\rightarrow$ teure Op

Potentialmethode
- Potential $\Phi(A) \geq 0$ abhängig vom Zustand von A
- amort. Kosten = tats. Kosten + $\Phi(A_{\text{nach}}) - \Phi(A_{\text{vor}})$

Amortisierte Analyse

Aufgabe

- Kugeln (drei Farben) in einer Reihe
- Operationen
  - `insert`(pos, color) in $\Theta(1)$

Aggregatmethode
- Gesamtkosten / Anzahl Ops

Charging
- Kostentoken: teuere Op $\rightarrow$ günstige Op

Kontomethode
- günstige Op $\rightarrow$ Konto, Konto $\rightarrow$ teure Op

Potentialmethode
- Potential $\Phi(A) \geq 0$ abhängig vom Zustand von A
- amort. Kosten = tats. Kosten + $\Phi(A_{\text{nach}}) - \Phi(A_{\text{vor}})$

Amortisierte Analyse

Aufgabe

- Kugeln (drei Farben) in einer Reihe
- Operationen
  - `insert`(pos, color) in $\Theta(1)$
  - `removeMost`(): Lösche eine Farbe mit den meisten Kugeln in $\Theta(n)$

Aggregatmethode
- Gesamtkosten / Anzahl Ops

Charging
- Kostentoken: teuere Op $\rightarrow$ günstige Op

Kontomethode
- günstige Op $\rightarrow$ Konto, Konto $\rightarrow$ teure Op

Potentialmethode
- Potential $\Phi(A) \geq 0$ abhängig vom Zustand von A
- amort. Kosten = tats. Kosten + $\Phi(A_{\text{nach}}) - \Phi(A_{\text{vor}})$

Amortisierte Analyse

Aufgabe

- Kugeln (drei Farben) in einer Reihe
- Operationen
  - `insert`(pos, color) in $\Theta(1)$
  - `removeMost`(): Lösche eine Farbe mit den meisten Kugeln in $\Theta(n)$

Aggregatmethode

- Gesamtkosten / Anzahl Ops

Charging

- Kostentoken: teuere Op $\rightarrow$ günstige Op

Kontomethode

- günstige Op $\rightarrow$ Konto, Konto $\rightarrow$ teure Op

Potentialmethode

- Potential $\Phi(A) \geq 0$ abhängig vom Zustand von A
- amort. Kosten = tats. Kosten + $\Phi(A_{\text{nach}}) - \Phi(A_{\text{vor}})$

Amortisierte Analyse

Aufgabe

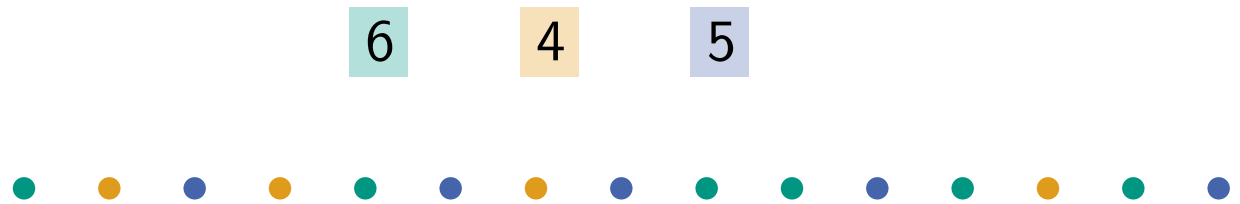
- Kugeln (drei Farben) in einer Reihe
- Operationen
  - `insert`(pos, color) in $\Theta(1)$
  - `removeMost`(): Lösche eine Farbe mit den meisten Kugeln in $\Theta(n)$

Aggregatmethode
- Gesamtkosten / Anzahl Ops

Charging
- Kostentoken: teuere Op $\rightarrow$ günstige Op

Kontomethode
- günstige Op $\rightarrow$ Konto, Konto $\rightarrow$ teure Op

Potentialmethode
- Potential $\Phi(A) \geq 0$ abhängig vom Zustand von A
- amort. Kosten = tats. Kosten + $\Phi(A_{\text{nach}}) - \Phi(A_{\text{vor}})$



Amortisierte Analyse

Aufgabe

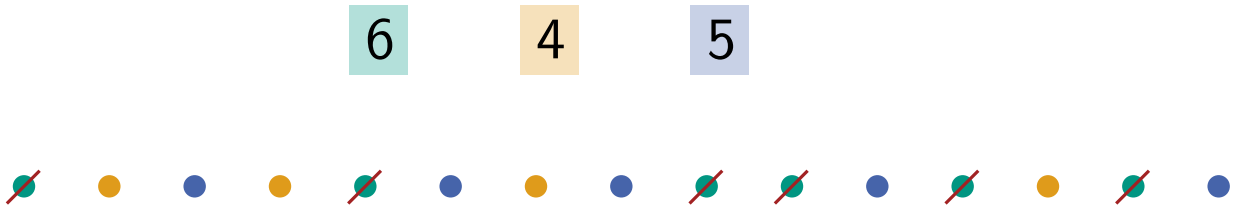
- Kugeln (drei Farben) in einer Reihe
- Operationen
  - `insert`(pos, color) in $\Theta(1)$
  - `removeMost`(): Lösche eine Farbe mit den meisten Kugeln in $\Theta(n)$

Aggregatmethode
- Gesamtkosten / Anzahl Ops

Charging
- Kostentoken: teuere Op $\rightarrow$ günstige Op

Kontomethode
- günstige Op $\rightarrow$ Konto, Konto $\rightarrow$ teure Op

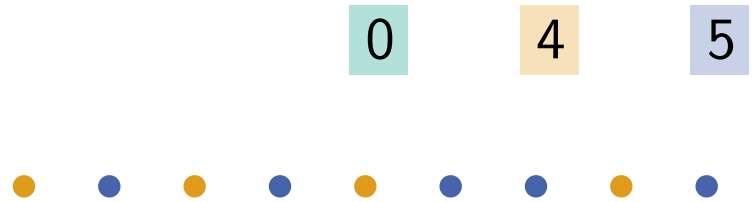
Potentialmethode
- Potential $\Phi(A) \geq 0$ abhängig vom Zustand von A
- amort. Kosten = tats. Kosten + $\Phi(A_{\text{nach}}) - \Phi(A_{\text{vor}})$



Amortisierte Analyse

Aufgabe

- Kugeln (drei Farben) in einer Reihe
- Operationen
  - `insert`(pos, color) in $\Theta(1)$
  - `removeMost`(): Lösche eine Farbe mit den meisten Kugeln in $\Theta(n)$



Aggregatmethode
- Gesamtkosten / Anzahl Ops

Charging
- Kostentoken: teuere Op $\rightarrow$ günstige Op

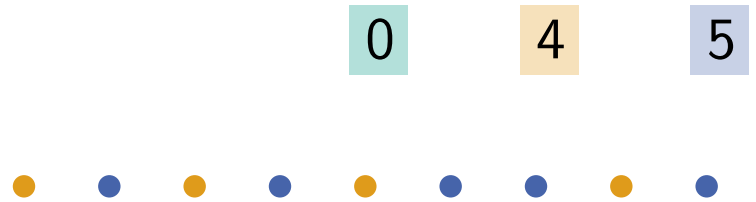
Kontomethode
- günstige Op $\rightarrow$ Konto, Konto $\rightarrow$ teure Op

Potentialmethode
- Potential $\Phi(A) \geq 0$ abhängig vom Zustand von A
- amort. Kosten = tats. Kosten + $\Phi(A_{\text{nach}}) - \Phi(A_{\text{vor}})$

Amortisierte Analyse

Aufgabe

- Kugeln (drei Farben) in einer Reihe
- Operationen
  - `insert`(pos, color) in $\Theta(1)$
  - `removeMost`(): Lösche eine Farbe mit den meisten Kugeln in $\Theta(n)$
- Einsicht: immer $\geq \frac{n}{3}$ gelöscht



Aggregatmethode
- Gesamtkosten / Anzahl Ops

Charging
- Kostentoken: teuere Op $\rightarrow$ günstige Op

Kontomethode
- günstige Op $\rightarrow$ Konto, Konto $\rightarrow$ teure Op

Potentialmethode
- Potential $\Phi(A) \geq 0$ abhängig vom Zustand von A
- amort. Kosten = tats. Kosten + $\Phi(A_{\text{nach}}) - \Phi(A_{\text{vor}})$

Bonusblatt

- besteht nur aus Bonuspunkten
- ab 14 Uhr auf der Webseite
- Aufgaben zu vielen verschiedenen Themen
- Abgabe normal im ILIAS
- nur garantierte Korrektur zwischen 110 und 130 Punkten
- trotzdem gute Übung für die Klausur!