einfach

INFORMATIK

Daten darstellen, verschlüsseln, komprimieren

Begleitband

7–9

SEKUNDARSTUFE I

JURAJ HROMKOVIČ

Klett und Balmer Verlag

Klett

einfach INFORMATIK 7–9

Daten darstellen, verschlüsseln, komprimieren

SEKUNDARSTUFE I

Begleitband

Autor
Juraj Hromkovič

Klett und Balmer Verlag

5 Suche und Ordnung in Daten

Schulbuch, Seite 70 bis 79
Kopiervorlage 9

In diesem Kapitel lernen die S, weshalb es sich lohnt, Ordnung in den Daten zu halten. Ohne systematische Datenorganisation gibt es keine schnelle Suche. Die S lernen die beiden bekanntesten Suchstrategien kennen: die binäre Suche und Hashing. Ausserdem erhalten sie einen Einblick in die Funktionsweise von Suchmaschinen.

Lernziele

Die Schülerinnen und Schüler …

- … entdecken, dass die Suche in unsortierten Daten unvergleichbar aufwändiger ist als in gut sortierten Daten.
- … können den Algorithmus der binären Suche in sortierten Folgen erfolgreich anwenden, um gewünschte Daten sehr schnell zu finden.
- … können die binäre Suche zur effizienten Sortierung von Zahlen in einer aufsteigenden Folge verwenden.
- … entdecken Hashing als Methode zur Abspeicherung von Daten, die eine extrem schnelle Suche ermöglicht. Sie können Hashing in konkreten Situationen anwenden, geeignete Hashfunktionen wählen und ihre Qualität mit Säulendiagrammen beurteilen.
- … verstehen, wie man Suchmaschinen baut, um so schnell wie möglich Anfragen nach Informationen beantworten zu können.

Kompetenzen des Lehrplans 21

MI 2.1 h > Dokumente so ablegen, dass auch andere sie wieder finden

MI 2.2 i > verschiedene Algorithmen zur Lösung desselben Problems vergleichen und beurteilen

MI 2.3 i > **grundsätzliche Funktionsweise von Suchmaschinen verstehen**

MI 2.3 j > lokale Geräte, lokales Netzwerk und das Internet als Speicherorte für private und öffentliche Daten unterscheiden

MI 2.3 m > Internet als Infrastruktur von seinen Diensten unterscheiden

Fachdidaktische Überlegungen

Das Thema dieses Kapitels liegt an der Grenze zwischen Datendarstellung, Datenorganisation und Algorithmen. Man könnte die Entwicklung von schnellen **Suchalgorithmen** vollständig in den Bereich des Algorithmenentwurfs einordnen, aber in diesem Fall würde man etwas Wesentliches verlieren. Es gibt keinen schnellen Algorithmus für die Suche in chaotisch abgelegten Daten, genauso wie keine schlaue Strategie zur Suche nach einer Nadel im Heuhaufen existiert. Um eine schnelle Suche zu ermöglichen, muss man zuerst dafür sorgen, dass die Daten überschaubar in einer gut überlegten **Ordnung** abgespeichert sind. Sich diese Ordnung gut zu überlegen ist das Mass für eine erfolgreiche und schnelle Suche nach konkreten Daten.

Die **binäre Suche** ist wahrscheinlich der am häufigsten verwendete Algorithmus, weil Computer heute überwiegend mit Suchaufgaben beauftragt werden. Den Aufwand (bzw. die Berechnungskomplexität) von Suchalgorithmen messen wir hier in der Anzahl der Speicherplätze, deren Inhalt man sich anschauen muss, um das Gesuchte zu finden. Eine Suche in einer unsortierten Zahlenfolge kann unvorhersehbar lange dauern, aber erwartungsgemäss entspricht der Suchaufwand in etwa der Hälfte der Länge der Folge. Mit binärer Suche in einer aufsteigenden Zahlenfolge ist das Anschauen von nur logarithmisch vielen Speicherplätzen nötig. Das heisst, die Suche ist exponentiell schneller als in einer unsortierten Folge. Die binäre Suche besteht aus einer Folge von einfachen Schritten. Man schaut sich den Inhalt des Speicherplatzes in der Mitte an und vergleicht diesen mit der gesuchten Zahl. Wenn die gesuchte Zahl grösser ist als die in der Mitte gespeicherte Zahl, dann wiederholt man die Suche mit den Zahlen rechts der Mitte, ansonsten mit den Zahlen links davon. Somit halbiert man mit jedem Schritt den Suchraum. Daher kommt auch der logarithmische Aufwand. Es ist wichtig zu verstehen, dass es nicht bloss ums Spielen mit Zahlen geht. Die Zahlen können als Dateinamen dienen, die man sucht. Man kann aber auch Texte als Dateinamen verwenden und sie lexikografisch wie in Wörterbüchern sortieren.

Es gibt keinen schnelleren Suchalgorithmus in **sortierten Folgen** als die binäre Suche. Diese funktioniert aber nur, wenn die Daten in einer Folge sortiert sind. Das ist nicht selbstverständlich. Nicht weil der Aufwand für das Sortieren zu gross wäre, aber heute verändern sich die Datensammlungen ununterbrochen. Neue Dateien kommen hinzu, alte werden gelöscht. Somit wächst der Aufwand, um die Daten nach jeder Änderung in einer sortierten Folge zu behalten. Dies war der Auslöser für das Konzept von **Hashing**. Man nutzt den Namen der Datei, um den Speicherort auszurechnen. Die Berechnung muss schnell gehen. Somit kann man augenblicklich vom Namen der gesuchten Datei den Speicherplatz ablesen. Das geht gut, solange keine Kollisionen auftreten, wenn mehrere Dateien im gleichen Speicherplatz landen sollten. Die S lernen in diesem Kapitel, unterschiedliche Wege zur Bestimmung des Speicherplatzes und ihre Güte in konkreten Situationen zu beurteilen. Somit können sie unterschiedliche Algorithmen (genannt Hashfunktionen) zur Berechnung der Speicheradresse aus dem Dateinamen vergleichen und eine gute Wahl einer Hashfunktion treffen.

Aufgrund der wilden und unkontrollierbaren Verteilung von Webseiten im Internet mit ständigen dynamischen Veränderungen (neue Rechner, neue Server, neue Verbindungen, neue Webseiten) kann man im Internet nicht für eine zentrale Ordnung beim Ablegen von Dateien sorgen. Wir erklären hier, wie Maschinen für die Suche nach gewissen Informationen gebaut werden. Das machen wir nur im Rahmen eines Einblicks, weil die detailliertere Beschreibung der Arbeitsweise von Suchmaschinen nicht nur geheim ist, sondern die dabei verwendeten Konzepte auch ziemlich anspruchsvoll sind.

Wir vermitteln hier die binäre Suche und Hashing auf eine Weise, dass diese Grundkonzepte allen S zugänglich werden und dass alle S beide Konzepte in einfachen, konkreten Situationen erfolgreich einsetzen können. Bei der Arbeit mit diesen Konzepten ist es wichtig, die reale Situation nachzuahmen. Der Computer kann nicht auf einmal den ganzen Inhalt des Speichers anschauen und danach entsprechend handeln. Er kann sich nur dazu entscheiden, jeweils einen Speicherplatz zu besuchen, diesen zu öffnen und die darin enthaltene Datei zu lesen. Danach muss er den Speicherplatz wieder schliessen und kann sich einen anderen Speicherplatz anschauen. Eine Möglichkeit, dies physisch nachzuahmen, ist, die Dateinamen auf Kärtchen zu schreiben und sie verdeckt auf den Tisch zu legen. Dann kann man die Suche im Speicher simulieren, indem man eine Karte umdreht, sie sich anschaut und sie dann wieder verdeckt zurücklegt.

Zur Durchführung

Schnelle Suche in sortierten Folgen

Hier machen wir zuerst darauf aufmerksam, wie viel Aufwand es bedeutet, in unsortierten Daten zu suchen. Weil die Datenmengen heutzutage riesig sind, ist der Vergleich mit der Suche nach einer Nadel im Heuhaufen nicht übertrieben. Somit motivieren wir die S, bei der Abspeicherung von Daten Ordnung zu halten.

Beispiel 1

Hier wird anhand eines konkreten Beispiels die Vorgehensweise bei der binären Suche vorgeführt. Wichtig ist dabei, die Suche mit verdeckten Karten zu spielen, weil der Computer immer nur einen einzelnen Speicherplatz aufsuchen kann, um den Inhalt anzuschauen.

Neue Konzepte

Der Algorithmus aus Beispiel 1 wird hier **binäre Suche** genannt. Zentral ist, dass das Anschauen des Inhalts eines Speicherplatzes den Suchraum halbiert.

1 Hier besteht natürlich die Möglichkeit, die binäre Suche mit verdeckten, aber sortierten Karten mehrfach durchzuführen.

Man kann die binäre Suche aber auch konsequent mit der offenen Zahlenfolge durchführen, nur um zu sehen, wie man sich langsam der gesuchten Zahl annähert. Für die Durchführung empfehlen wir die folgende Darstellung:

Speicheradresse	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Inhalt des Speicherplatzes	1	2	6	7	9	11	12	13	14	21	37	38	39	43	51

Wenn man mit offenen Karten sucht, kann man die Suche gut beschreiben. Man geht zum Speicherplatz 8, in dem die 13 liegt. Weil 37 grösser ist als 13, sucht man rechts der Position 8 weiter, also auf den Positionen 9 bis 15. In der Mitte zwischen 9 und 15 ist die Position 12. Man öffnet den Speicherplatz 12 und sieht die Zahl 38, die grösser ist als die gesuchte Zahl 37. Somit liegt die Zahl 37 auf der Position 9, 10 oder 11. Man schaut dementsprechend die Zelle 10 an und sieht die Zahl 21, die kleiner ist als 37. Somit muss die 37 auf der Position 11 liegen, vorausgesetzt, die 37 kommt in der Folge vor. Als letzten Schritt öffnet man die Adresse 11, um die Vermutung zu überprüfen.

Die Länge der Folge ist 15. Sie wurde absichtlich so gewählt, dass man bei der binären Suche die Mitte immer eindeutig bestimmen kann. Die nächste Zahl mit dieser Eigenschaft ist 31. Allgemein haben alle derartigen Zahlen die Form 2^{n-1}.

Es lohnt sich, mehrere solcher Beispiele durchzuführen. Ab und zu sollte es auch vorkommen, dass man die gesuchte Zahl früher entdeckt, weil sie in der Mitte eines untersuchten Intervalls liegt.

2 Auch ohne Logarithmen können die S die Frage durch Probieren beantworten. Nach dem Anschauen des Wortes in der Mitte reduziert sich der Suchraum von 1024 auf höchstens 1024 : 2 = 512 Adressen. Nach maximal zehn Schritten bleibt genau eine Adresse übrig. Somit muss man höchstens 11 Adressen anschauen. Allgemein bedeutet das, dass man höchstens $\log_2 n + 1$ Adressen anschaut.

Beispiel 2 Hier erklären wir, dass die binäre Suche auch dazu verwendet werden kann, eine neue Datei so in die Reihenfolge einzuordnen, dass die Folge sortiert bleibt.

3 Hier zeigen wir, dass man die binäre Suche auch mit Namen durchführen kann. Alle Objekte, für die eine lineare Ordnung existiert, können verwendet werden.

Basis und Weiterführung

Die binäre Suche in dieser Form ist allen S zugänglich und alle sollten fähig sein, die binäre Suche mit verdeckten Karten durchzuführen. Für die Förderung von stärkeren S kann man die binäre Suche zum Sortieren verwenden. Dazu nimmt man Karten mit beliebigen Nummern und legt irgendeine der Karten verdeckt auf den Tisch. Ein allgemeiner Schritt beim Einfügen einer neuen Karte sieht wie folgt aus: Auf dem Tisch liegen verdeckte Karten mit von links nach rechts aufsteigenden Werten. Man nimmt die nächste noch nicht eingeordnete Karte, findet ihren Platz mithilfe der binären Suche und ordnet die Karte entsprechend in die Reihenfolge ein. Am Ende entsteht eine sortierte Folge von Karten.

Hashing

Hashing ist eine Alternative zur binären Suche, die bei erfolgreicher Datenorganisation noch schneller sein kann als die binäre Suche. Hashing ist eines der fundamentalsten und meistangewandten Konzepte der Datenverwaltung.

Neue Konzepte

Hashing ist nur möglich, wenn man mit den Adressen der Plätze im Computerspeicher arbeitet. Die Idee ist, die Platzierung einer Datei im Speicher schnell vom Dateinamen ablesen zu können. Das geht gut, solange nicht zu viele Dateien dem gleichen Speicherplatz zugeordnet werden. Die Kunst, Hashing erfolgreich anzuwenden, liegt in der Wahl einer Hashfunktion, die alle Dateinamen gleichmässig auf die Speichereinheiten verteilt.

Ideale Datenorganisation

Die sorgfältig präsentierte Motivation für das Konzept von Hashing ist lang und somit nicht für alle S vollständig begreifbar. Die stärkeren S können von den vorgestellten Überlegungen aber stark profitieren. Die LP hat die Möglichkeit, gezielt mit physisch umgesetzten Beispielen schrittweise die Grundideen zu erklären. Das erste Beispiel, in dem eine Karte auf diejenige Speicheradresse gelegt wurde, die dem Kartenwert entspricht, ist sehr einfach. Man kann aber auch gut nachvollziehen, dass es nicht eine effiziente Speichernutzung ist, wenn man drei bis vier Karten auf tausend Adressen abspeichert und die restlichen Speicherplätze leer lässt.

Schwieriger ist es zu erklären, warum man bei dynamischen Veränderungen der Datensammlung zu viel Aufwand mit der Neuordnung der Daten hat. Der Aufwand entsteht nicht hauptsächlich durch ein Update der aktuellen Speicheradressen, sondern durch die Tatsache, dass jede Einfügung einer neuen Datei erwartungsgemäss die Umplatzierung der Hälfte aller Dateien verursacht. Beim Durchführen des Beispiels mit Karten kann man die Speicheradressen notieren und somit die Position der Speicherplätze festlegen. So können die S beim Einfügen einer neuen Karte mit binärer Suche gut beobachten, wie viele Karten man auf neue Speicheradressen verschieben muss.

Beispiel 3

Die LP kann dieses Beispiel illustrieren, indem sie mithilfe der S rund 100 Artikel abhängig von den letzten zwei Ziffern des EAN-Codes auf die Adressen zwischen 0 und 99 physisch verteilt. So wird deutlich, welcher Speicherplatz mit den meisten Artikeln belastet wurde.

Neue Konzepte

Die Idee von Hashing wird an dieser Stelle formuliert. Für die LP kann es nützlich sein zu wissen, was die tatsächlichen Vorteile von Hashing gegenüber der binären Suche aus der Sicht der Berechnungskomplexität (des Computeraufwands) sind:

Die besten Hashfunktionen verteilen m Dateien «zufällig» auf m Speicherplätze. Auch die besten Hashfunktionen können es nicht vermeiden, dem am stärksten belasteten Speicherplatz erwartungsgemäss rund $\log_2 m$ Dateien zuzuordnen. Wenn man eine von diesen $\log_2 m$ Dateien sucht, kostet das vergleichbar viel Aufwand wie die binäre Suche, weil man schlimmstenfalls alle Dateien an der Adresse anschauen muss, um die gesuchte zu finden. Trotzdem ist Hashing vorteilhaft, weil der Durchschnittsaufwand pro gesuchte Datei konstant ist (er wächst nicht mit m). Der Grund dafür ist, dass an den meisten Speicheradressen höchstens zwei Dateien liegen. Somit sind die Dateien relativ gleichmässig verteilt. Der andere bereits erwähnte Vorteil von Hashing gegenüber der binären Suche ist, dass Hashing effizient mit einer sich ständig ändernden Sammlung von Dateien arbeiten kann.

4 Dies ist eine Aufgabe, die alle S einzeln durchführen können. Die Artikel, die sie sich aussuchen, dürfen nicht nach Nummern ausgesucht werden. Sie müssen zuerst die Artikel wählen und dann die Nummer ablesen. Das Ziel der Aufgabe ist die Einführung von Säulendiagrammen, die dokumentieren, wie gut die Hashfunktion für die betrachtete Datensammlung ist.

Einblick in die Computer-Technologie

In diesem Abschnitt stellen wir die Dienstleistungen von Suchmaschinen vor. Der eigentliche Bau von Suchmaschinen ist so komplex, dass man an dieser Stelle nicht detailliert darauf eingehen kann. Ausserdem werden Suchmaschinen ununterbrochen modifiziert, um zu verhindern, dass jemand den Priorisierungsmechanismus so gut versteht, dass er daraus Profit schlagen kann. Hier sollen die S zunächst begreifen, dass man Preprocessing in der Form von Indexierung machen muss, um die Suche nach Webseiten mit den gewünschten Themen hinreichend schnell durchführen zu können. Zweitens machen wir darauf aufmerksam, welche Kriterien bei der Priorisierung der Webseiten eine Rolle spielen können.

Neue Konzepte

Die S sollen für konkrete Datenverteilungen selbstständig Säulendiagramme erstellen und dabei ein Gefühl dafür entwickeln, wann man eine Verteilung als gütig (gut geeignet für die Datenspeicherung) betrachten kann.

5 Die Aufgabe fokussiert auf den Vergleich unterschiedlicher Datenverteilungen, ohne dabei formal auf die Konzepte der beschreibenden Statistik einzugehen. Die S sollten beobachten, dass die Verteilung in Teilaufgabe C nach dem Geburtsjahr sehr ungünstig ist. Die Resultate in den Teilaufgaben A und B hängen von den tatsächlichen Geburtsdaten der Klasse ab. Erwartungsgemäss sollte die Verteilung nach Tagen (Teilaufgabe A) die günstigste sein. Bei einer Verteilung nach Monaten könnte es bereits vorkommen, dass es im Jahr gewisse Monate gibt, in welchen mehr Geburtstage vorkommen als in anderen.

6 In dieser Aufgabe versucht man die Häufigkeiten durch Geburtsjahre (davon gibt es wahrscheinlich nur zwei oder drei) und geburtsreichere Monate zu überwinden, indem man mit den Ziffern eine willkürliche Berechnung der Speicheradresse anstellt. Hier könnte man versuchen, durch unterschiedliche Zahlen (hier 23) zu teilen und die resultierenden Verteilungen zu vergleichen. Eine natürliche Möglichkeit ist die Klassengrösse. Man kann dabei beobachten, dass die Teilung durch Primzahlen eher zu besseren Datenverteilungen führt.

Basis und Weiterführung

Alle S sollen es schaffen, für konkrete Daten und gegebene Hashfunktionen die Datenverteilung zu bestimmen und durch Säulendiagramme zu visualisieren. Die stärkeren S kann die LP auffordern, das folgende Kriterium zur Bewertung der Datenverteilung zu betrachten und anzuwenden. Man kann jeder Datei den Aufwand zuordnen, den man bei gegebener Verteilung hat, um die Datei zu finden. Der Aufwand kann als die Höhe der Säule betrachtet werden, zu der die Datei gehört. Die Säule steht für die Anzahl aller Dateien, die sich an dieser Speicheradresse befinden. Die Güte der Verteilung (also wie gut die Verteilung für die Datenorganisation ist) kann man als den Durchschnittsaufwand der Suche über allen Dateien messen. Für das obige Säulendiagramm mit 10 Dateien ist dies konkret:

$$\frac{2 \cdot 2 + 1 \cdot 1 + 3 \cdot 3 + 1 \cdot 1 + 1 \cdot 1 + 2 \cdot 2}{10} = \frac{20}{10} = 2$$

Je kleiner der durchschnittliche Aufwand, desto besser ist die Hashfunktion. Dieses Beispiel kann man auch zur Einführung des gewichteten Durchschnitts verwenden. Im Säulendiagramm gibt es

- 3 Dateien, die jeweils alleine einer Adresse zugeordnet sind (an den Adressen 2, 6 und 7),
- 4 Dateien, die jeweils in Gruppen der Grösse 2 platziert sind (Adressen 1 und 9),
- 3 Dateien, die in einer Dreiergruppe platziert sind (Adresse 4).

Dies ergibt den folgenden gewichteten Durchschnitt:

$$\frac{3 \cdot 1 + 4 \cdot 2 + 3 \cdot 3}{10} = \frac{3 + 8 + 9}{10} = 2$$

Also muss man, wenn man eine Datei sucht, im Durchschnitt zwei Dateien anschauen. Das ist sehr effizient.

Als LP kann man im Hinterkopf behalten, dass dies eine sehr gute Vorbereitung ist für das Konzept des Erwartungswerts in der Wahrscheinlichkeitstheorie, die in der heutigen Wissenschaft eine immer wichtigere Stellung einnimmt. Wenn man die Güte der Verteilung genau beurteilen möchte, würde man wahrscheinlich den Durchschnitt der Säulenhöhe (d. h. die Hälfte der Säulenhöhe) statt der vollen Höhe nehmen. Das kommt dadurch, dass man wieder im Durchschnitt die Hälfte der Dateien einer Säule anschauen muss, wenn man eine konkrete Datei sucht.

Neue Konzepte

Hier machen wir darauf aufmerksam, dass es keine gute Idee ist, nur einen Teil der Dateibezeichnung zur Adressenbestimmung zu verwenden. Es besteht immer das Risiko, dass viele Dateien gerade in diesem verwendeten Teil der Bezeichnung viele Ähnlichkeiten aufweisen. Hingegen ist eine Rechnung, in der jedes einzelne Symbol des Dateinamens einen wesentlichen Einfluss auf die ausgerechnete Adresse hat, eine gute Strategie für die **Wahl einer Hashfunktion**.

Hier lohnt es sich, auch folgende Aufgabentypen zu betrachten: Eine Gruppe von S verteilt eine Datensammlung mit Dateinamen (z. B. auf Karten notiert) mit einer gegebenen Hashfunktion. Dann legt sie die Karten verdeckt in Stapeln an die entsprechenden Adressen. Die LP gibt einer anderen Gruppe die Hashfunktion und einige Dateinamen, welche die Gruppe suchen soll. Die S dieser zweiten Gruppe rechnen die Adresse aus und suchen im entsprechenden Stapel Karte für Karte nach der gesuchten Datei.

7 Für die Lösung dieser Aufgabe steht die **Kopiervorlage 9** zur Verfügung. Auf den Karten stehen die Ordnungen der Buchstaben im Alphabet, um schnell aus den Namen den Hashwert auszurechnen. Die Aufgabe zeigt eine Möglichkeit zur Berechnung der Speicheradresse aus dem Dateinamen. Hier kann tatsächlich die Änderung jedes einzelnen Buchstabens einen wesentlichen Einfluss auf das Endresultat haben. Für die LP ist es wichtig zu wissen, dass das Teilen durch eine Primzahl keine untergeordnete Rolle spielt. Die S sollen wieder die Verteilung der Daten der Klassenmitglieder visualisieren und mit anderen Verteilungen (z. B. nach dem Geburtsdatum) vergleichen.

Weiterführung

Die stärkeren S können die vorgeschlagene Güte der Verteilung als Durchschnittsaufwand bei der Suche nach einer Datei ausrechnen und somit einen genaueren Vergleich machen.

8 Wir schlagen eine alternative Hashfunktion zu derjenigen in Aufgabe 7 vor. Die S sollen erkennen, dass diese Funktion eine Schwäche hat. Wenn nur ein einziger Buchstabe eine gerade Ordnung im Alphabet hat, wird das Resultat der Multiplikation eine gerade Zahl. Somit erhält man bei fast allen Namen eine gerade Zahl. Der Rest nach der Teilung durch 23 wird dann ungerade sein und somit werden die Zahlen mit geraden Adressen kaum verwendet.

Zusammenfassung und Reflexion

Wir haben in diesem Kapitel drei fundamentale Suchprobleme vorgestellt. Die **binäre Suche** hat unzählige Anwendungen und ist unschlagbar für linear geordnete Datensammlungen, welche stabil sind und nicht oft modifiziert werden. In unserer Welt von sich ständig ändernden Datensammlungen ist **Hashing** das fundamentalste und erfolgreichste Konzept. Hier muss man bei der Datenorganisation aus dem Dateinamen die Speicheradresse bestimmen.

Bei der **Suche im Internet** hat man nur beschränkt die Möglichkeit, die Daten zu organisieren. In allen Computern der Welt werden permanent neue Webseiten angelegt, alte Webseiten gelöscht oder aktualisiert und die Auswahl ist entsprechend riesig. Suchmaschinen sind Produkte der Arbeit unzähliger hochqualifizierter Informatikerinnen und Informatiker. Die Konzepte dahinter sind zu komplex, um an dieser Stelle detaillierter erklärt zu werden. Deswegen haben wir nur die grundlegenden Zielsetzungen und Strategien angedeutet.

Basis und Weiterführung

Sämtliche in diesem Kapitel vorgestellten Themen sollen allen S zugänglich sein. Alle S sollten fähig sein, in konkreten Situationen die binäre Suche durchzuführen. Die meisten können es auch schaffen, mit der binären Suche zu sortieren. Auch ohne Kenntnisse von logarithmischen Funktionen sollen alle S begreifen, dass der Aufwand bei der binären Suche im Vergleich zur Länge der Folge extrem klein ist. Ist auch nur eine zusätzliche Anfrage erlaubt, kann die Länge der Folge bereits verdoppelt werden. Die starken S können den Aufwand der Sortierung mithilfe der binären Suche an konkreten Beispielen messen. Als Hintergrundwissen für die LP führen wir hier die Aufwandanalyse auf: Man muss n Elemente richtig einfügen und das Einfügen kostet jeweils höchstens $\log_2 n$ Speicherbesuche. Somit ist die Sortierung mit $n \cdot \log_2 n$ Aufwand möglich. Weil die S den Logarithmus nicht kennen, sollen sie mit konkreten Zahlen arbeiten und zum Beispiel für $n = 15$ den Aufwand bestimmen. Danach können die S den Aufwand mit dem quadratischen Aufwand im Schulbuch «Programmieren» vergleichen, wo sie mithilfe der Bestimmung des Minimums arbeiten. Diese konkreten Begegnungen mit dem logarithmischen Verhalten sind die beste Vorbereitung auf die allenfalls spätere abstrakte Begegnung mit logarithmischen Funktionen.

Alle S sollen für gegebene Dateien und eine gegebene Hashfunktion die Dateien auf die Speicheradressen verteilen und die Verteilung visualisieren können. Alle S sollen auch erkennen, in welchen Fällen die Verteilung ungünstig ist. Die stärkeren S können den Erwartungswert des Aufwandes als den Durchschnittswert des Aufwandes über alle Dateien berechnen und anhand dieser Werte die Qualität der Verteilungen vergleichen.

Teste dich selbst

Konzepte

1 Die meiste Zeit verbringen Rechner mit der Suche. Gut organisierte Abspeicherung von Daten ist dabei entscheidend für die Effizienz der Suche.

2 Die Funktionsweise der binären Suche ist im Text ausführlich beschrieben. Mit der Schätzung $2^{10} = 1024 \geq 1000$ und somit $2^{20} = (2^{10})^2 \geq 1000^2 = 1\,000\,000$ kann man sich dem Resultat annähern. Ohne den Logarithmus teilt man die Zahl 1 Million 20 Mal durch 2, bis man als Resultat 1 erhält.

3 Die Idee von Hashing ist, eine Hashfunktion zu wählen, mit der man aus dem Dateinamen die Speicheradresse zum Ablegen der Datei ausrechnen kann. Im vorgeschlagenen Fall gibt es beliebig viele Möglichkeiten. Man muss nur darauf achten, dass man die Adresse nicht nur aus den letzten acht Ziffern berechnet, weil dann alle Dateien mit gleichem Erstelldatum in der gleichen Speichereinheit landen. Man könnte beispielsweise die Buchstaben im Dateinamen durch zweiziffrige Zahlen (die Position des Buchstabens im Alphabet) ersetzen. Dann kann die Ziffer des Datums ausmultipliziert werden und die Zahlen aus der Buchstabenkodierung können addiert werden. Das Resultat kann man durch eine Primzahl teilen, die ungefähr so gross ist wie die Anzahl der Dateien.

4 Man kann die Qualität der Verteilung gut mithilfe eines Säulendiagramms schätzen. Zu hohe Säulen sind nicht gut und zu viele unbenutzte Speicheradressen ebenfalls nicht. Man könnte auch den Durchschnittsaufwand für die Suche nach einer Datei in Betracht ziehen.

5 Die allerbeste Verteilung platziert auf jeder Adresse genau eine Datei.

6 Es kann sein, dass gewisse Jahrgänge häufiger vertreten sind als andere. In diesem Fall landen zu viele Dateien auf derselben Speicheradresse.

7 Man muss die gleiche Sprache (Schrift) zur Erzeugung der Webseiten verwenden. Man hat sich auf die Sprache HTML geeinigt, welche die Strukturierung der digitalen Dokumente festlegt.

8 Der Index einer Webseite legt fest, für welche Themen diese Seite relevant ist. Dies muss im Voraus erledigt werden. Dann kann eine Suchmaschine bei der Suche nach einem Begriff schauen, wie relevant welche Webseiten für das Thema sind, und man erhält sehr schnell entsprechende Vorschläge. Wenn man diese Vorarbeit nicht leisten würde, würde man es nicht schaffen, alle Webseiten in einer vernünftigen Zeit nach dem gesuchten Begriff zu durchsuchen.

9 Man kann prüfen, wie häufig der gesuchte Begriff in den Texten vorkommt sowie ob der Begriff auch im Titel oder im Untertitel vorkommt. Man registriert auch die Anzahl der Besuche dieser Seite, insbesondere von am Thema interessierten Leserinnen und Lesern.

10 Man muss höchstens n + 1 Elemente anschauen, um das gesuchte Element zu finden.

Aufgaben

1 Dies ist eine einfache Aufgabe zur binären Suche.

2 In dieser Aufgabe geht es um den Vergleich zwischen der binären Suche und Hashing, angewendet auf die gleiche Sammlung von Daten.

A Hier geben die S an, wie viele Speichereinheiten (Karten) sie mittels binärer Suche anschauen müssen, um den eigenen Namen zu finden.

B Hier speichern die S die Klassendaten mit Hashing und begutachten die Qualität der Verteilung. Zusätzlich könnten die S prüfen, mit wie vielen anderen ihr Name unter der gleichen Speicheradresse abgelegt ist. Diese Zahl vergleichen sie mit der Anzahl Versuche bei der binären Suche. Danach kann die Klasse abstimmen, welcher der beiden Suchalgorithmen für jeden Einzelnen und jede Einzelne bei der Suche nach dem eigenen Namen weniger Aufwand bedeutet.

IMPRESSUM

einfach INFORMATIK 7–9
Daten darstellen, verschlüsseln, komprimieren
Begleitband

Entwickelt in Zusammenarbeit mit dem Ausbildungs- und Beratungszentrum für Informatik (ABZ) der ETH Zürich.

Autor: Prof. Dr. Juraj Hromkovič
Projektleitung und Redaktion: Sara Venzin, Buchlabor Venzin
Projektleitung Verlag: Daniela Ganter
Satz: tiff.any GmbH, Berlin
Umschlag und Gestaltungskonzept: Hansen Typografische Gestaltung, Luzern
Grafische Illustrationen: Brigitte Gubler, Zürich
Korrektorat: Stefan Zach, z.a.ch GmbH

1. Auflage, 2018
© Klett und Balmer AG, Baar 2018

Alle Rechte vorbehalten.
Nachdruck, Vervielfältigung jeder Art oder Verbreitung – auch auszugsweise – nur mit schriftlicher Genehmigung des Verlags.

ISBN 978-3-264-84467-2

Daten darstellen, verschlüsseln, komprimieren:
Schulbuch 978-3-264-84466-5
Digitale Ausgabe für Lehrpersonen 978-3-264-84468-9

Programmieren:
Schulbuch 978-3-264-84463-4
Begleitband 978-3-264-84464-1
Digitale Ausgabe für Lehrpersonen 978-3-264-84465-8

Strategien entwickeln:
Schulbuch 978-3-264-84469-6
Begleitband 978-3-264-84470-2
Digitale Ausgabe für Lehrpersonen 978-3-264-84471-9

www.einfachinformatik.ch
www.klett.ch
info@klett.ch

Dieser Begleitband bietet Ihnen die notwendigen Werkzeuge und Hintergrundinformationen, um Ihren Informatikunterricht kompetent und nach Lehrplan 21 durchzuführen – unabhängig vom Informatikvorwissen.

Der Begleitband enthält:

- Reiseführer durch den Band «Daten darstellen, verschlüsseln, komprimieren»
- Einfache Schritt-für-Schritt-Anleitungen
- Lösungen zu allen Aufgaben
- Dreijahresplanung zu verschiedenen Stundendotationen
- Hintergrundwissen über die Informatik
- Fachdidaktische Überlegungen und Lernziele zu jedem Kapitel
- Hinweise und Kompetenzraster zu den Kompetenzen im Lehrplan 21
- Empfehlungen zu Grund- und erweiterten Anforderungen
- Kopiervorlagen mit Arbeitsmaterialien

01100101
01101001
01101110
01100110
01100001
01100011
01101000
00100000
01001001
01001110
01000110
01001111
01010010
01001101
01000001
01010100
01001001
01001011

In dieser Reihe ebenfalls erhältlich sind die Bände «Programmieren» und «Strategien entwickeln».
Neu zur Vorbereitung und im Unterricht: Digitale Ausgaben für Lehrpersonen.

FAIR KOPIEREN!
URHEBERRECHT
ACHTEN.
www.fair-kopieren.ch

ISBN 978-3-**264-84467**-2
9 783264 844672