Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement Informatik
Markus Püschel David Steurer Peter Widmayer
Chih-Hung Liu

20. November 2017

Algorithmen & Datenstrukturen Blatt 9 HS 17

Übungsstunde (Raum & Zeit): _____

Abgegeben von: _____

Korrigiert von: _____

erreichte Punkte: ____

Hinweis zur Beschreibung eines dynamischen Programms: Eine vollständige Beschreibung eines dynamischen Programms behandelt **immer** die folgenden Aspekte (interessant auch für die Klausur!):

1) *Definition der DP-Tabelle:* Welche Dimensionen hat die DP-Tabelle $DP[\ldots]$?
Was ist die Bedeutung jedes Eintrags (in klar formulierten Worten)?

2) *Berechnung eines Eintrags:* Welche Werte der Tabelle werden wie initialisiert? Wie berechnet sich ein Eintrag aus den Werten von anderen Einträgen? Welche Einträge hängen nicht von anderen Einträgen ab?

3) *Berechnungsreihenfolge*: In welcher Reihenfolge kann man die Einträge berechnen, so dass die jeweils benötigten anderen Einträge bereits vorher berechnet wurden?

4) *Auslesen der Lösung:* Wie lässt sich die Lösung am Ende aus der Tabelle auslesen?

Die Laufzeit eines dynamischen Programms berechnet sich üblicherweise einfach aus der Grösse der Tabelle multipliziert mit dem Aufwand, jeden Eintrag zu berechnen. Manchmal überwiegt jedoch auch der Aufwand um die Lösung auszulesen.

Aufgabe 9.1 *Seam Carving.*

Angenommen, Sie schiessen in Ihrem Urlaub ein Bild im Breitformat (n Pixel hoch, m Pixel breit, $n < m$), welches Sie auf Social Media in einem *quadratischen* Format veröffentlichen möchten. Skalieren oder Zuschneiden kommt nicht in Frage – schliesslich sollen alle relevanten Bildinhalte unverzerrt und komplett wiedergeben werden. Deshalb beschliessen Sie, ihr Bild mit der sogenannten Seam Carving Methode[1] zu bearbeiten: In total $k := m - n$ Runden wird das Urlaubsfoto in ein neues Bild vom Format n Pixel hoch, n $(= m - k)$ Pixel breit verändert. Die Runden verlaufen alle ähnlich, wir betrachten deshalb nur die erste dieser Runden.

In Runde 1 soll aus jeder Zeile des Bildes genau ein Pixel entfernt werden. Dieser Pixel darf jedoch nicht zu wichtig für den Bildinhalt sein! Als Grundlage erhalten Sie eine $n \times m$ Tabelle A, welche für jeden Pixel (in Zeile i und Spalte j) angibt, wie sehr er den benachbarten Pixeln zur Linken und zur Rechten ähnelt (mit Tabelleneinträgen $A[i, j] \in \mathbb{N}$ mit folgender Bedeutung: kleiner Wert $A[i, j] \cong$ hohe Ähnlichkeit $\cong$ geringer Unterschied zu den seitlichen Nachbarn).

[1] Shai Avidan und Ariel Shamir. *Seam carving for content-aware image resizing.* SIGGRAPH 2007.

Abbildung 1: a) Originales Bild, b) Skaliert, c) Zugeschnitten, d) mit Seam Carving.
Bilder von Wikimedia Commons, User Newton2, https://de.wikipedia.org/wiki/Inhaltsabhängige_Bildverzerrung

Gleichzeitig ist darauf zu achten, dass sich Zeilen mit der Zeit nicht zu stark gegeneinander verschieben. Deshalb gilt in jeder Runde folgende Regel: Wird in Zeile i der Pixel in Spalte j gelöscht, so muss der in Zeile $i+1$ gelöschte Pixel aus einer der Spalten $j-1, j, j+1$ stammen.

Geben Sie einen Algorithmus der *dynamischen Programmierung* an, der basierend auf Tabelle A gemäss der obigen Vorgabe aus jeder der n Zeilen einen Pixel (i, j_i) löscht, sodass die Summe $\sum_{i=1}^{n} A[i, j_i]$ minimalen Wert hat. Gesucht ist einerseits der minimale Wert, andererseits sollen am Schluss aus der DP-Tabelle auch die zu löschenden Pixel ausgelesen werden. Geben Sie die Laufzeit Ihres Verfahrens in Abhängigkeit von n und m an.

Aufgabe 9.2 *Hase Hoppel.*

Hase Hoppel steht auf einer Linie, gegeben als x-Achse, auf Position s und möchte zur Position t hüpfen. Zur Verfügung steht ihm eine Menge S von n Sprungweiten $S = \{s_1, s_2, \ldots, s_n\}$, mit Werten $s_1, s_2, \ldots, s_n, s, t \in \mathbb{N}$. Hoppel darf jede dieser Sprungweiten in jede Richtung höchstens einmal benutzen. Gibt es zwei Teilmengen $A, B \subseteq S$, sodass unser Hase von s nach t gelangt, wenn er zuerst mit den Sprungweiten aus A nach rechts hüpft (um insgesamt $+\sum_{s_i \in A} s_i$) und danach mit Sprungweiten aus B nach links (um insgesamt $-\sum_{s_j \in B} s_j$)?

Geben Sie einen möglichst effizienten Algorithmus der *dynamischen Programmierung* an, der berechnet, ob es zwei solche Teilmengen $A, B \subseteq S$ gibt. Wie lassen sich diese Teilmengen A und B (falls sie existieren) aus Ihrer DP-Tabelle auslesen? Ist die Laufzeit Ihres Algorithmus polynomiell (mit Begründung)?

Abgabe: Am Montag, den 27. November 2017 zu Beginn Ihrer Übungsgruppe.