

Übungsaufgaben – Blatt 9

Zürich, 18. Januar 2006

Zusammenfassung und Aufgaben

Die Aufgabenstellungen, die wir bisher betrachtet haben, zeichnen sich dadurch aus, dass wir für eine gegebene Problem Instanz (eine konkrete Fragestellung) eine Lösung (richtige Antwort) berechnen sollen. Wir kennen also von Anfang an alle Informationen (die gesamte Eingabe), die man zur Berechnung der Lösung braucht. Es gibt in der Praxis auch viele Aufgaben, bei denen man am Anfang nur einen Teil der Information über eine Problem Instanz zur Verfügung hat und eine Teillösung berechnen und umsetzen muss, bevor man den nächsten Teil der Eingabe erhält.

Wir veranschaulichen dies an einem Beispiel. Wir stellen uns ein Dienstleistungszentrum vor, zum Beispiel ein Notarztzentrum mit 50 fahrenden Ärzten. Jeder Arzt verfügt über einen eigenen Krankenwagen. Wenn ein Anruf in der Zentrale eingeht, muss einer der Ärzte zum Unfallort oder zu der Wohnung des Patienten fahren, um dort die Behandlung aufzunehmen. Die Zentrale kann versuchen, die Fahrten so zu steuern, dass gewisse Parameter optimiert werden. Zum Beispiel kann versucht werden,

- die mittlere Wartezeit auf die ärztliche Hilfe so klein wie möglich zu halten,
- die längste mögliche Wartezeit so klein wie möglich zu halten, oder
- die Fahrtkosten und damit die Gesamtlänge der gefahrenen Strecken zu minimieren.

Um dies zu erreichen, hat die Zentrale mehrere Entscheidungsmöglichkeiten: Soll zu einem neuen Notfall ein Arzt aus dem Krankenhaus (der Zentrale) fahren oder ein Arzt, der gerade an einem anderen (vielleicht nahegelegenen) Ort seine Arbeit abgeschlossen hat? Soll ein Arzt nach der abgeschlossenen Behandlung eines Patienten zurück zur Zentrale fahren oder dort, wo er ist, auf weitere Einsätze warten oder sogar eine neue strategische Position einnehmen?

In der klassischen Problemformulierung sind vorher alle Unfallorte, Unfallzeitpunkte und notwendigen Behandlungszeiträume bekannt, und wir sollen eine Lösung (eine Zuteilung der Aufgaben auf die einzelnen Ärzte) suchen, die den gewählten Parameter optimiert. Es liegt auf der Hand, dass man in der Realität dieses Wissen über die Zukunft nicht haben

kann. Die Zentrale kann nicht ahnen, wann und wo der nächste Unfall passieren wird, und trotzdem fordern wir, dass die Zentrale eine, wenn schon nicht optimale, dann zumindest gute Entscheidungsstrategie verfolgt. Solche Problemstellungen nennen wir *Online-Probleme*, und die zugehörigen Lösungsstrategien werden *Online-Algorithmen* genannt.

Von solchen Situationen wie der oben beschriebenen kann man sich viele vorstellen, zum Beispiel die Steuerung einer Taxizentrale oder einer Polizeistation. Ähnlich geht es auch vielen Betrieben, die ihren Kundenaufträgen Kapazitäten (Mitarbeiter und Maschinen) zur Bearbeitung zuteilen müssen, ohne zu ahnen, wie viele, in welcher Weise strukturierte, wie dringende oder wie lukrative neue Aufträge in der nächsten Zeit ankommen werden. Solche Probleme nennt man *Arbeitsplanungsprobleme* („*scheduling*“ im englischen). Diese Online-Probleme sind meistens sehr schwer, weil die Zukunft sehr ungünstig verlaufen kann. Es kann zum Beispiel passieren, dass man einen Arzt zur Zentrale zurückzieht und direkt, nachdem er dort angekommen ist, ein Notruf von einem unmittelbaren Nachbarn des soeben behandelten Patienten eingeht. Unter diesen Bedingungen eine gute Strategie zu finden, kann sehr schwer sein, und oft ist es auch so, dass es gar keine Strategie gibt, die für jeden möglichen Verlauf der Zukunft eine akzeptable Lösung finden kann. Die Kunst der Algorithmik ist es, zu entdecken, für welche Online-Probleme gute Strategien (Online-Algorithmen) überhaupt existieren, und wann wir keine Chance haben, gute Entscheidungen für alle zukünftigen Fälle zu treffen. Es ist aber auch nicht selten so, dass man auf den ersten Blick meinen würde, gegen die unbekannte Zukunft gar nicht spielen zu können, und dennoch gelingt es, Online-Algorithmen zu entwerfen, die für jeden Verlauf der Zukunft annähernd optimale Lösungen effizient finden können. Dies sind die kleinen Wunder in diesem Teil der Algorithmik.

Ein Beispiel für ein solches kleines Wunder wollen wir im folgenden vorstellen. Wir betrachten Online-Probleme als Optimierungsprobleme.

Definition: Sei I eine Instanz eines Optimierungsproblems U . Mit $\text{Opt}_U(I)$ bezeichnen wir die Kosten der optimalen Lösungen für I . Wenn man die ganze Eingabe kennt, kann man hypothetisch immer eine optimale Lösung finden. Sei A ein Online-Algorithmus für U . Wir bezeichnen mit $\text{Lösung}_A(I)$ die von A berechnete Lösung für die Problem Instanz I und mit

$$\text{Kosten}(\text{Lösung}_A(I))$$

die Kosten von $\text{Lösung}_A(I)$. Wir nehmen an, dass U ein Minimierungsproblem ist, dass man also eine zulässige Lösung für I mit minimalen Kosten sucht.

Ein Online-Algorithmus A kann nicht garantieren, eine optimale Lösung zu berechnen, weil A Entscheidungen treffen muss, bevor A die ganze Problem Instanz (alle Anforderungen) kennt. Die Güte von A messen wir daran, welche Nähe zu einem Optimum von A garantiert werden kann.

Die *Konkurrenzgüte* $\text{Konk}_A(I)$ des Online-Algorithmus A auf der Instanz I ist definiert als die Zahl

$$\text{Konk}_A(I) = \frac{\text{Kosten}(\text{Lösung}(I))}{\text{Opt}_U(I)} .$$

Damit beschreibt $\text{Konk}_A(I)$, wieviel mal schlechter die von A berechnete Lösung für I

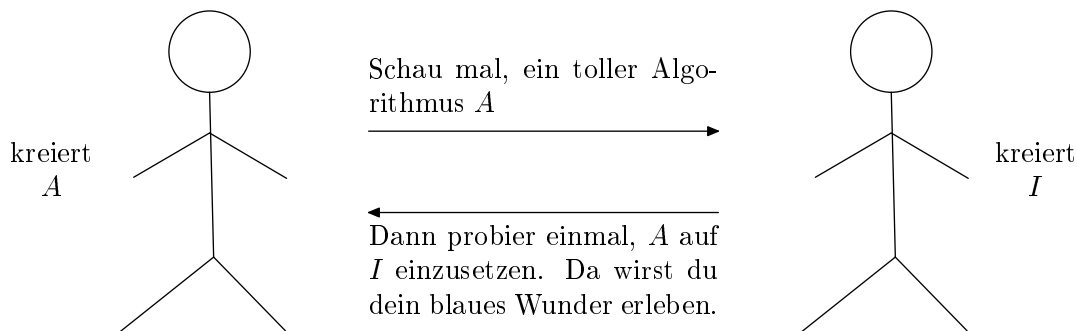


Abbildung 1:

gegenüber einer optimalen Lösung ist.

Aufgabe 24

Für jede Probleminstanz I und jeden Online-Algorithmus A für U haben wir $\text{Opt}_U(I)$ und $\text{Kosten}(\text{Lösung}_A(I))$ definiert. Was drückt die Zahl

$$\frac{\text{Kosten}(\text{Lösung}_A(I)) - \text{Opt}_U(I)}{\text{Opt}_U(I)} \cdot 100$$

aus?

10 Punkte

Wir sagen, dass A ein δ -konkurrenzfähiger Online-Algorithmus für U ist, wenn für alle Instanzen I von U

$$\text{Konk}_A(I) \leq \delta$$

gilt.

Nehmen wir jetzt an, wir entwerfen eine Online-Strategie für ein Optimierungsproblem U und wollen $\text{Konk}_A(I)$ bestimmen. In der Informatik sagt man, dass wir die Konkurrenzgüte von A *analysieren* wollen. Dies ist manchmal ein sehr schweres Vorhaben. Wir kennen viele Fälle (Probleme und Online-Algorithmen zu ihrer Lösung), in denen wir auch nach jahrelangen Versuchen den Wert von $\text{Konk}_A(I)$ nicht annähernd bestimmen können. Die Schwierigkeit hängt damit zusammen, dass man den maximalen Wert von $\text{Konk}_A(I)$ über alle potentiell unendlich vielen Probleminstanzen I bestimmen soll. Für die Analyse von $\text{Konk}_A(I)$ haben sich Forscher ein hilfreiches Spiel zwischen einem Algorithmen-Designer und einem Gegner ausgedacht. Der Algorithmen-Designer versucht, einen Online-Algorithmus zu entwerfen, und sein Gegner versucht, durch Schaffung von schweren Probleminstanzen für die entworfenen Algorithmen zu beweisen, dass diese schlecht sind (vgl. Abbildung 1).

In diesem Spiel kann man den Algorithmen-Designer als Optimisten betrachten, der sich über das Produkt seiner Arbeit freut. Seinen Gegner kann man als Pessimisten betrachten, der alle Produkte des Algorithmen-Designers anzweifelt und deren Nutzen zu widerlegen versucht. Gute Forschungsteams brauchen beide Arten von Mitarbeitern, Optimisten wie

Pessimisten. So kommen begeisterte Ideen zustande, die sorgfältig überprüft, korrigiert und letztendlich verbessert werden.

Bei der Analyse von Online-Algorithmen betrachten wir den Gegner als *gmein*. Dies resultiert aus seiner vorteilhaften Situation. Er kennt den Online-Algorithmus A und kann nun die Zukunft so gestalten, dass A nicht erfolgreich ist. Weil er A kennt, weiss der Gegner für jede Teilinstanz ganz genau, welche Teillösung A liefert. Damit sieht das Spiel wie folgt aus. Der Gegner zeigt einen Teil der Zukunft und wartet ab, was A damit macht. Danach kreierte der Gegner ein weiteres Stück der Zukunft. Nachdem er sich angesehen hat, wie A darauf reagiert, denkt er sich die nächsten Anforderungen aus. Der Gegner hat also gute Möglichkeiten, A an der Nase herumzuführen. Wenn ihm dies gelingt, hat er damit bewiesen, dass Konk_A nicht sehr gut sein kann.

Wir präsentieren jetzt ein Beispiel für ein Online-Problem, das sogenannte *Problem des Blätterns* (*Blätter-Problem*, engl. „Paging“): Innerhalb eines Rechners sind die Daten in Einheiten, den sogenannten *Seiten*, gespeichert. Ein Rechner hat zwei Arten von Speichern, einen kleinen Speicher, den *Cache* mit Platz für k Seiten, in dem wir einen sehr schnellen Zugriff haben, und einen riesigen *Hauptspeicher* (*HS*), aus dem wir nicht direkt lesen können (vgl. Abbildung 2). Wenn wir Daten aus dem HS ansehen wollen, müssen wir sie zunächst aus dem HS in den Cache übertragen und dann aus dem Cache lesen. Der Cache ist meistens voll, und wenn wir eine Seite aus dem HS dort hinbringen müssen, so muss zunächst im Cache Platz geschaffen werden, d.h. eine Seite aus dem Cache muss in den HS zurückgeschickt werden. Wir bezeichnen mit

$$i \leftrightarrow j$$

die Aktion, bei der die Seite i im Cache gegen die Seite j im HS ausgetauscht werden muss. Weil die Kosten für den Zugriff auf den Cache sehr gering sind, betrachten wir das Lesen aus dem Cache als kostenlos (Kosten 0). Der Aktion

$$i \leftrightarrow j$$

ordnen wir die Kosten 1 zu.

Wir betrachten nun als Beispiel die folgende Situation: Wir haben einen Cache der Grösse 3 ($k = 3$), der genau die drei Seiten 5, 30 und 107 enthält. Jetzt betrachten wir die schon vollständig bekannte Probleminstanz $I = (3, 107, 30, 1201, 73, 107, 30)$. Die optimale Lösung hat Kosten 3: Der Rechner holt zuerst die Seite 3 in den Cache und schickt dafür die Seite 5 zurück in den Hauptspeicher. Die nächsten angeforderten Seiten 107 und 30 stehen schon im Cache zur Verfügung. Danach holt man die Seite 1201 auf den Speicherplatz der Seite 5 in den Cache. Die Seiten 30 und 107 werden nicht aus dem Cache entfernt, weil wir wissen, dass sie später noch gebraucht werden. Im sechsten Schritt tauschen wir die Seite 1201 gegen die Seite 73 aus. Die letzten zwei Anforderungen können wir dann offensichtlich ohne zusätzlichen Aufwand realisieren. Diese Lösung können wir als

$$5 \leftrightarrow 3, \bullet, \bullet, 3 \leftrightarrow 1201, 1201 \leftrightarrow 73, \bullet, \bullet$$

beschreiben, wobei \bullet einen Schritt beschreibt, in dem kein Austausch zwischen Cache und Hauptspeicher stattfindet.

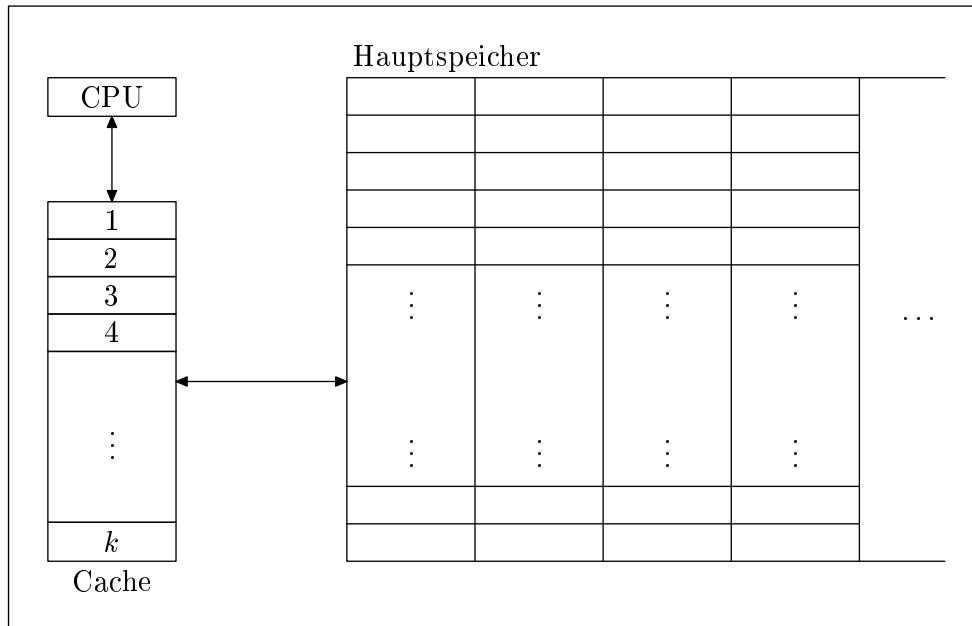


Abbildung 2:

Aufgabe 25

Finden Sie optimale Lösungen für die folgenden Instanzen des Blätter-Problems:

- (a) Es sei $k = 3$, der Cache enthält die Seiten 1, 2 und 3, und $I = (7, 9, 3, 2, 14, 8)$.
- (b) Es sei $k = 5$, der Cache enthält die Seiten 1, 101, 1001, 1002 und 9, und $I = (1002, 7, 5, 1001, 101, 3, 8, 1, 1002)$.

5+5 Punkte

Es gibt manchmal viele mögliche Lösungen (bei jeder neuen Anforderung einer Seite, die nicht im Cache vorhanden ist, hat man k Möglichkeiten, eine im Cache vorhandene Seite gegen die neue auszutauschen), und die Lage kann sehr unübersichtlich sein, aber mit Sicherheit besteht die Möglichkeit, eine optimale Lösung zu finden. Wenn man aber das Blätter-Problem als ein Online-Problem betrachtet, ändert sich die Situation auf extreme Weise. Die Anforderungen kommen einzeln. Erst nachdem man eine Anforderung erfüllt, und dafür möglicherweise eine Seite aus dem Cache entfernt hat, wird die nächste Anforderung bekanntgegeben. Der Gegner hat hier einen grossen Spielraum und kann richtiggehend gemein werden. Er stellt als nächste Anforderung gerade die Seite, die man soeben entfernt hat. Egal, welche Strategie man für den Seitenaustausch verwendet, der Gegner findet immer genau die Seite, die man gerade aus dem Cache entfernt hat. Damit muss es bei der Bearbeitung einer von dem Gegner hergestellten Instanz bei jeder Anforderung zu einem Austausch zwischen dem Cache und dem Hauptspeicher kommen und somit sind die Kosten die maximal möglichen.

Wir spielen dies an einem Beispiel durch: Sei $k = 3$, und der Cache enthalte die Seiten 1, 2 und 3. Dann fordert der Gegner die Seite 4 an. Eine Seite muss also aus dem Cache entfernt werden. Nehmen wir an, die betrachtete Strategie entfernt die Seite 2, um dafür die Seite 4 in den Cache zu laden. Dann fordert der Gegner im nächsten Schritt die Seite 2 an. Die Seite 2 ist nicht mehr im Cache vorhanden, muss also erneut aus dem HS geholt werden. Nehmen wir an, die Online-Strategie führt den Austausch $4 \leftrightarrow 2$ durch. Dann fordert der Gegner im nächsten Schritt die Seite 4 an. Falls sich die Strategie für den Austausch $1 \leftrightarrow 4$ entscheidet und diesen durchführt, dann fordert der Gegner die Seite 1 an, die zum Beispiel durch den Austausch $4 \leftrightarrow 1$ geholt werden kann. Somit entstehen die Instanz

$$(4, 2, 4, 1)$$

und ihre zulässige Lösung

$$2 \leftrightarrow 4, 4 \leftrightarrow 2, 1 \leftrightarrow 4, 4 \leftrightarrow 1$$

mit maximal möglichen Kosten 4. Eine optimale Lösung für diese Instanz wäre aber

$$3 \leftrightarrow 4, \bullet, \bullet, \bullet$$

mit Kosten 1.

Aufgabe 26

Betrachten Sie die folgende Online-Strategie: Es wird immer die Seite mit der kleinsten Nummer aus dem Cache entfernt. Nehmen wir an, dass $k = 4$ und dass der Cache die Seiten 1, 3, 5 und 7 enthält. Spielen Sie die Rolle des Gegners, und entwerfen Sie eine Problem Instanz mit 10 Anforderungen (also eine Instanz der Länge 10), so dass diese Online-Strategie eine Lösung mit den Kosten 10 produziert. **10 Punkte**

Bonus-Aufgabe 10

Ein Online-Algorithmus nehme immer diejenige Seite aus dem Cache heraus, die bisher am wenigsten nachgefragt wurde. Wenn mehrere solche Seiten vorhanden sind, dann wird diejenige mit der grössten Nummer entfernt. Spielen Sie den Gegner für diese Strategie, und entwerfen Sie jeweils eine Instanz, für die diese Strategie zu maximalen Kosten führt, für folgende Startsituationen und Zielsetzungen:

- (a) Sei $k = 4$, der Cache enthalte die Seiten 1, 2, 3 und 4, und die Instanz I soll die Länge 4 haben und es soll $\text{Opt}_{\text{Blättern}}(I) = 1$ gelten.
- (b) Sei $k = 5$, der Cache enthalte die Seiten 1, 7, 103, 5 und 9, und für die entworfene Instanz I soll $\text{Opt}_{\text{Blättern}}(I) = 2$ gelten.

5+5 Bonus-Punkte

Ihre Lösungen zu den Aufgaben können Sie entweder persönlich bei der Open-Class-Veranstaltung am 25. Januar 2006 abgeben oder bis zum 25. Januar per E-Mail (möglichst als PDF-Datei) an hjb@inf.ethz.ch oder per Post an folgende Adresse schicken:

Dr. Hans-Joachim Böckenhauer
Informationstechnologie und Ausbildung
ETH Zentrum CAB F 11.1
Universitätsstrasse 6
8092 Zürich

Bitte vergessen Sie nicht, Ihre Lösung mit Ihrem Namen und Ihrer E-Mail-Adresse zu versehen.

Falls Ihre Lösung uns bis zum 25. Januar 2006 um 12.00 Uhr erreicht, können Sie die korrigierte Lösung bereits in der Veranstaltung am 25. Januar abholen. Ansonsten können Sie die Lösung ab dem 27. Januar bei Herrn Dr. Böckenhauer im Büro abholen. Auf Wunsch erhalten Sie die Lösung auch per Post zugeschickt, in diesem Fall vermerken Sie bitte Ihre vollständige Postadresse auf Ihrer Lösung.