

Übungsserie 9

Datenstrukturen & Algorithmen

Universität Bern
Frühling 2018

Testat

- > **Wenn es knapp wird ...**
 - Es gibt Bonusserie für zusätzliche Punkte

 - > **Wenn es **dennoch** knapp wird ...**
 - Kontaktiert uns (Fall zu Fall Beurteilung)
 - Z.B. Zusatzaufgabe

 - > **Überprüft Testatliste auf Ilias!**
-

Übungsserie 9

- > **Dynamische Programmierung**
 - > 4 theoretische Aufgaben
 - > 3 praktische Aufgaben
 - Content-aware image resizing (Seam Carving)

 - > Poolstunde 30. April 1700 – 1800
-

CUT-ROD

- > **Aufgabe 1** Zeige, dass `CUT-ROD` naiv implementiert exponentielle Zeitkomplexität hat

- > **Tipps**
 - Induktionsbeweis
 - Hintergrund: Vorlesung ab Folie 8, Buch Kapitel 15.1

Fibonacci

- > **Aufgabe 2a** Pseudocode, um n -te Fibonacci-Zahl mit dynamischer Programmierung in $O(n)$ zu berechnen
- > **Aufgabe 2b** Teilproblem-Graph zeichnen (Buch s. 371)

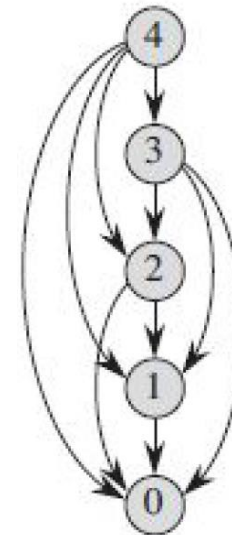
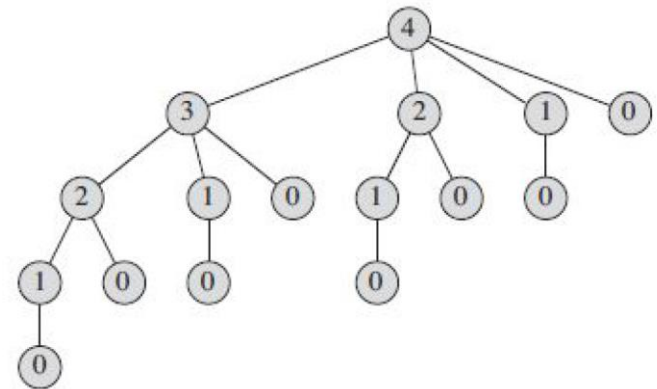
$$F_0 = 0$$

$$F_1 = 1$$

$$F_{k+1} = F_k + F_{k-1}$$

Teilproblem-Graph

- > **Rekursionsbaum** Ein rekursives Problem hat einen Rekursionsbaum, hier am Beispiel CUT-ROD
- > **Teilproblem-Graph** Jedes Teilproblem ist nur einmal verzeichnet
 - Jeder Knoten stellt ein Teilproblem dar
 - Jede (gerichtete) Kante stellt eine direkte Abhängigkeit dar



LCS

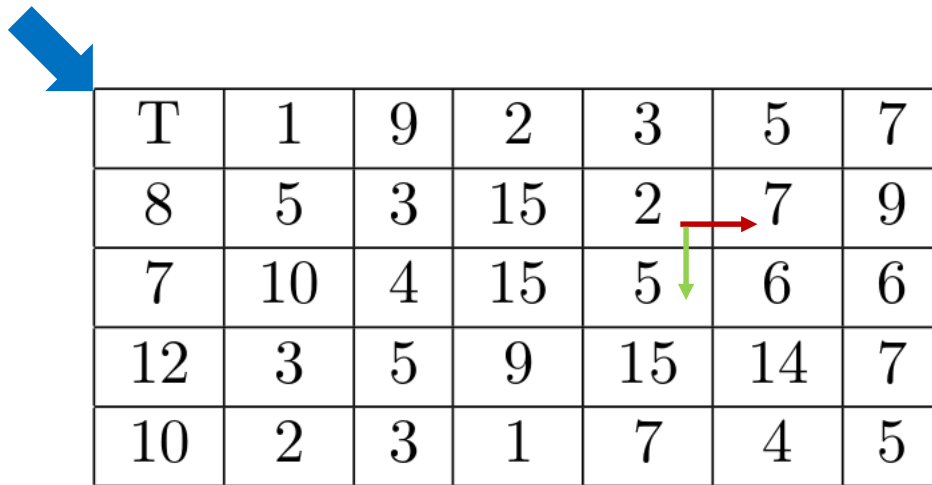
- > **Aufgabe 3** Bestimme längste gemeinsame Teilsequenz (LCS) zweier Sequenzen.

- > **Tipps**
 - Benutze in der Vorlesung besprochenen Algorithmus, baue Tabelle $c[i, j]$ (siehe Vorlesungsfolien)
 - Buch Kapitel 15.4

New York Cab

> Aufgabe 4 Finde günstigsten Pfad durch Strassengitter

Start



T	1	9	2	3	5	7
8	5	3	15	2	7	9
7	10	4	15	5	6	6
12	3	5	9	15	14	7
10	2	3	1	7	4	5

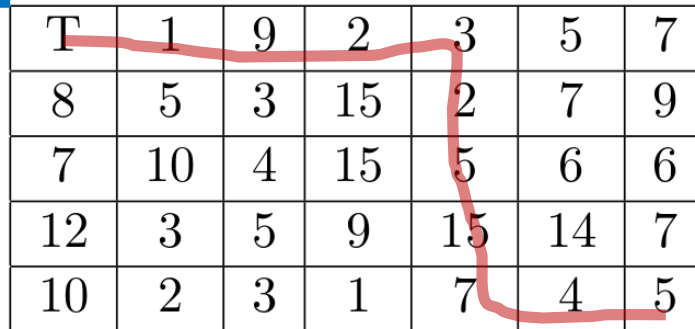
Ziel

> Einbahnstrassen! Erlaubte Züge:

- Horizontal nach rechts →
- Vertikal nach unten ↓

New York Cab

Start



T	1	9	2	3	5	7
8	5	3	15	2	7	9
7	10	4	15	5	6	6
12	3	5	9	15	14	7
10	2	3	1	7	4	5

Kosten = 53

Ziel

- > **Kosten eines Pfades** Summe aller vorkommenden Felder
- > **Tipps**
 - Sehr ähnliche wie LCS
 - **Teilproblem** Pfad mit minimalen Kosten bis und mit Feld $[i, j]$

Seam Carving

- > **Problemstellung** Ein zu breites Bild (1) soll verkleinert werden
 - Skalieren verzerrt Bildinhalt (2)
 - Cropping entfernt Bildinhalt (3)
 - **Seam Carving** Content-aware image resizing (4)



(1)



(2)



(3)



(4)

Seam Carving

- > **Algorithmus** Entferne iterativ «Seams»
 - **Seam** 1-Pixel-breiter Pfad vom oberen zum unteren Bildrand
 - Entferne in jeder Iteration den Seam mit dem geringsten Content (der am wenigsten wichtige Seam)



Seam Carving

- > **Kostenfunktion** Jedem Pixel werden Kosten $e(x, y)$ zugewiesen
- > $e(x, y)$
 - Pixel, welche sich wenig von der Umgebung unterscheiden erhalten **tiefe** Kosten
 - Pixel, welche sich stark von der Umgebung unterscheiden erhalten **hohe** Kosten
- > **Optimaler Seam** Pfad mit geringsten Kosten



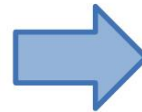
Seam Carving

> Teilaufgaben

- Implementiere `computeCosts()` für die Berechnung der Kostentabelle M (Code für die Berechnung der per-pixel Kosten $e(x, y)$ wird vorgegeben)
- Implementiere `computeSeam()` für die Berechnung des optimalen Seams



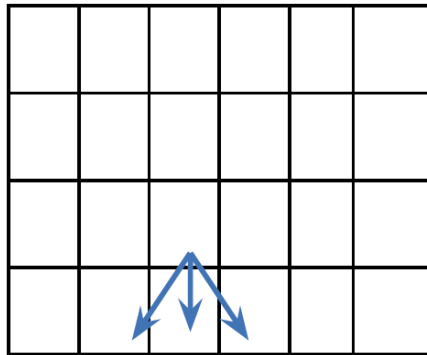
Input



Output

Seam Carving

- > **Detail** Ein Seam darf von Zeile zu Zeile *um höchstens 1 Pixel* verschoben sein!



- > **Seamkosten** Summe der Kosten aller besuchten Pixel
 - Vgl. LCS & theoretische Aufgabe 4