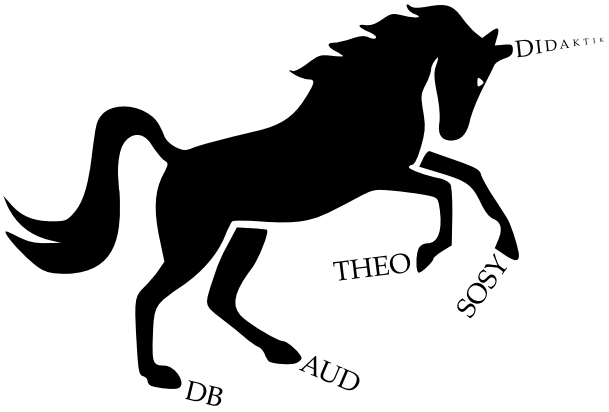
46115
Frühjahr
2016

Theoretische Informatik / Algorithmen / Datenstrukturen (nicht vertieft)

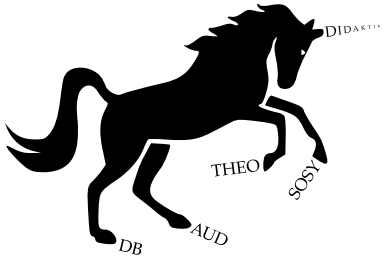
Aufgabenstellungen mit Lösungsvorschlägen



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Aufgabenübersicht



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Thema Nr. 1

Aufgabe 5 [NP]

Beschreiben Sie, was es heißt, dass ein Problem (Sprache) NP-vollständig ist. Geben Sie ein NP-vollständiges Problem Ihrer Wahl an und erläuteren Sie, dass (bzw.) warum es in NP liegt.

Lösungsvorschlag

NP-vollständig: NP-schwer und in NP

[Beliebiges Problem] liegt in NP, da der entsprechende Algorithmus dieses Problem nicht-deterministisch in Polynomialzeit löst → Algorithmus rät nichtdeterministisch Lösung, prüft sie in Polynomialzeit

Aufgabe 8 [Bubble- und Quicksort bei 25,1,12,27,30,9,33,34,18,16]

(a) Sortieren Sie das Array mit den Integer Zahlen

25, 1, 12, 27, 30, 9, 33, 34, 18, 16

(i) mit *BubbleSort*

Lösungsvorschlag

```
 25  1   12  27  30  9   33  34  18  16  Eingabe
 25  1   12  27  30  9   33  34  18  16  Durchlauf Nr. 1
>25  1<  12  27  30  9   33  34  18  16  vertausche (i 0<>1)
 1  >25  12< 27  30  9   33  34  18  16  vertausche (i 1<>2)
 1   12  25  27 >30  9<  33  34  18  16  vertausche (i 4<>5)
 1   12  25  27  9   30  33 >34  18< 16  vertausche (i 7<>8)
 1   12  25  27  9   30  33  18 >34  16< vertausche (i 8<>9)
 1   12  25  27  9   30  33  18  16  34  Durchlauf Nr. 2
 1   12  25 >27  9<  30  33  18  16  34  vertausche (i 3<>4)
 1   12  25  9   27  30 >33  18< 16  34  vertausche (i 6<>7)
 1   12  25  9   27  30  18 >33  16< 34  vertausche (i 7<>8)
 1   12  25  9   27  30  18  16  33  34  Durchlauf Nr. 3
 1   12 >25  9<  27  30  18  16  33  34  vertausche (i 2<>3)
 1   12  9   25  27 >30  18< 16  33  34  vertausche (i 5<>6)
 1   12  9   25  27  18 >30  16< 33  34  vertausche (i 6<>7)
 1   12  9   25  27  18  16  30  33  34  Durchlauf Nr. 4
 1  >12  9<  25  27  18  16  30  33  34  vertausche (i 1<>2)
 1   9   12  25 >27  18< 16  30  33  34  vertausche (i 4<>5)
 1   9   12  25  18 >27  16< 30  33  34  vertausche (i 5<>6)
 1   9   12  25  18  16  27  30  33  34  Durchlauf Nr. 5
 1   9   12 >25  18< 16  27  30  33  34  vertausche (i 3<>4)
 1   9   12  18 >25  16< 27  30  33  34  vertausche (i 4<>5)
 1   9   12  18  16  25  27  30  33  34  Durchlauf Nr. 6
 1   9   12 >18  16< 25  27  30  33  34  vertausche (i 3<>4)
 1   9   12  16  18  25  27  30  33  34  Durchlauf Nr. 7
 1   9   12  16  18  25  27  30  33  34  Ausgabe
```

(ii) mit *Quicksort*, wenn als Pivotelement das jeweils erste Element gewählt wird.

Beschreiben Sie die Abläufe der Sortierverfahren

(i) bei *BubbleSort* durch eine Angabe der Zwischenergebnisse nach jedem Durchlauf

(ii) bei *Quicksort* durch die Angabe der Zwischenergebnisse nach den rekursiven Aufrufen.

(b) Welche Laufzeit (asymptotisch, in O-Notation) hat BubbleSort bei beliebig großen Arrays mit n Elementen. Begründen Sie Ihre Antwort.

Thema Nr. 2

Aufgabe 2 [SUBSET SUM, Raumausstattungsunternehmen]

Ein Raumausstattungsunternehmen steht immer wieder vor dem Problem, feststellen zu müssen, ob ein gegebener rechteckiger Fußboden mit rechteckigen Teppichresten ohne Verschnitt ausgelegt werden kann. Alle Längen sind hier ganzzahlige Meterbeträge. Haben sie beispielsweise zwei Reste der Größen 3×5 und einen Rest der Größe 2×5, so kann ein Fußboden der Größe 8×5 ausgelegt werden.

Das Unternehmen beauftragt eine Softwarefirma mit der Entwicklung eines Programms, welches diese Frage für beliebige Größen von Fußboden und Teppichresten entscheiden soll. Bei der Abnahme weist die Softwarefirma darauf hin, dass das Programm im wesentlichen alle Möglichkeiten durchprobiert und daher für große Eingaben schnell ineffizient wird. Auf die Frage, ob man das nicht besser machen könne, lautet die Antwort, dass das vorgelegte Problem NP-vollständig sei und daher nach derzeitigem Kenntnisstand der theoretischen Informatik nicht mehr zu erwarten sei.

> **Exkurs:**
> Das **Teilsummenproblem** (SUBSET SUM oder SSP) ist ein spezielles Rucksackproblem. Gegeben sei eine Menge von ganzen Zahlen $I = \{w_1, w_2, \ldots, w_n\}$. Gesucht ist eine Untermenge, deren Elementsumme maximal, aber nicht größer als eine gegebene obere Schranke c ist.

(a) Fixieren Sie ein geeignetes Format für Instanzen des Problems und geben Sie konkret an, wie die obige Beispielinstanz in diesem Format aussieht.

Lösungsvorschlag

Problem L

1. Alternative $I = \{x_1, y_1, \ldots, x_n, y_n, c_x, c_y\}$

2. Alternative $I = \{w_1, \ldots, w_n, c\}$

(b) Begründen Sie, dass das Problem in NP liegt.

Lösungsvorschlag

Es existiert ein nichtdeterministischer Algorithmus der das Problem in Polynomialzeit entscheidet:

- nichtdeterministisch Untermenge raten $(\mathcal{O}(n))$
- Prüfe: $(\mathcal{O}(n))$

1. Alternative Elementsumme der Produkte (x_i, y_i) aus Untermenge $= c$

2. Alternative Elementsumme der Untermenge $= c$

(c) Begründen Sie, dass das Problem NP-schwer ist durch Reduktion vom NP-vollständigen Problem SUBSET-SUM.

Lösungsvorschlag

$\preceq_p$ L

1. Alternative Die Funktion f ersetzt jedes w_i durch w_i , 1 und c durch c, 1 und startet TM für L

Berechenbarkeit: Hinzufügen von 1 für jedes Element, offensichtlich in Polynomialzeit

Korrektheit: $w \in \Leftrightarrow f(w) \in L$, offensichtlich, selbes Problem mit lediglich anders notierter Eingabe

2. Alternative Die Funktion f startet TM für L

Berechenbarkeit: Identität, offensichtlich in Polynomialzeit

Korrektheit: $w \in \Leftrightarrow f(w) \in L$ offensichtlich, selbes Problem

Aufgabe 3 [Bruchsicherheit von Smartphones]

Sie sollen mithilfe von Falltests eine neue Serie von Smartphones auf Bruchsicherheit testen.

Dazu wird eine Leiter mit n Sprossen verwendet; die höchste Sprosse, von der ein Smartphone heruntergeworfen werden kann ohne zu zerbrechen, heiße *„höchste sichere Sprosse"*. Das Ziel ist, die höchste sichere Sprosse zu ermitteln. Man kann davon ausgehen, dass die höchste sichere Sprosse nicht von der Art des Wurfs abhängt und dass alle verwendeten Smartphones sich gleich verhalten. Eine Möglichkeit, die höchste sichere Sprosse zu ermitteln, besteht darin, ein Gerät erst von Sprosse 1, dann von Sprosse 2, etc. abzuwerfen, bis es schließlich beim Wurf von Sprosse k beschädigt wird (oder Sie oben angelangt sind). Sprosse $k-1$ (bzw. n) ist dann die höchste sichere Sprosse. Bei diesem Verfahren wird maximal ein Smartphone zerstört, aber der Zeitaufwand ist ungünstig.

(a) Bestimmen Sie die Zahl der Würfe bei diesem Verfahren im schlechtesten Fall.

Lösungsvorschlag

Die Zahl der Würfe im schlechtesten Fall ist $\mathcal{O}(k)$, wobei k die Anzahl der Sprossen ist. Geht das Smartphone erst bei der höchsten Sprosse kaputt, muss es k mal heruntergeworfen werden. Die Komplexität entspricht der der linearen Suche.

(b) Geben Sie nun ein Verfahren zur Ermittlung der höchsten sicheren Sprosse an, welches nur $\mathcal{O}(\log n)$ Würfe benötigt, dafür aber möglicherweise mehr Smartphones verbraucht.

Lösungsvorschlag

Man startet bei Sprosse $\frac{n}{2}$. Wenn das Smartphone kaputt geht, macht man weiter mit der Sprosse in der Mitte der unteren Hälfte, ansonsten mit der Sprosse in der Mitte der oberen Hälfte. Das Ganze rekursiv.

(c) Es gibt eine Strategie zur Ermittlung der höchsten sicheren Sprosse mit $\mathcal{O}\left(\sqrt{n}\right)$ Würfen, bei dessen Anwendung höchstens zwei Smartphones kaputtgehen. Finden Sie diese Strategie und begründen Sie Ihre Funktionsweise und Wurfzahl.

Tipp: der erste Testwurf erfolgt von Sprosse $\lceil \sqrt{n} \rceil$.

Exkurs: Interpolationssuche

Die Interpolationssuche, auch Intervallsuche genannt, ist ein von der binären Suche abgeleitetes Suchverfahren, das auf Listen und Feldern zum Einsatz kommt.

Während der Algorithmus der binären Suche stets das mittlere Element des Suchraums überprüft, versucht der Algorithmus der Interpolationssuche im Suchraum einen günstigeren Teilungspunkt als die Mitte zu erraten. Die Arbeitsweise ist mit der eines Menschen vergleichbar, der ein Wort in einem Wörterbuch sucht: Die Suche nach Zylinder wird üblicherweise am Ende des Wörterbuches begonnen, während die Suche nach Aal im vorderen Bereich begonnen werden dürfte. [a]

[a] https://de.wikipedia.org/wiki/Quadratische_Binärsuche

Exkurs: Quadratische Binärsuche

Quadratische Binärsuche ist ein Suchalgorithmus ähnlich der Binärsuche oder Interpolationssuche. Es versucht durch Reduzierung des Intervalls in jedem Rekursionsschritt die Nachteile der Interpolationssuche zu vermeiden.

Nach dem Muster der Interpolationssuche wird zunächst in jedem rekursiven Schritt die vermutete Position k interpoliert. Anschließend wird – um die Nachteile der Interpolationssuche zu vermeiden – das Intervall der Länge $\sqrt{n}$ gesucht, in dem sich der gesuchte Wert befindet. Auf dieses Intervall wird der nächste rekursive Aufruf der Suche angewendet.

Auf diese Weise verkleinert sich der Suchraum bei gegebener Liste der Länge n bei jedem rekursiven Schritt auf eine Liste der Länge $n\ \sqrt{n}$. [a]

[a] https://de.wikipedia.org/wiki/Quadratische_Binärsuche

Lösungsvorschlag

Das Vorgehen ist folgendermaßen: Man beginnt auf Stufe 0 und falls das Handy nicht kaputt geht, addiert man jeweils Wurzel n. Falls das Handy kaputt geht, geht man linear in Einerschritten das Intervall von der unteren Grenze (ðvon der Stufe vor der letzten Addition) bis zur Kaputtstufe ab. [a]

[a] http://www.inf.fu-berlin.de/lehre/WS06/HA/skript/vorlesung6.pdf