Karlsruher Institut für Technologie

Institut für Theoretische Informatik
T.T.-Prof. Dr. Thomas Bläsius
https://scale.iti.kit.edu

Übungsblatt 13

Algorithmen I – Sommersemester 2023

Abgabe im ILIAS bis 28.07.2023, 18:00 Uhr

Die Abgabe erfolgt als *eine* PDF-Datei über das Übungsmodul in der Gruppe deines Tutoriums im ILIAS.
Beachte bitte die Hinweise zum Bearbeiten auf der Webseite.

Aufgabe 1 - Hochstapler (5 Punkte)

Die Sache spitzt sich zu. Diverse Geheimdienste intrigieren einander und im ganzen Land bricht Chaos aus: Biber-Agenten gegen Adler-Agenten, überall Maulwürfe, Zombiber gegen Dachse an den Bahnhöfen.

Als die ~~Faultiere~~ Biber im Spaßbad davon Wind bekommen, wird der Urlaub rasch beendet. Sie beschließen dem Wahnsinn ein Ende zu setzen und planen die Tiere des Waldes (und der Luft und der Erde) gegen einen gemeinsamen Feind zu vereinen: Dr. Meta. Der Plan ist simpel: nördlich der Alpen wird ein riesiger Damm gebaut. In einem Land ohne Wasser kann Dr. Meta mit seinen Dämmen nichts mehr anfangen.

Wie zuvor wird der Damm in Schichten gebaut. Allerdings fällt ab und zu auf, dass eine Schicht aus schlechtem Holz besteht. Dann müssen die Schichten darüber abgebaut und die schlechte Schicht ersetzt werden, bevor es mit dem Bau weiter gehen kann. Der Dammbau wird mit einer Datenstruktur verwaltet, die folgende Operationen unterstützt:

- `build`: Fügt in Zeit $\Theta(1)$ eine neue Schicht zum Damm hinzu.
- `dismantle`(k): Entfernt die obersten $k > 0$ Schichten vom Damm. Dafür wird Zeit $\Theta(k)$ benötigt.

Diese Operationen werden auf zwei Biber aufgeteilt: Jean-Claude führt `build` aus und Justin kümmert sich um `dismantle`.

a) Biber Justin beschwert sich, dass Jean-Claude am Damm viel weniger Arbeit hat als er: Es gibt sehr viele teure `dismantle`-Operationen obwohl `build` immer billig ist! Gib für jedes $n \in \mathbb{N}$ eine Abfolge von $\Theta(n)$ Operationen an, sodass

- es unter den `dismantle`-Operationen $\Theta(\sqrt{n})$ viele gibt, die Laufzeit $\Theta(\sqrt{n})$ haben und
- `dismantle`-Operationen niemals mit dem gleichen k aufgerufen werden.

(2 Punkte)

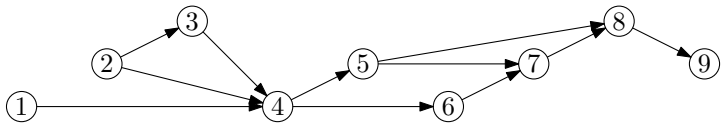
b) Beruhige Justin, indem du zeigst, dass in jeder beliebigen Abfolge von `build`- und `dismantle`-Operationen jede einzelne nur amortisiert konstante Laufzeit hat. (3 Punkte)

Hinweis: Verwende dazu eine der vorgestellten Methoden für amortisierte Analyse aus der Vorlesung.

Aufgabe 2 - Viele Wege führen zur Senke? (5 Punkte)

Gegeben sei ein gerichteter, azyklischer Graph G mit genau einer Senke t. Mithilfe dynamischer Programmierung soll nun für jede Quelle q rausgefunden werden, wie viele Pfade es von q zu t gibt. Dabei machen wir Gebrauch von der topologischen Sortierung des Graphen: Wir nummerieren die Knoten des Graphen in dieser Reihenfolge durch und legen ein Array C an welches einen Eintrag pro Knoten enthält. In diesem Array sollen die Teillösungen verwaltet werden.

a) Gib für die Quellen 1 und 2 in folgendem Graphen jeweils an, wie viele Pfade zur Senke 9 führen. (1 Punkt)



b) Beschreibe für jeden Knoten i was an Stelle $C[i]$ gespeichert werden soll und stelle darauf aufbauend die Rekurrenz auf. (3 Punkte)

c) Gib an, wie man aus dem Array C schließlich für eine gegebene Quelle schließlich die Anzahl aller Pfade von q zu t bestimmen kann. (1 Punkt)

Aufgabe 3 - IKEA (10 Punkte)

Für das Großprojekt Dammbau brauchen die Biber eine Menge Holz, das von der Biber-Community gespendet und in der IKEA (Intelligente Klein- und Echtholz Annahmestelle) zwischengelagert wird, bevor es zur Baustelle transportiert wird. Da die Biber auf der Baustelle aber nicht besonders fleißig sind und auch noch ständig Schichten wieder abbauen müssen, geht bei der IKEA langsam der Platz für das ganze Holz aus. Zum Glück gibt es noch einen schmalen Flur, den die Biber auch noch mit Holz vollstellen können! Die Biber haben den Flur komplett mit Paletten ausgelegt, wobei zwei Reihen mit je n Paletten in den Flur passen. Da auch die Biber, die Paletten herstellen, nicht

besonders gewissenhaft sind, sind manche Paletten besser als andere und halten unterschiedlich viel aus. Die Palette (i, j) (Reihe $i \in \{1, 2\}$, Spalte $j \in \{1, \ldots, n\}$) kann dabei höchstens $h(i, j) > 0$ Einheiten Holz tragen. Außerdem soll der Flur natürlich auch noch benutzbar sein und die Biber wollen noch von einem Ende zum anderen kommen. Dabei können die Biber nur von einer Palette zu einer (nicht diagonal) benachbarten Palette laufen. Paletten, auf denen Holz steht, sind nicht benutzbar.

1	1	1	1
1	1	1	1

Beispiele für Verteilungen von Holz, die nicht erlaubt sind.

2	3	5	1	7
3	2	3	4	2

Eine erlaubte Verteilung von Holz, mit der man 12 Holzeinheiten lagern kann. Diese Verteilung ist aber *nicht* optimal!

Leider sind die Biber in der Annahmestelle nicht so intelligent, wie der Name suggestiert und sie brauchen deine Hilfe! Wie viel Einheiten Holz können maximal im Flur der IKEA gelagert werden, wenn der Flur auch noch benutzbar sein soll?

a) Gebe eine (möglichst kleines) Instanz an, bei der es in jeder optimalen Lösung eine Spalte gibt, in der gar kein Holz gelagert wird. (1 Punkt)

b) Zeige oder widerlege: Es gibt eine Instanz, bei der es in jeder optimalen Lösung zwei *aufeinanderfolgende* Spalten gibt, in denen gar kein Holz gelagert wird. (1 Punkt)

Wir wollen das Problem mit einem DP lösen und die Teillösungen in einem zweidimensionalen Array H mit zwei Zeilen und $n + 1$ Spalten speichern.

c) Wie sollen die Einträge $H[1][0]$ und $H[2][0]$ initialisiert werden? Beschreibe außerdem, was in dem Eintrag $H[i][j]$ mit $i \in \{1, 2\}$ und $j \in \{1, \ldots, n\}$ gespeichert werden soll. (1 Punkt)

d) Stelle eine Rekurrenzgleichung für den Eintrag $H[i][j]$ auf. (3 Punkte)
Hinweis: Welche Einträge links von Spalte j sind relevant, wenn auf Palette (i, j) Holz gelagert wird? Überlege dazu, auf welchen Paletten links von Spalte j dann kein Holz gelagert werden darf.

e) Beschreibe, wie man mit H die maximale Anzahl an Holzeinheiten, die im Flur gelagert werden kann, bestimmen kann. (1 Punkt)

f) Welche Informationen müssen bei der Berechnung von H zusätzlich gespeichert werden, damit eine Lösung rekonstruiert werden kann? Beschreibe außerdem einen Algorithmus, der die Lösung mit Hilfe der zusätzlichen Informationen in $O(n)$ Laufzeit rekonstruiert. (3 Punkte)