Kapitel 2 (Sortieralgorithmen)

·) Problemgröße: Maß für Aufwand(=)

↳ in diesem Kapitel: meist Anzahl d. zu sortierenden Objekte

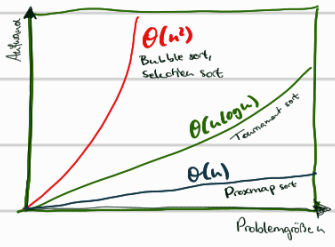
·) "divide et impera" → Problem in Teilprobleme aufteilen

·) Mensch vs. Maschine: Maschine kann nichts "i. d. Hand" haben

↳ mehrere Schritte bzw. ein Zwischenspeicher ist erforderlich

SORTIERALGORITHMEN

"Klassische" Algorithmen: Selection sort, Bubble sort, Tournament sort



↳ Selection sort → $\Theta(n^2)$

↳ Einfach durch Liste iterieren bis man das kleinstmöglichste Element findet

↳ Bubble sort → $O(n^2)$

↳ Elemente werden in Zweierpaaren verglichen bis alle an der richtigen Stelle sind

↳ Tournament sort → $\Theta(n \log n)$

↳ Elemente treten gegeneinander an → "Hierarchie" entsteht

↳ eignet sich v.a. bei großem n

Man kann mathematisch zeigen, dass es beim Sortieren nichts besseres als $\Theta(n \log n)$ gibt!

AUßER man verwendet Zusatzinformationen über die Daten:

Proxmap sort → $\Theta(n)$

↳ Position wird anfangs schon abgeschätzt

↳ Ähnlichkeiten zum Hashing: Desto mehr Speicher im Vergleich zu Elementen, desto weniger Kollisionen (→ Belegungsgrad optimieren)

↳ Funktionswert nur gut, wenn Verteilung d. Elemente bekannt ist, sonst ist es ineffizient

Kapitel 3 (Rucksackproblem)

·) Dynamische Programmierung → viele Teilprobleme lösen & als Bausteine verwenden

↳ Problemgröße hier: Kistengröße & Anzahl d. Schätze

DAS RUCKSACKPROBLEM

↳ Es gibt eine Kiste bestimmter Größe & "Schätze" bestimmter Größe/Wertigkeit

↳ Fragestellung: Wie viele von welchen Schätzen packt man in die Kiste, um den größtmöglichen Wert zu erzielen?

Lösungsverfahren nach der dynamischen Programmierung:

Pseudocode:

Fülle alle Kisten mit dem kleinsten Schatz

Für alle Schätze vom zweitkleinsten zum größten:

- Für jede Kiste A von der kleinsten zur größten:
  - Lege den Schatz zur Kiste A
  - Kiste B := Kiste d. Größe A + dem aktuellen Schatz
  - Ist Inhalt von (A + Schatz) wertvoller als der von B?
    - wenn ja: B := Inhalt von A + Schatz

VOLLSTÄNDIGE INDUKTION

↳ Induktionsanfang / -verankerung: Aussage gilt für eine Zahl a

↳ Induktionsschritt: Aussage gilt für n-1, also auch für n

↳ Also: Aussage gilt für alle Zahlen $n \geq a$ mit $n \in \mathbb{N}$

Bsp.: Man zeige, dass gilt: $1+2+3+\ldots+n = \frac{n(n+1)}{2}$

Induktionsanfang: $a=1 \rightarrow 1 = \frac{1+1}{2}$ ✓

Induktionsschritt: $1+2+3+\ldots+(n-1)+n = \frac{n(n+1)}{2} \rightarrow \frac{(n-1)n}{2}+n = \frac{n(n+1)}{2} \rightarrow \frac{(n-1)n+2n}{2} = \frac{n(n+1)}{2}$ ✓

Vollständige Induktion beim Rucksackproblem:

1. Lösung stimmt, wenn nur ein Schatz verfügbar ist
2. Kiste d. Größe n+1 wird optimal gefüllt durch Inhalt v. Kiste d. Größe n + Schatz S (oder dadurch, dass man sie gleich lässt)

Also: Wir beweisen, dass der Algorithmus für immer höhere Problemgrößen gilt

P=NP PROBLEM

·) Rucksackproblem funktioniert nur für $n \in \mathbb{Z}_0^+$ (unrealistisch)

↳ Bei kleinsten Änderungen der Problemstellung ist das Problem nicht mehr in polynomieller Zeit lösbar

↳ "NP-vollständige Probleme" → bspw. allg. Rucksackproblem & travelling salesman problem

↳ Wenn wir eine Lösung für eines finden, gibt es für alle eine

↳ Wir konnten bisher nicht beweisen, dass die Probleme nicht in polynomieller Zeit lösbar sind

Die Frage ist: Wurde einfach kein Algorithmus gefunden oder sind sie wirklich nicht in polynomieller Zeit lösbar?

Kapitel 9 (Netzwerke)

·) Eindeutige Identifikation durch IP-Adressen

↳ vergleichbar mit Tel.-Nr. (Vorwahl & Anschluss-Nr. so wie Netzwerk und Host-Teil)

·) Kommunikation zw. Netzwerken → Router (= Default Gateway)

·) Routing-Tabellen dienen Wegweisung

↳ Nachricht aus welchem Netzwerk muss wohin, um ihr Ziel zu erreichen?

·) DNS ("Domain Name Service")

↳ enthält Mapping von Domains zu IP-Adressen

·) IPv6 → 128bit Länge

OSI-MODELL

7. Application (e.g. HTTPS)

↳ Interpretation d. Inhalts d. Datenpakete

6. Presentation → Darstellung (ASCII, Unicode, ...), Kompaktierung v. Daten

5. Session → Aufbau einer Session / Sitzung

4. Transport (e.g. TCP, UDP)

↳ Datenflusskontrolle, Sicherheit (v.a. bei TCP)

3. Network (e.g. Layer 3 Switch, Router) → Routing

2. Data Link (e.g. Switch)

↳ Prüfsummen, Nummern, etc. (zuverlässige Übertragung)

1. Physical (e.g. Hub) → Bits/elektr. Signale

·) IANA ("Internet Assigned Numbers Authority") → vergibt IP-Adressen (in Blöcken)

·) ICANN ("Internet Corp for Assigned Names and Numbers") → vergibt Domains (max. 255 Zeichen)

·) Internet ist Kombination sehr vieler Router → divide et impera (jedes Netzwerk ist kleiner Teil)

·) Paradigma → Denkmuster o. Vorbild, anhand dessen ein System o. Konzept konstruiert wird

↳ Hier: wir wenden Strukturen der Realität an → "Paradigmenbildung"

Kapitel 11 (Ordnung, Hashing)

·) sequentielle Suche →  durch ganze Liste iterieren

·) binäre Suche →  Anfang i.d. Mitte → größer o. kleiner? → $O(n \log n)$

sortiert!

HASHING

↳ Grundidee: Man muss nicht nach Elementen suchen, sondern bestimmt direkt ihre Position

↳ Nachteil: Speicherlastig

↳ Man muss ein sinnvolles & effizientes Sortierkriterium finden, sodass jedes Element einen von den anderen Elementen unabhängigen Platz (& jeder Platz nur ein Element) hat

↳ Kollisionsbehandlung, wenn Platz belegt ist

↳ Belegungsgrad → % vom Speicher, der genutzt wird

↳ notwendige Zugriffe steigen mit Belegungsgrad → ideal: < 85% (je nach Relevanz d. Schnelligkeit)

↳ Hashfunktion: Schlüssel → $h(s)$ → Speicherposition

↳ wie optimiert man $h(s)$? → menschl. Zahlensysteme meist ungeeignet (zu viele Kollisionen)

↳ Lösung: modulo durch Primzahl: bspw. $h(s) = s \bmod 11$

↳ auch problematisch: gute Kollisionsbehandlung → $[h_i(s) = h_0(s) + i \bmod 103]$ ist z.B. bei Häufungen nicht ideal

Anwendung: DB, Echtzeitsysteme, OS, ...

ZUFALLSZAHLEN

↳ Pseudozufallszahlen: Bestimmung rekursiv

↳ "echte" Zufallszahlen → oft Zeit seit Einschalten des PCs in ms oder µs

↳ Linearer Kongruenzgenerator → bspw. $z(i) = (z(i-1) \cdot 21 + 17) \bmod 40$

Kapitel 12 (Kryptographie)

CÄSAR-VERSCHLÜSSELUNG

↳ Alphabet verschieben → Substitutions-Chiffre (Buchstaben werden ersetzt, aber Position bleibt gleich)

↳ Schwachstellen: Bruteforce, Analyse d. Häufigkeit d. Buchstaben

→ bei symm. Verschlüsselung: Schlüsselaustauschproblem (Wie kommuniziert man sicher den Schlüssel?)

↳ Lösung: Man kommuniziert den symm. Schlüssel mittels asymm. Verfahren (meist RSA)

=> B verschlüsselt die Nachricht mit A's public key

=> A entschlüsselt die Nachricht mit seinem private key

(zur Authentifizierung kann man auch etwas mit seinem private key verschlüsseln. Da nur man selbst seinen private key hat, wissen andere, dass die Nachricht wirklich von einem selbst kommt, wenn man sie mit dem public key entschlüsseln kann)

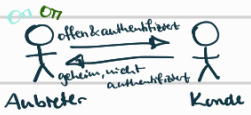
CERTIFICATION AUTHORITY

↳ Man erfährt den public key seines Gesprächspartners durch CA

↳ weil: ·) Der public key ist verifiziert (bspw. durch offizielle Dokumente)

·) Der key kann auf dem Weg nicht ausgetauscht werden, da niemand ihn mit dem private key der CA verschlüsseln kann. Die Kommunikation ist stets verschlüsselt

↳ Der public key der CA muss nicht kommunizert werden, er ist in der Software schon enthalten



Oft haben nur Anbieter Schlüsselpaare

→ Software ist enthalten im OS o. Browser

symm. Verfahren sind schneller, deswegen verwendet man asymm. oft nur zum Schlüsselaustausch

Bsp. für symm. Verfahren: AES, DES, RC4, IDEA (International Data Encryption Alg.)

·) Oft steigt die Anzahl d. mögl. Komb. mit der Länge d. Schlüssels (ist aber je nach Verfahren anders)

↳ Empfehlung: > 128 bit (symm.), > 3000 bit (RSA/asymm.)

RSA (Rivest, Shamir, Adleman)

↳ Grundidee: Seien p, q (große) Primzahlen, so kann man aus $n := p \cdot q$ nur sehr schwer auf p und q schließen

↳ Digitale Signaturen: Verschlüsselung mit private key

↳ beinhalten auch Zeitstempel, damit eine Nachricht nicht bspw. doppelt gesendet wird

Oft wird nur eine Art Prüfsumme verschlüsselt anstatt die ganze Nachricht

·) "Möchten Sie sich trotzdem verbinden?" → Anbieter implementiert eigene CA → unsicher!

↳ Installation eines neuen Zertifikats riskant → zuerst überprüfen bspw. mittels Fingerabdruck d. Zertifikats

·) Wem kann man im Internet vertrauen?

Anbietern ✓ (größtenteils), CA ✓ (staatliche Überwachung)

RSA mit langen Schlüsseln ✓, ABER es gibt keinen Beweis, dass man nicht effizient vom public auf den private key schließen kann

symm. Verfahren mit langen Schlüssel ✓, sich selbst und seinem PC ✗ → Da entstehen die meisten Risiken

ALSO:

↳ sichere Kommunikation mit OM

Kapitel 14 (Fehlererkennung)

FEHLERERKENNUNG & -KORREKTUR

·) DVDs & Blu-Ray: 87% "echte" Daten, 13% Redundanz

Bsp.: A B C D E F G H

000000 000111 011001 011110 101010 101101 110011 110100

↳ **HAMMING-DISTANZ**

↳ Anzahl unterschiedlicher Stellen zw. zwei Codes

Wenn ein Code fehlerhaft ist, vergleicht man ihn mit gültigen Codes. Der mit der größten Übereinstimmung ist der Wahrscheinlichste

Die kleinste Hamming-Distanz zweier Codes ist die gesamte

↳ (so, wie ich es verstehe), können bis zu $\frac{n}{2}-1$ fehlerhafte Bits korrigiert werden. Bei $\frac{n}{2}$ ist es 50/50. Danach wird falsch korrigiert.

Kapitel 15 (Affenpuzzle)

Fragestellung: Sind Computer allmächtig, wenn sie nur richtig programmiert werden?

DAS AFFENPUZZLE

↳ NPC (nicht deterministisch polynomiell berechenbar)

↳ n Puzzleteile → n! Permutationen mit 4 Drehrichtungen, also $n! \cdot 4^n$ Möglichkeiten

↳ min. 4 korrekte Lösungen (wegen Rotation)

↳ wenn es genau 4 Lösungen gibt, muss man durchschnittlich $\frac{1}{8}$ aller Möglichkeiten durchgehen

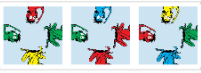
↳ Gesamtaufwand: $\frac{n! \cdot 4^n}{8}$

·) "Branch and Bound" → Ausprobieren & Identifizieren v. "Sackgassen"

↳ Man legt eine Karte nach der anderen und merkt irgendwann, dass keine mehr passt

·) Ab 7x7 Puzzle kann nicht mal Supercomputer lösen (Buch ist Stand 2017)

DOMINO-PROBLEM ("Problem des Fliesenlegers")

·) Fragestellung: Kann man mit diesen Fliesen alle mögl. Flächen füllen? → 

↳ Antwort: ja! (Man kann ein 3x3 bauen, das sich aneinander legen lässt)

↳ Es gibt nur einen gelben & einen blauen Affen, also ist es trivial, wie sie gelegt werden müssen

ABER: Es gibt auch Fliesen, mit denen das nicht geht, z.B.: 

Also: Gibt es ein Verfahren, mit dem man das bestimmen kann?

↳ Antwort: Nein! (bzw. nicht wirklich), oft kommen die Programme zu keinem Schluss und terminieren nicht

↳ Das Problem gilt dann als "nicht entscheidbar"

DAS HALTEPROBLEM

·) Kann ein Programm entscheiden, wann ein anderes terminiert?

↳ Antwort: Nein, Menschen sind nicht i.d. Lage, ein Verfahren zu formulieren