ETH zürich

Übung 11 – Dynamische Programmierung

Informatik II

29. / 30. April 2025

Heutiges Programm

- Mars-Mission
- Der Weg des Königs
- Der Weg des gierigen Königs

Letzte Woche: Dynamische Programmierung

Gibt es Fragen?

- Tribonacci
- Catalan-Zahlen
- Blöcke
- Aufgabeplanung v2.0

1. Mars-Mission

Problem Beschreibung

Die Aufgabe betrifft einen Rover namens Perseverance, der auf dem Mars an einer mit S (Start) bezeichneten Position gelandet ist.

S	9	2	5	11	8
17	21	32	5	15	3
2	2	3	8	1	5
8	2	8	11	15	9
0	5	3	10	4	Z

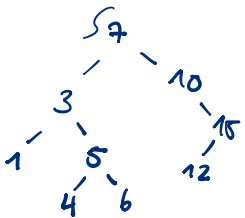
Problem Beschreibung

Der Rover will eine Zielposition Z (Ende) erreichen

S	9	2	5	11	8
17	21	32	5	15	3
2	2	3	8	1	5
8	2	8	11	15	9
0	5	3	10	4	Z

Problem Beschreibung

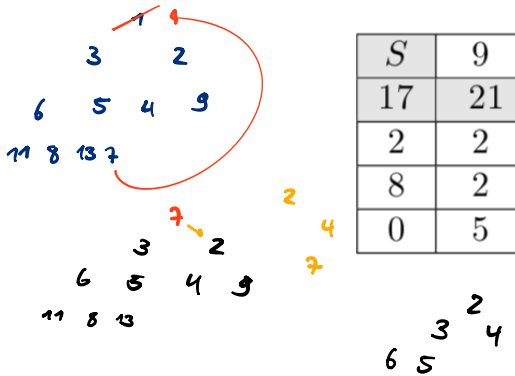
Der Rover kann sich nur in Richtung Osten (rechts) oder Süden (unten) bewegen.



S	9	2	5	11	8
17	21	32	5	15	3
2	2	3	8	1	5
8	2	8	11	15	9
0	5	3	10	4	Z

Problem Beschreibung

Jede Zelle des Gitters enthält einen Wert, der eine Gesteinsprobe darstellt, die vom Rover gesammelt werden kann, wenn er diese Zelle passiert.



S	9	2	5	11	8
17	21	32	5	15	3
2	2	3	8	1	5
8	2	8	11	15	9
0	5	3	10	4	Z

Problem Beschreibung

Ziel ist es, einen Weg von S nach Z zu finden, der den Gesamtwert der gesammelten Gesteinsproben maximiert.

S	9	2	5	11	8
17	21	32	5	15	3
2	2	3	8	1	5
8	2	8	11	15	9
0	5	3	10	4	Z

Problem Beschreibung

- **Eingabe:** Eine $m \times n$ Matrix A

Problem Beschreibung

- **Eingabe:** Eine $m \times n$ Matrix A
- **Ausgabe 1:** Maximal möglicher Wert, der an Steinen gesammelt werden kann, die auf einem Süd-Ost-Pfad von $(0, 0)$ nach $(m-1, n-1)$ liegen.

Problem Beschreibung

- **Eingabe:** Eine $m \times n$ Matrix A
- **Ausgabe 1:** Maximal möglicher Wert, der an Steinen gesammelt werden kann, die auf einem Süd-Ost-Pfad von $(0, 0)$ nach $(m-1, n-1)$ liegen.
- **Ausgabe 2:** Der Pfad, der diesem Maximalwert entspricht.

Schritt 1: Teilprobleme identifizieren

Was sind die Teilprobleme?

Schritt 1: Teilprobleme identifizieren

Was sind die Teilprobleme?

Für $i < m$ und $j < n$ sei $S(i, j)$ der maximal mögliche Wert, der an Steinen gesammelt werden kann, die auf einem südöstlichen Weg von (i, j) nach $(m - 1, n - 1)$ liegen.

Schritt 2: Rekursion identifizieren

Welche Optionen haben Sie, wenn Sie sich an der Position (i, j) befinden?

Schritt 2: Rekursion identifizieren

Welche Optionen haben Sie, wenn Sie sich an der Position (i, j) befinden?

- Bewege dich nach Osten in Richtung $(i, j + 1)$, wenn $j < n - 1$.
- Bewege dich nach Süden in Richtung $(i + 1, j)$, wenn $i < m - 1$.

Schritt 2: Rekursion identifizieren

Definieren Sie eine Rekursion mittels der zuvor definierten Handlungsoptionen.

Schritt 2: Rekursion identifizieren

Definieren Sie eine Rekursion mittels der zuvor definierten Handlungsoptionen.

$$S(i,j)\begin{cases}\max(A[i+1,j]+S(i+1,j), & \\ \qquad A[i,j+1]+S(i,j+1)) & \text{if } i<m-1 \text{ and } j<n-1,\\ A[i+1,j]+S(i+1,j) & \text{if } i<m-1 \text{ and } j=n-1,\\ A[i,j+1]+S(i,j+1) & \text{if } i=m-1 \text{ and } j<n-1,\\ 0 & \text{if } i=m-1 \text{ and } j=n-1.\end{cases}$$

Schritt 3: Implementierung

Datenstruktur. Welche Datenstruktur benötigen Sie, um die Lösungen für die Teilprobleme zu speichern?

Schritt 3: Implementierung

Datenstruktur. Welche Datenstruktur benötigen Sie, um die Lösungen für die Teilprobleme zu speichern?

Eine Matrix `s` der Größe $m \times n$, wobei `s[i][j]` den maximal erreichbaren Wert der Gesteinsproben auf einem Süd-Ost-Weg von (i, j) nach $(m-1, n-1)$ enthält.

Schritt 3: Implementierung

Abhängigkeiten. Was benötigen Sie, um $S(i, j)$ zu berechnen?

Schritt 3: Implementierung

Abhängigkeiten. Was benötigen Sie, um $S(i, j)$ zu berechnen?
$S(i + 1, j)$ if $i < m - 1$ und $S(i, j + 1)$, if $j < n - 1$.

Schritt 3: Implementierung

Berechnungsreihenfolge. Wie sollten Sie S ausfüllen?

Schritt 3: Implementierung

Berechnungsreihenfolge. Wie sollten Sie S ausfüllen?

Zeile für Zeile von unten nach oben. Jede Zeile wird von rechts nach links ausgefüllt. Um den Basisfall abzudecken, setzen wir zunächst $S[m-1][n-1] = 0$.

Schritt 3: Implementierung

Extraktion der Lösung.

Schritt 3: Implementierung

Extraktion der Lösung. Der maximal erreichbare Wert der Gesteinsproben wird in dem Eintrag $S[0][0]$ gespeichert. Um den Weg selbst zu rekonstruieren:

- Beginnen wir mit dem Eintrag $S[0][0]$ und geben $(0, 0)$ aus.
- Als nächstes prüfen wir, ob $S[0, 0] = A[0, 1] + s[0, 1]$.
- Wenn ja, gehen wir nach Osten und fahren von dort aus fort
- Andernfalls ist $S[0, 0] = A[1, 0] + s[1, 0]$, also gehen wir nach Süden und machen von dort aus weiter.
- Die Rekonstruktion wird fortgesetzt, bis die untere rechte Ecke $(m - 1, n - 1)$ erreicht ist.

Schritt 3: Implementierung

Wie lange ist die Laufzeit?

Schritt 3: Implementierung

Wie lange ist die Laufzeit? Die Berechnung des Maximalwertes kann in der Zeit $\Theta(mn)$ durchgeführt werden

- Die Matrix S hat die Größe $m \cdot n$ und jeder Eintrag kann in der Zeit $\Theta(1)$ berechnet werden.

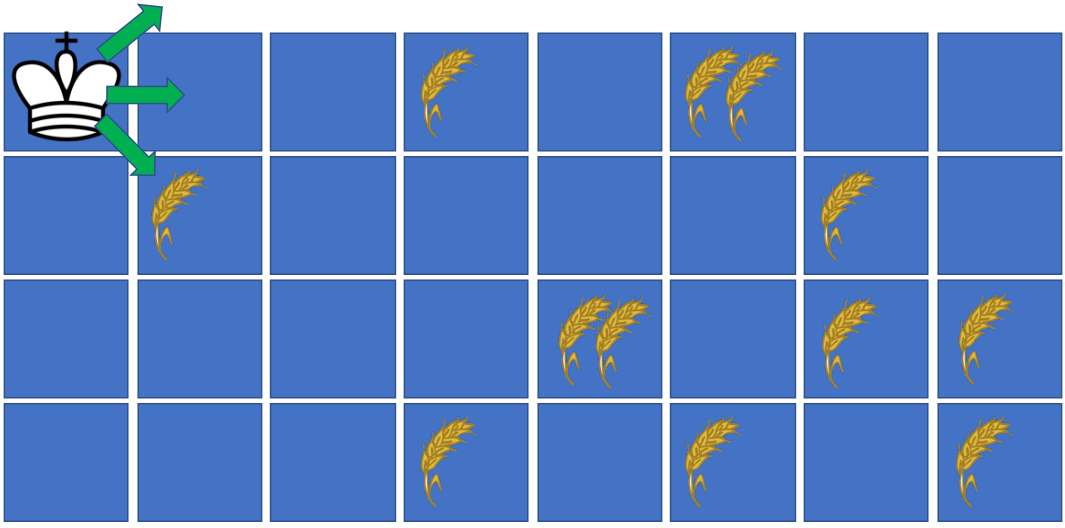
Die Rekonstruktion des gesamten Pfades kann in der Zeit $\Theta(m + n)$ erfolgen.

- Der rekonstruierte Weg hat die Länge $m + n - 1$, und an jeder Position können wir in der Zeit $\Theta(1)$ entscheiden, ob wir nach Süden oder Osten gehen müssen.

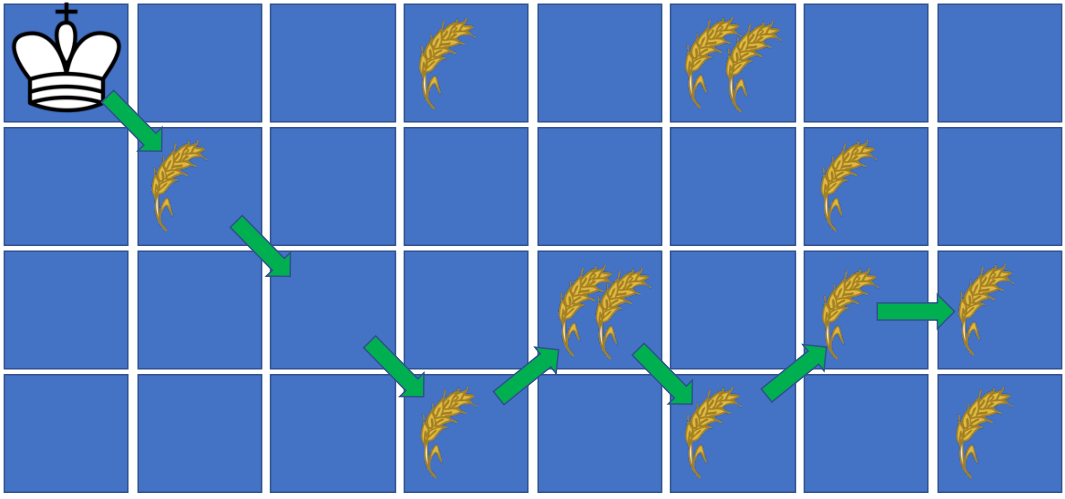
Zusammen mit der Zeit, die zum Füllen der Tabelle benötigt wird, ergibt sich die Gesamtlaufzeit von $\Theta(mn)$.

2. Königsweg

Problem Beschreibung



Problem Beschreibung



Problem Beschreibung

- Ein König befindet sich auf der Zelle $(0, 0)$ eines $m \times n$ Schachbretts.

Problem Beschreibung

- Ein König befindet sich auf der Zelle $(0, 0)$ eines $m \times n$ Schachbretts.
- Der König kann sich nur nordostwärts, ostwärts oder südostwärts bewegen. Das bedeutet, wenn der König sich in der Position (i, j) befindet, kann er nur zu $(i - 1, j + 1)$, $(i, j + 1)$, oder $(i + 1, j + 1)$ ziehen, solange er nicht das Spielfeld verlässt.

Problem Beschreibung

- Ein König befindet sich auf der Zelle $(0, 0)$ eines $m \times n$ Schachbretts.
- Der König kann sich nur nordostwärts, ostwärts oder südostwärts bewegen. Das bedeutet, wenn der König sich in der Position (i, j) befindet, kann er nur zu $(i - 1, j + 1)$, $(i, j + 1)$, oder $(i + 1, j + 1)$ ziehen, solange er nicht das Spielfeld verlässt.
- Jede Zelle ist mit einer Belohnung markiert.

Problem Beschreibung

- Ein König befindet sich auf der Zelle $(0, 0)$ eines $m \times n$ Schachbretts.
- Der König kann sich nur nordostwärts, ostwärts oder südostwärts bewegen. Das bedeutet, wenn der König sich in der Position (i, j) befindet, kann er nur zu $(i - 1, j + 1)$, $(i, j + 1)$, oder $(i + 1, j + 1)$ ziehen, solange er nicht das Spielfeld verlässt.
- Jede Zelle ist mit einer Belohnung markiert.
- Welches ist der beste Weg, den der König gehen kann, um die maximale Belohnung zu erhalten?

Problem Beschreibung

- Ein König befindet sich auf der Zelle $(0, 0)$ eines $m \times n$ Schachbretts.
- Der König kann sich nur nordostwärts, ostwärts oder südostwärts bewegen. Das bedeutet, wenn der König sich in der Position (i, j) befindet, kann er nur zu $(i - 1, j + 1)$, $(i, j + 1)$, oder $(i + 1, j + 1)$ ziehen, solange er nicht das Spielfeld verlässt.
- Jede Zelle ist mit einer Belohnung markiert.
- Welches ist der beste Weg, den der König gehen kann, um die maximale Belohnung zu erhalten?
- Belohnungen sind nicht negativ.

Problem Beschreibung

- **Eingabe:** Eine 2D-Liste `a` der Grösse $m \times n$ von nicht negativen Ganzzahlen.

Problem Beschreibung

- **Eingabe:** Eine 2D-Liste `a` der Grösse $m \times n$ von nicht negativen Ganzzahlen.
- **Ausgabe:** Der maximale Wert, den der König erreichen kann, und die Sequenz der Züge, um ihn zu erreichen.

Schritt 1: Teilprobleme identifizieren

Was sind die Teilprobleme?

Schritt 1: Teilprobleme identifizieren

Was sind die Teilprobleme?

Für $i < m$ und $j < n$, sei $S(i, j)$ der maximale Wert, den der König erreichen kann, wenn er von (i, j) startet.

Schritt 2: Rekursion identifizieren

Was sind meine Optionen?

Schritt 2: Rekursion identifizieren

Was sind meine Optionen?

- Bei der Lösung von $S(i, j)$, für $j < n - 1$, muss ich wählen, ob ich den König nordostwärts, ostwärts oder südostwärts bewege.
- Drei Optionen:
  - Ziehe nach $(i - 1, j + 1)$, wenn $i > 0$.
  - Ziehe nach $(i, j + 1)$.
  - Ziehe nach $(i + 1, j + 1)$, wenn $i < m - 1$.

Schritt 2: Rekursion identifizieren

Verwenden Sie die Optionen, um eine Rekursion zu definieren.

Schritt 2: Rekursion identifizieren

Verwenden Sie die Optionen, um eine Rekursion zu definieren.

$$S(i,j) = \begin{cases} a[i][j] & j = n-1. \\ a[i][j] + \max S(i-1, j+1), S(i, j+1) & i = m-1,\, j < n \\ a[i][j] + \max S(i, j+1), S(i+1, j+1) & i = 0,\, j < n \\ a[i][j] + \max \left\{ \begin{array}{l} S(i-1, j+1), \\ S(i, j+1), \\ S(i+1, j+1) \end{array} \right\} & 0 < i < m-1,\, j < n \end{cases}$$

Schritt 3: Implementierung

Datenstruktur. Welche Datenstruktur benötigen Sie, um die Lösungen für die Teilprobleme zu speichern?

Schritt 3: Implementierung

Datenstruktur. Welche Datenstruktur benötigen Sie, um die Lösungen für die Teilprobleme zu speichern?

Eine Matrix `s` der Grösse $m \times n$, wobei `s[i][j]` $S(i, j)$ enthält, für $i < m$ und $j < n$.

Schritt 3: Implementierung

Abhängigkeiten. Was benötigen Sie, um $S(i, j)$ zu berechnen?

Schritt 3: Implementierung

Abhängigkeiten. Was benötigen Sie, um $S(i, j)$ zu berechnen?
$S(i, j + 1)$, $S(i - 1, j + 1)$ wenn $i > 0$, und $S(i + 1, j + 1)$ wenn $i < m - 1$.

Schritt 3: Implementierung

Berechnungsreihenfolge. Wie sollten Sie `s` ausfüllen?

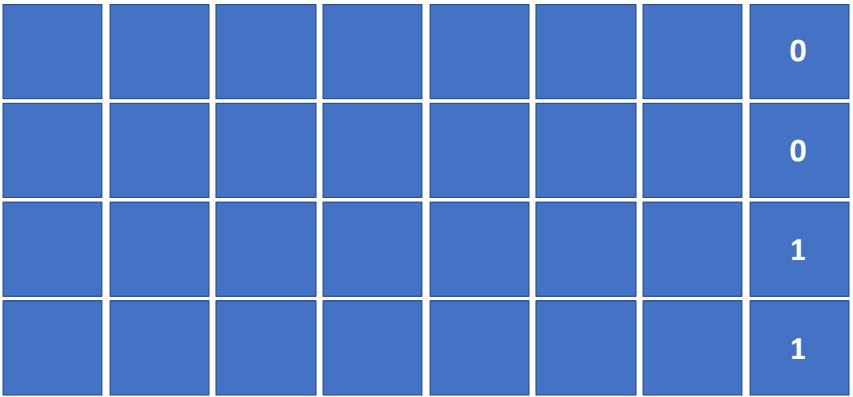
Schritt 3: Implementierung

Berechnungsreihenfolge. Wie sollten Sie `s` ausfüllen?

Spalte für Spalte von rechts nach links. Jede Spalte kann in beliebiger Reihenfolge ausgefüllt werden.

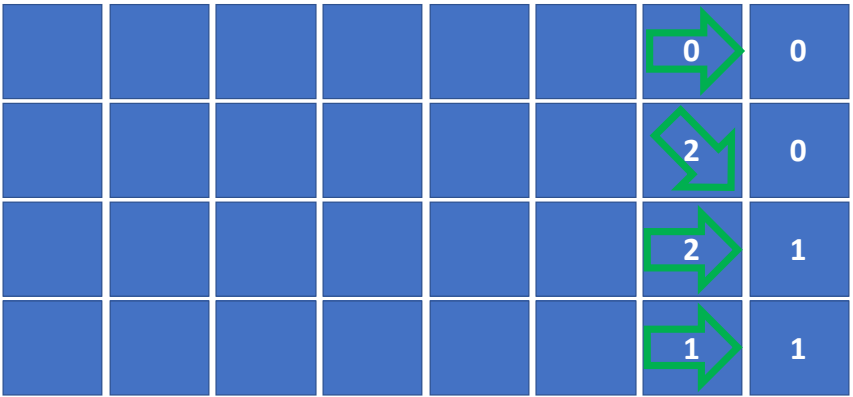
Schritt 3: Implementierung

Füllen Sie die Matrix `s` aus. Markieren Sie für jedes Teilproblem die optimale Entscheidung.



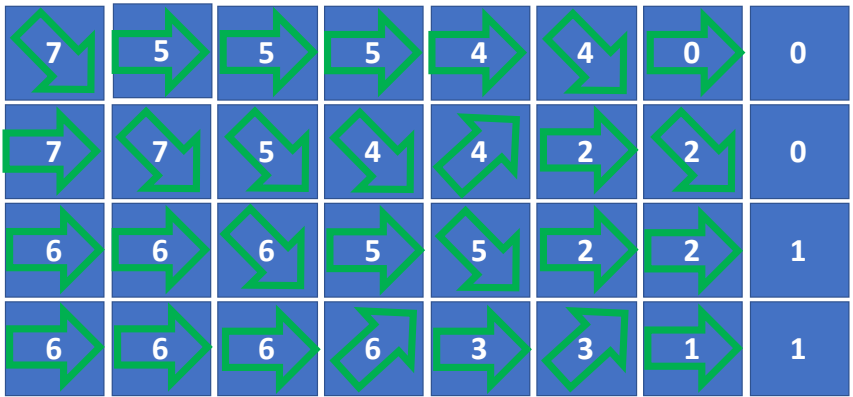
Schritt 3: Implementierung

Füllen Sie die Matrix `s` aus. Markieren Sie für jedes Teilproblem die optimale Entscheidung.



Schritt 3: Implementierung

Füllen Sie die Matrix `s` aus. Markieren Sie für jedes Teilproblem die optimale Entscheidung.



Schritt 3: Implementierung

Python-Implementierung im Code-Expert.

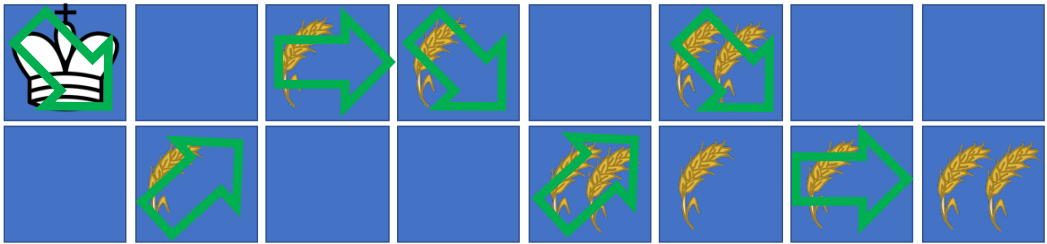
3. Gieriger Königsweg

Problem Beschreibung

Was ist, wenn Sie nur 2 Zeilen haben?

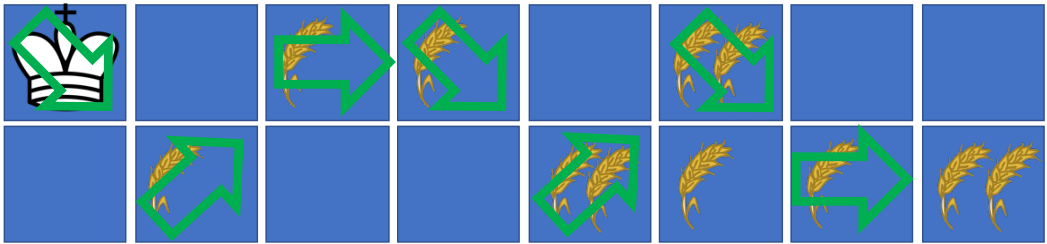
Problem Beschreibung

Brauchen Sie wirklich DP?



Problem Beschreibung

Brauchen Sie wirklich DP?



Nein! Im Allgemeinen, wenn die optimal lokale Lösung zu einer global optimalen Lösung führt, dann reicht ein gieriger Algorithmus aus!

Gierige Lösung

Wenn der König sich in Position (i, j) befindet, ...

Gierige Lösung

Wenn der König sich in Position (i, j) befindet, ...

- ... schaut der König auf $a[0][j+1]$ und $a[1][j+1]$ und
- ... geht er zur Zelle, die den höchsten Wert hat.

Diese lokal optimalen Entscheidungen führen zu einer global optimalen Lösung!

Gierige Lösung

Strategie: Verwenden Sie eine for-Schleife, um den König zur nächsten Zelle mit dem höchsten Wert zu bewegen.

Was ist die Invariante für diese gierige Lösung?

Nach Iteration $j = 0, 1, \ldots, n-2$,

Gierige Lösung

Strategie: Verwenden Sie eine for-Schleife, um den König zur nächsten Zelle mit dem höchsten Wert zu bewegen.

Was ist die Invariante für diese gierige Lösung?

Nach Iteration $j = 0, 1, \ldots, n-2$,

- `value` enthält den maximalen Wert, den der König nach $j+1$ Zügen erreichen kann.

Gierige Lösung

Strategie: Verwenden Sie eine for-Schleife, um den König zur nächsten Zelle mit dem höchsten Wert zu bewegen.

Was ist die Invariante für diese gierige Lösung?

Nach Iteration $j = 0, 1, \ldots, n-2$,

- `value` enthält den maximalen Wert, den der König nach $j+1$ Zügen erreichen kann.
- `path` enthält die Züge, die der König machen muss, um diesen Wert zu erreichen.

Gierige Lösung

Strategie: Verwenden Sie eine for-Schleife, um den König zur nächsten Zelle mit dem höchsten Wert zu bewegen.

Was ist die Invariante für diese gierige Lösung?

Nach Iteration $j = 0, 1, \ldots, n-2$,

- `value` enthält den maximalen Wert, den der König nach $j+1$ Zügen erreichen kann.
- `path` enthält die Züge, die der König machen muss, um diesen Wert zu erreichen.
- `(k_i, j)` beschreibt die Position des Königs.

Implementierung

Python-Implementierung im Code-Expert.

4. Homework

Übung 9: Dynamische Programmierung

On https://expert.ethz.ch/enrolled/SS25/mavt2/exercises

- Mission Mars with lava
- Binomial coefficients
- Triangle
- Longest common substring

Abgabedatum: Montag 28.04.2025, 20:00 MEZ **NO HARDCODING**