Prüfungsstoff

35% Fachwissen aus dem Buch
im Multiple-Choice-Format

15% Textverständnis anhand Fachbezogenen Texten auf Deutsch und Englisch
Sinnerfassendes Lesen

50% Allgemein-kognitive-Kompetenzen, insbesondere formal-analytisches und logisch-schlussfolgerndes Denken
IQ-Test

Analogien (Analoges Schließen)

Bei diesen Aufgaben werden immer 3 Wörter vorgegeben. Zwischen dem ersten und zweiten Wort gibt es eine Beziehung, die ähnlich auch zwischen dem dritten Wort und einem der fünf Auswahlwörter besteht. Dieses Auswahlwort soll herausgefunden werden.

Tag : Nacht = Licht : ... ?
a) Abend b) Sonne c) Morgen d) Mond **e) Finsternis**

Zahlenfolgen

Die Aufgabe besteht hier immer darin, Zahlenfolgen um ein Glied zu ergänzen. Es ist jene ganzzahlige Zahl zu finden, die die Folge unter Verwendung der Grundrechenarten in richtiger Weise anstelle des Fragezeichens fortsetzt.

2 4 6 8 10 12 14 ... ?
In diesem Falle ist „16" die Zahl, die die Folge in richtiger Weise fortsetzt.
Es sind nämlich immer 2 zu addieren, um zur nächsten Zahl zu gelangen.

Logikaufgaben, Syllogismen und Prämisse

Logikaufgaben sind Aufgaben zum schlussfolgernden Denken. Auf der Grundlage von zwei Aussagen sind gültige Schlussfolgerungen abzuleiten, die sich aus jeder dieser Aussagen oder ihrer logischen Verknüpfung ergeben.

Beispiel
Aussage 1: Zumindest ein Amiv ist Plosa
Aussage 2: Alle Plosa sind Urov

a) Alle Urov sind Plosa
b) Zumindest ein Plosa ist Urov
c) Alle Amiv sind Plosa
d) Zumindest ein Amiv ist Urov
e) Alle Urov sind Amiv

Matrizen

Bei der Lösung von Matrizenaufgaben müssen Sie Figuren, die nach einem bestimmten System angeordnet sind, sinnvoll ergänzen. Die Lösung ist jeweils aus einzelnen Elementen zu konstruieren. Konkrete Fragen zu den Aufgaben unterstützen Sie dabei, schrittweise zur Lösung zu gelangen.

Komplexität

Methode der Abstraktion
Reduktion auf das wesentliche, Wichtigstes herausheben -> Ubahn Karte

Methode der Gleichformung
Alles auf die gleichen Grundelemente zurückzuführen, vereinheitlichen

Algorithmus
Ein Algorithmus ist eine Handlungsvorschrift zur Lösung eines Problems
die sich im Allgemeinen mit einem Computerprogramm umsetzen lässt

Durch die Berechnung der Komplexität (auch Aufwand oder Kosten) eines Algorithmus lassen sie sich unter einander vergleichen.

Aufwandsabschätzung

Das genutzte Maß für den Berechnungsaufwand:

- Anzahl der benötigten Rechenschritte / Zeit — Zeitkomplexität
- Bedarf an Speicherplatz der Speicherbedarf — Platzkomplexität

Zeitkomplexität

- Beziehung zwischen Problemgröße und Zeitaufwand.
- Man lässt Konstanten weg- Können mit schnellerem Computer ausgeglichen werden.

Elementaroperationen
Die Arbeitsschritte die am meisten Zeit brauchen (z.B. Speichervorgänge)

Problemgröße „n"
Maß, wie schwierig bzw. umfangreich die Aufgabe bzw. das Problem ist.
Der zu leistende Aufwand hängt von ihr ab.

Zeit-Komplexitäts-Klassen

Die O-Notation, „Ordnung von O(n)"

Klasse P

Lässt sich in polynomieller Laufzeit lösen und durch ein Polynom ausdrücken.

Proxmap-Sort, Sequentielle / Lineare Suche, Hashing - $O(n)$
Tournament-Sort, Binäre Suche - $O(n \log_2 n)$
Dijkstra-Algorithmus, Selection-Sort, Bubble-Sort - $O(n^2)$

Klasse NP

Die Klasse der „Nicht praktisch berechenbaren" / „nicht polynomiellen" Probleme.
Man kann eine Antwort raten und dann in polynomieller Zeit überprüfen, ob man richtig lag.

Brute-Force - $O(n^n)$

Klasse NP-Vollständig

Selbst wenn man eine Lösung rät, kann man die Richtigkeit nicht in polynomieller Zeit überprüfen.

Traveling-Salesman-Problem (durch alle Städte und zurück)
Rucksackproblem

Das P-NP-Problem

Die Frage, in welcher Beziehung die beiden Komplexitätsklassen P und NP zueinanderstehen.
$P \stackrel{?}{=} NP$, sind alle Probleme der Klasse NP auch in polynomieller Zeit lösbar?

Sortierverfahren

Das „divide et impera" Prinzip
Das Gesamtproblem geteilt in Teilprobleme – Wird dadurch beherrschbar - *„Teile und Herrsche"*

Speicherung im Computer

- Jede Speicheradresse nur ein Wert -> neuer Wert eingegeben - alter Wert überschrieben
- Ein Computer kann immer nur gleichzeitig eine einzige Adresse anschauen
- Zwischenspeicher (Speicheradressen, die besonders schnell und komfortabel abgefragt werden können)

Bubble-Sort

Jede Runde verschiebt sich eines der größten Werte an die letzte Stelle wie ein Bläschen, das hochsteigt

Algorithmus

1. Gehe die Reihe mit Karten von Anfang bis zum Ende durch
2. Vergleiche dabei immer zwei nebeneinanderliegende Karten
    i. Sind sie richtig sortiert **dann** mache gar nichts
    ii. Sind sie falsch sortiert **dann** vertausche sie

Gesamtaufwand = Anzahl der Karten [n] * Aufwand pro Karte [1,3 * n] = $1{,}3 * n^2$

Selection-Sort

Der gesamte Speicher wird durchlaufen, es wird der niedrigste Wert gesucht und mit dem ersten Wert ausgetauscht

Anschließend wird der Vorgang ab der nebenstehenden Speicheradresse wiederholt bis die letzte Karte sortiert wurde. („Anfang" wird nach rechts verschoben)

Algorithmus

1. Suche die [Höchste] Zahl
2. Tausche die [Höchste] Zahl mit der Zahl an der [Erste]n Stelle aus
3. Markiere den Nachbarn der [Höchste]n als die [Erste] Zahl und fange wieder von 1 an bis es keinen weiteren Nachbarn gibt.

Gesamtaufwand = Anzahl der Karten [n] * Aufwand pro Karte [0,6 * n] = $0{,}6 * n^2$

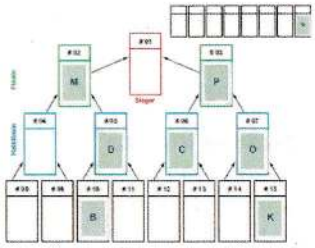
Proxmap-Sort

Braucht zusätzliche Daten: **Verteilung der Karten** – durch statistische Analyse.
Optimal: Gleichverteilung der Karten über Speicherplätze.

Kollisionsbehandlung durch dynamische Anpassung der Speicherplätze
Es entstehen am Ende Lücken zwischen den Werten, die durch Zusammenrücken letztlich beseitigt werden.
Jeder Wert kommt mit jeweils zwei Operationen aus: Einsortieren und Zusammenrutschen O (2n) → O (n)

Tournament-Sort

1. ein Turnierraster wird mit Werten aufgefüllt
2. es treten jeweils die benachbarten Werte gegeneinander an und der höhere Wert steigt in die nächste K.-o.-Runde auf
3. sobald leere Felder in den K.-o.-Runden entstehen, rutscht die nächste Karte aus der unteren Runde nach
4. wenn der Sieger (der höchste Wert) ermittelt wurde, wird dieser am Ende der sortierten Liste abgelegt
5. die unteren Karten rutschen nach und es werden weiter die Sieger bestimmt und in die Liste eingegliedert, bis alle Karten das Raster durchlaufen haben und die Liste sortiert ist



Anzahl der Speicherplätze

- In der unsortierten Reihe

  n Karten

- In der Siegerreihe

  Die ursprünglichen n Karten aber sortiert

- In der K.O.-Tabelle

  In der untersten Ebene sind n Felder und in allen darüberstehenden Ebenen immer halb so viele Felder - dadurch ergibt sich die geometrische Reihe $\frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \frac{n}{16} + \frac{n}{32} + \cdots + 1 \approx n$

Teil 1 - Die Belegung der K.O.-Tabelle

Aufwand = n [Durchläufe / Karten] * h [Verschiebeoperationen / Höhe]

Experimentell lässt sich nachweisen, dass:

2^h Karten → h+1 Ebenen (Wir ignorieren zur Vereinfachung das +1)

2^h Karten → h Ebenen

Es müssen n Felder in der K.O.-Tabelle sein:

„2 hoch was ist n?“

$n \approx 2^h \rightarrow h \approx log_2(n)$

Aufwand = $n * log_2(n)$

Teil 2 - Nachrücken

Füllung des Feldes durch Verschiebung in die nächsthöhere Ebene. Das passiert höchstens so oft es Ebenen in der Tabelle gibt - $\log_2(n)$

Aufwand = $n * log_2(n)$

Gesamtaufwand = $2 * n * \log_2(n)$ wobei Konstanten ignoriert werden

Was bedeutet das *log*?

- Je mehr Karten desto langsamer wächst die Ebenenanzahl
- Je mehr Karten desto langsamer wächst der Rechenaufwand

Aufwand pro Karte steigt logarithmisch mit der Gesamtanzahl der Karten → bei kleiner Anzahl schlechter, bei großer sehr gut

O(n log n) ist unter klassischen Sortierverfahren der günstigste Fall.

Das schnelle Abrufen von Daten

Sequentielle / Lineare Suche

- weil der Reihe nach Karte für Karte gesucht wird
- Speicherinhalt chaotisch oder sortiert

Im Durchschnitt muss die Hälfte der Karten gesehen werden = O(n)

Binäre Suche

- weil man den Suchbereich immer in 2 Hälften teilt
- Speicherinhalt sortiert

Wie Ebenen aus Tournament-Sort = $O(\log_2 n)$

Hash-Funktion

Hash, Durcheinander: Die Ordnung erscheint für den Menschen chaotisch.
Mit einer einzigen Operation kann ein Wert gespeichert und abgerufen werden.

Schlüssel	Der Wert oder die Zeichenfolge nach der Sortiert wird. Der Schlüssel dient als Input der Funktion zur Berechnung der Position.
Hashfunktion	h(s) = Hashwert bzw. Position in der Hashtabelle

Eine Hashfunktion sollte schnell berechenbar sein und die Hashwerte möglichst gleichverteilen.
Ähnliche Eingabewerte sollen zu völlig verschiedenen Hashwerten führen. – Wie ein Zufallsgenerator

Beispiel
Modulo bzw. Mod 13
27 mod 13 = 1

Man sollte durch eine Primzahl teilen:
Schlüssel sind oft vom Dezimalsystem geprägt und sich sehr ähnlich: Bestimmte Ziffern stehen für bestimmte Funktionen (Jahreszahl, Mitgliedsnummer, ...) → Kollisionen und Häufungen

Abschneiden der letzten 3 Ziffern
Modulo 1000

Die doppelte Quersumme
945630 -> 9+4+5+6+3+0 = 27 -> 2+7 = 9

Anwendung
Datenbanken, Betriebsysteme, Echtzeitsysteme

- Speicherorte zuzuweisen
- Prüfsummen berechnen

Vor und Nachteile von Hashing

Vorteil - Schnell
Rechner kann mit einer einzigen Operation die Position errechnen

Nachteil – Benötigt viel Speicherplatz
Daten häufen sich sonst zu stark
Bei Belegungsgrad 80% - 95% mit vielen Datensätzen effektiver als binäre Suche mit Sortierung
In der Praxis: max. 85%

Dynamische Anpassung der Speicher-Tabelle nach statistischer Analyse

Werte lassen sich besser verteilen aber Kollisionen unvermeidbar

Kollisionsbehandlung

Die Belegung der nächsten freien Speicherstelle

- Dadurch kann man aber nicht mehr Werte mit der Hashfunktion abrufen
- Es ist besser die Kollisionsbehandlung vom Schlüssel abhängig zu machen als einfach die freie Stelle daneben zu füllen – so verschiebt man nicht nur die Häufungen nach links oder rechts

Beispiel

Die Kollisionsbehandlung der Hashfunktion

$H_0(s) = s \bmod 103$

(H_0 ist Versuch Nummer 0 die Zahl unterzubringen)

$h_i(s) = (h_0(s) + i) \bmod 103$

„für den i-ten Versuch, addiere i zur ursprünglichen Hashfunktion und nutze sie erneut als Schlüssel".

Pseudozufallszahlengeneratoren

Wenn man nicht alle Lösungswege durchprobieren kann verwendet man Zufallszahlen um sich annähern zu können

Anforderung

Alle Zahlen der Reihe sollten einmal vorkommen – ähnlich wie Hashfunktionen die alle Speicherstellen einmal adressieren sollten.

Pseudo-Zufallszahlengeneratoren

Algorithmen die zufällig aussehende Zahlen bilden nennt man Kongruenzgeneratoren.

„Mid-Square-Generator"

Man quadriert eine Zahl und nimmt seine mittleren Ziffern als neue Zahl.

42 * 42 = 1764 → 76 ist die nächste Zufallszahl.
Es ergibt sich die Folge 24 - 57 - usw.
Es kommen nur zwei Zahlen abwechselnd heraus. Immer nur kurze Folgen, ungeeignet.

Rekursive Funktion – Die Ausgabe wird erneut zur Eingabe

$z(i) = z(i-1) \bmod 40$

$z(1) = z(1-1) \bmod 40$
$z(2) = z(2-1) \bmod 40$
...
$z(n) = 1 \bmod 40$

$z(0) = 1$

Endlosschleife

Die resultierende Zufallszahlenfolge mit „1" als Zufallszahl 0 wäre
~~1, 6, 0, 5, 10, 4, 9, 3, 8 dann wiederholt sich die Reihe~~

Wir benötigen noch etwas, das diesen Startwert verändert:

$z(i) = (z(i-1) * 21 + 17) \bmod 40$

Mit 1 als Startwert erhalten wir
1 - 38 - 15 - 12 - 29 - 26 - 3 - 0 - 17 - 14 - 31 - 28 - 5 - 2 - 19 - 16 - 33 - 30 - 7 - 4 - 21 - 18 - 35 - 32 - 9 - 6 - 23 - 20 - 37 - 34 - 11 - 8 - 25 - 22 - 39 - 36 - 13 - 10 - 27 - 24 - 1 - dann wiederholt sich die Reihe

weil <40

Die meisten Generatoren arbeiten mit Zahlen zwischen 0 und 18 Trilliarden und es dauert sehr lange bis sich die Zyklen wiederholen. Wenn man kleinere Zufallszahlen braucht nimmt man die letzten Stellen.

Physikalische Zufallszahlgeneratoren

Computer nehmen die vergangene Zeit bis Programm ausgeführt wurde seit Computerstart

Das Internet

Im Internet werden Datenpakete von Computern verschickt.
Damit nicht alle Rechner gleichzeitig kommunizieren und um das Netz zu entlasten gibt es eine Hierarchie.

IP Internet Protokoll Nummer / Adresse

- Zahl um Computer zu identifizieren
- 4 Nummern zwischen 0 und 255, mit Punkt getrennt

IANA Internet Assigned Numbers Authority
Vergibt IP-Nummern an Behörden, Organisationen und Internet-Provider.

Class-A-Adresse	130.*.*.*
Class-B-Adresse	130.83.*.*
Class-C-Adresse	130.83.123.* Zahl ganz rechts = Hostanteil

Aus der IP-Adresse lässt sich ablesen welchen Untergruppen man angehört.
Wie bei Hausnummern die Postleitzahl und bei Telefonnummern die Vorwahl.

IP Namen

Zusätzlich zur IP-Nummer

- Statt *http://194.175.173.45/* → IP Name *http://www.google.com/* zum abrufen
- Ein Computer kann auch mehrere Namen tragen → wenn es gleichzeitig Web-Server und Mail-Server ist.

Hierarchie bei den IP-Nummern von rechts nach links, bei IP-Namen umgekehrt: .de .com

ICANN Internet Corporation for Assigned Names and Numbers
Vergibt Domains / Namensbereiche

DENIC Deutsches Network Information Center
Domains mit der Endung „.de“

Man zahlt für die Domain „website.de“ kontinuierlich bei der DENIC eine Gebühr und kann Unterbereiche vergeben, wie „unterbereich.website.de“

DNS Domain Name Service

Alle die Domains und Domain-Unterbereiche administrieren betreiben einen **DNS-Server**.
Quasi Telefonbuch für die IP-Nummer des Zielrechners

Beispiel „ich.alpha.com“ ruft die Seite „www.du.beta.de“ ab

hin

1. ich.alpha.com **an** dns.alpha.com:	**Wie lautet die IP-Nummer von www.du.beta.de?**
2. dns.alpha.com **an** dns.com:	Wie lautet die IP-Nummer von www.du.beta.de?
3. dns.com **an** dns.de:	Wie lautet die IP-Nummer von www.du.beta.de?
4. dns.de **an** dns.beta.de:	Wie lautet die IP-Nummer von www.du.beta.de?

retour

5. dns.beta.de **an** dns.de:	Die Nummer lautet 10.49.88.17
6. dns.de **an** dns.com:	Die Nummer lautet 10.49.88.17
7. dns.com **an** dns.alpha.com:	Die Nummer lautet 10.49.88.17
8. dns.alpha.com **an** ich.alpha.com:	**Die Nummer lautet 10.49.88.17**

Paradigmenbildung

Ein Paradigma ist ein Denkmuster oder Vorbild, anhand dessen ein technisches System modelliert wird.
Dadurch wird es für die Anwender leichter, sich im System zurechtzufinden.
Man folgt quasi dem Beispiel der Wirklichkeit: Zeigt sich bei Programmiersprachen und Internet-Netzwerken

Router

Router sind spezialisierte Rechner, die Datenpakete mit **Routing-Tabellen** weiterleiten.
Sie besitzen zwei oder mehrere IP-Adressen, da sie in mehreren Netzen sind.

Gateways können im gegensatz zu den Routern (default gateways) auf allen Ebenen des OSI Modells weiterleiten.

LAN Local Area Network

- Computer eines Unternehmens oder einer Abteilung die eng miteinander verbunden sind.
- Dürfen bestimmte Anzahl an Benutzern nicht überschreiten.
- Es gibt weitere Hierarchiestufen die zB für die Verbindung zwischen Drucker und PC verantwortlich sind.

WAN Wide Area Network

- Die Verbindung von mehreren LANs
- Erstreckt sich über Kontinente.
- Es gibt weitere Hierarchiestufen die zB die großen Internetanbieter miteinander verbinden.

OSI Model - Allgemeines Modell zur Verbindung von Computersystemen

Open Systems Interconnection Reference Model

1-4 = transportorientiert
5-7 = anwendungsorientiert

1. Physical Layer / Physikalische Schicht

Mechanische und elektrische Übertragungsmittel: Leitungen, Stecker, Netzwerkkabeln
Umwandlung der Bits in elektrischen Signalen

2. Data-link Layer/ Datenverbindungs Schicht

Funktionen zur Fehlererkennung, Fehlerbehebung: Prüfsummen, gezielte Redundanz
Mittelgroße Netzwerke - allein mit der zweiten Schicht ohne Router. Ein **Switch** verbindet die einzelnen Computer und leitet Datenpakete weiter.

3. Network Layer / Netzwerk Schicht

Kommunikation der Computer, IP-Adressen, Routing, Routingtabellen

4. Transport Layer / Transport Schicht

Stauvermeidung durch Priorisierung von Datenpaketen
Telefonieren, höhere Priorität als das abrufen einer Website, kleine Verzögerungen verzerren die Sprache

5. Session Layer / Sitzungs Schicht

Zuordnung von Sitzungsnummern und Checkpoints

6. Presentation Layer / Darstellungs Schicht

Darstellung der Daten in eine für den Menschen verständliche Form mit ASCII oder Unicode

7. Application Layer / Anwendungs Schicht

Dienste, Anwendungen und die Dateneingabe und -ausgabe.
Stellt Funktionen für die Anwendungen zur Verfügung und kommuniziert mit den unteren Schichten Anwendungen: Webbrowser, E-Mail-Programm (die Anwendungen selbst gehören nicht zur Schicht) **Anwendungs-Protokolle:** Web HTTP Hypertext Transfer Protocol E-Mail SMTP Small Mail Transfer Protocol Dateien FTP File Transfer Protocol Zeitansage NTP Network Time Protocol Server die verschiedene Protokolle benützen können erschwert kommunizieren

OSI Modell - Open Systems Interconnection Reference Model

Allgemeines Modell zur Verbindung von Computersystemen

1-4 = transportorientierte Schichten
5-7 = anwendungsorientierte Schichten

OSI-Layer 1 (Physical Layer, Physikalische Schicht)

Umwandlung der Bits in ein zum Übertragungsmedium passendes Signal.
mechanische, elektrische und weitere funktionale Hilfsmittel • Die genaue Beschreibung der elektrischen Signale im Netzwerkkabel oder die Farbe des verwendeten Lichtes in Glasfaserkabel. • Hardware: Repeater, Hubs, Leitungen, Stecker, etc.

OSI-Layer 2 (Data-link Layer, Datenverbindungsschicht, ~~Sicherungsschicht~~)

~~Vermeidung von Datenverlust und Übertragungsfehlern (Verfälschung durch Leitungs-Störungen)~~
Funktionen zur Fehlererkennung, Fehlerbehebung und Datenflusskontrolle. • Zusätzliche Daten wie Nummern, Prüfsummen, mit denen man kontrollieren kann, ob eine Nachricht unverändert angekommen ist. • Wenn nicht, wird sie nochmals angefordert. • Hardware: Bridge, Switch

Mittelgroße Netzwerke können allein mit der zweiten Schicht arbeiten und benötigen keinen Router. Ein Switch verbindet die einzelnen Computer und leitet Datenpakete weiter.

OSI-Layer 3 (Network Layer, Netzwerkschicht oder Paketschicht, ~~Vermittlungsschicht~~)

Kommunikation der Computer im Internet: Die Wegsuche (Routing) zwischen Netzwerkknoten
• Bereitstellen der IP-Adressen • Routingtabellen • Fragmentierung von Datenpaketen – Unterteilung der Pakete zum Entlasten, Router sollen oft die Last auf den Datenleitungen gleichmäßig verteilen • Hardware: Router

OSI-Layer 4 (Transport Layer, Transportschicht)

Datenflusskontrolle durch Zuordnung von Prioritäten
• Priorisierung von Datenpaketen, Stauvermeidung

Hohe Priorität:	Telefonieren, kleine Verzögerungen verzerren die Sprache
Niedrige Priorität:	Ob eine WWW-Seite innerhalb einer zehntel oder einer halben Sekunde angezeigt wird, registrieren wir normalerweise gar nicht.

OSI-Layer 5 (Session Layer, Sitzungsschicht)

Zuordnung eines Datenpakets zu einer Sitzung
• Datenpaketes → Sitzung wird durch eine Sitzungs-Nummer • Wiederaufsetzpunkte (Fixpunkte oder Check Points), an denen die Sitzung nach einem Ausfall einer Transportverbindung wieder synchronisiert werden kann

OSI-Layer 6 (Presentation Layer, Darstellungsschicht)

Umwandlung von Zahlen in verschiedenen Codecs und Formaten
• Darstellung der Daten (z.B. ASCII, Unicode) in eine für den Menschen verständliche Form • weitere Aufgaben: Datenkompression, Verschlüsselung

OSI-Layer 7 (Application Layer, Anwendungsschicht)

Dienste, Anwendungen und Netzmanagement sowie die Dateneingabe und -ausgabe.
• Funktionen für die Anwendungen zur Verfügung • stellt die Verbindung zu den unteren Schichten her Anwendungen: Webbrowser, E-Mail-Programm (die Anwendungen selbst gehören nicht zur Schicht)

Anwendungs-Protokolle

Web	HTTP = Hypertext Transfer Protocol
E-Mail	SMTP = Small Mail Transfer Protocol
Dateien	FTP = File Transfer Protocol
Zeitansage	NTP = Network Time Protocol

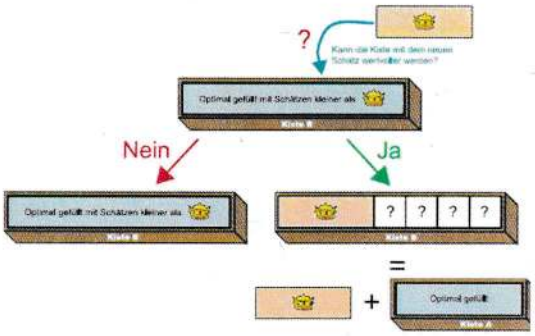
Ein E-Mail-Programm hat zum Beispiel Schwierigkeiten, mit einem WWW-Server zu reden, weil es SMTP spricht und der Server aber nur HTTP versteht.

Das Rucksackproblem

Eine Kiste mit einer bestimmten Größe soll mit Gegenständen unterschiedlicher Größe und Wertes so gefüllt werden, dass die Gegenstände insgesamt den größtmöglichen Wert annehmen. → **Optimierungsaufgabe**

Dynamische Programmierung

Strategie zur zur Lösung von Optimierungsaufgaben.
Kombinierung von Teilproblemen.



Algorithmus

Eine Kiste ist bereits mit kleineren Schätzen optimiert.
Wenn sie mit dem neuen Schatz nicht wertvoller wird bleibt sie wie sie ist.
Wenn sie mit dem neuen Schatz wertvoller wird dann wird der neue Schatz aufgenommen und das übrige Volumen mit einer kleineren optimierten Kiste gefüllt.

Ähnlich mit der vollständigen Induktion der Mathematik

zweiteiliges Beweisverfahren für ganze Zahlen

Beweise

$$1 + 2 + 3 + \cdots + n = \frac{n \cdot (n+1)}{2}$$

Induktions-Verankerung

Das Problem wird für eine kleinere Problemgröße gelöst und anschließend davon auf die nächstgrößere Lösung abgeleitet

$$1 = \frac{1 \cdot (1+1)}{2}$$

Bsp: Verankerung bei Problem des kürzesten Pfades: Weg von Startpunkt = 0km

Wir gehen davon aus dass, folgende Gleichung richtig ist:

$$1 + 2 + 3 + \cdots + (n-1) = \frac{(n-1) \cdot [(n-1)+1]}{2} = \frac{(n-1) \cdot n}{2}$$

$$1 + 2 + 3 + \cdots + (n-1) + n = \frac{n \cdot (n+1)}{2}$$

$$\frac{(n-1) \cdot n}{2} + n = \frac{n \cdot (n+1)}{2}$$

Problem

Beziehung zwischen dynamischer Programmierung und vollständiger Induktion: **Die Lösung funktioniert nur für „ganzzahlige" Problemgrößen.**

Alle Kisten und Schätze bestehen aus einer ganzzahligen Anzahl von Quadraten.
Hier konnte man leicht von einer kleineren Kiste auf eine entsprechend größere schließen.
Reale Probleme nicht!

Darüber hinaus berücksichtigt es die **begrenzte Anzahl der Gegenstände** nicht.

Datenverschlüsselung

Verstecken statt Verschlüsseln

müssen zum Entschlüsseln erst gefunden werden.
Die Sicherung des geistigen Eigentums – Urheberschaftsnachweis – **digitale Wasserzeichen**

Skytalen

Erste **Chiffriermaschine** der Welt von Spartanern.
Papyrus spiralförmig um Holzstäben mit gleichen Durchmessern gewickelt.
Transpositions-Chiffre (genau dieselben Zeichen in veränderter Reihenfolge).

Cäsar-Scheibe

Symmetrische Verschlüsselung von Cäsar, Alphabet um eine beliebige Anzahl an Stellen verschoben.
Erzeugt eine **Substitutions-Chiffre** (Position bleibt gleich aber Zeichen werden ausgetauscht)

Lässt sich durch Analyse der **Buchstabenhäufigkeit** knacken - „E" als häufigster Buchstabe in der deutschen Sprache. Deshalb müssen der Buchstaben mit zB der Huffman-Kodierung gleichverteilt werden.

symmetrische Verschlüsselung

ein und derselbe Schlüssel wird für die Ver- und Entschlüsselung verwendet

- Wenn Empfänger in reichweite ist kann Schlüssel persönlich ausgetauscht werden
- Im Internet müsste der Schlüssel unverschlüsselt verschickt werden

asymmetrische Verschlüsselung

zwei unabhängige Schlüssel werden wechselweise zur Codierung und Decodierung von Nachrichten verwendet.

Experiment: „Asymmetrische Chiffriermaschine"
Eigentlich symmetrisch aber soll eine asymmetrische Verschlüsselung darstellen: Cäsar-Scheibe mit einer unsortierten Liste an Substitutionsbuchstaben

ein Schlüssel (z.B. EMU) wird wiederholend über den zu verschlüsselnden Klartext geschrieben und erzeugt dadurch eine verschlüsselte Nachricht

es gibt immer zwei zusammenhängende Schlüsselpaare wie bei asymmetrischer Verschlüsselung
EMU → FIR

1. A → B geheime Botschaft
2. B hat ein Schlüsselpaar
   Public key „BOT": Jeder, der ihn wissen möchte, kann ihn bekommen
   Private key „OBP": bleibt geheim
3. C ist der Postbote und will A´s geheime Botschaft entschlüsseln

Versuch 1

A → B	„Wie lautet dein Schlüssel?" - unverschlüsselt
C überbringt	
A ← B	„Public key: BOT" - unverschlüsselt
C überbringt	
A → B	„Geheime Botschaft" - verschlüsselt mit public key
C überbringt	
	B entschlüsselt „geheime Botschaft" mit private key Kein anderer könnte die mit der public key verschlüsselte Botschaft ohne private key entschlüsseln.
A ← B	„habe es erhalten" - unverschlüsselt

Sabotage 1

A → B	„Wie lautet dein Schlüssel?" - unverschlüsselt
C überbringt	
A ← B	„Public key: BOT" - unverschlüsselt
C überbringt	
A → B	„Geheime Botschaft" - verschlüsselt mit public key
C behält´s	Kann die Nachricht aber ohne private key nicht entschlüsseln
A ← C	„habe es erhalten" - unverschlüsselt

Problem: Bestätigung des Erhalts nicht authentifiziert, kann vorgetäuscht werden

Versuch 2

A → B	„Wie lautet dein Schlüssel?" - unverschlüsselt
C überbringt	
A ← B	„Public key: BOT" - unverschlüsselt
C überbringt	
A → B	„Geheime Botschaft" - verschlüsselt mit public key
C überbringt	
	B entschlüsselt „geheime Botschaft" mit private key
A ← B	**„habe es erhalten" - verschlüsselt mit private key**

Lösung: Authentifizierung

Unterschrift mit Private-Key -> jeder kann die Nachricht mit öffentlichem Schlüssel lesen, aber nicht manipulieren
Man kann die mit private key verschlüsselte Nachricht decodieren, aber selbst nicht erzeugen.

Sabotage 2

A → B	„Wie lautet dein Schlüssel?“ - unverschlüsselt
C überbringt	
C ← B	„Public key: BOT“ - unverschlüsselt
C behält´s	
A ← C	„Public key: CLU“ – Ist eigentlich Public key von C. B hätte keine authentifizierte Nachricht schicken können weil A erst jetzt seine public key erhalten hätte
A → B	„Geheime Botschaft“ - verschlüsselt mit public key von C
C überbringt	
	B entschlüsselt „geheime Botschaft“ mit private key
A ← B	„habe es erhalten“ - **verschlüsselt mit private key**

Problem: die Schlüsselübergabe ist nicht sicher C bildet eine Art Zwischeninstanz
Um keinen Verdacht zu erwecken, mit Ottos öffentlichem Schlüssel BOT codiert, an Otto weitergesendet

Versuch 3

Lösung: Zertifizierungsinstanz (Certification Authority)

Instanzen, die die sichere Aufbewahrung und Weitergabe von Schlüsseln gewährleisten

DST
Equifax
Deutsche Telekom
Verisign

Bekannte Beispiele: DST, Equifax, Deutsche Telekom, Verisign

- CAs gelten allgemein als sicher, sind staatlich überwacht
- CAs können bei Ausspähung eines Schlüssel diesen sperren

Die öffentlichen Schlüssel der wichtigsten CAs sind bereits in Browsern oder auf Betriebssystemen vorinstalliert. CAs, die ihre Schlüssel noch nicht vorinstalliert haben, haben ihre öffentlichen Schlüssel bei bekannteren CAs zertifizieren lassen.

Bei wichtiger Daten-Übermittlung im Internet prüft der Browser anhand einer CA den korrekten öffentlichen Schlüssel des Anbieters.

- CA übergibt jedem persönlich den eigenen öffentlichen Schlüssel, um die Echtheit zu garantieren (keine Verfälschung möglich)
- B gibt persönlich CA seinen Schlüssel
- CA prüft die Authentizität seiner Identität und archiviert den Schlüssel

A → CA	„Wie lautet B´s Schlüssel?“ - verschlüsselt mit CAs public key
C überbringt	Kann ohne private key nicht entschlüsseln
A ← CA	„Public key: BOT“ – verschlüsselt mit CAs private key (authentifiziert)
C überbringt	
A → B	„Geheime Botschaft“ - verschlüsselt mit public key
C überbringt	
	B entschlüsselt „geheime Botschaft“ mit private key
A ← B	„habe es erhalten“ - verschlüsselt mit private key (authentifiziert)

Sabotage 3

	C trifft CA und hinterlegt dort seinen öffentlichen Schlüssel CLU Ab sofort kann die CA auch seinen Schlüssel auf Anfrage weitergeben
A → CA	„Wie lautet B´s Schlüssel?" - verschlüsselt mit CAs public key
C behält´s	weil C rät, dass A den Schlüssel von B erfragt
C → CA	„Hallo ich bin A, wie lautet C´s Schlüssel?" (kann nicht authentifiziert werden weil A keine private und public key hat)
A ← CA	„Public key: CLU" – verschlüsselt mit CAs private key (authentifiziert)
C überbringt	und bildet eine Zwischeninstanz
A → B	„Geheime Botschaft" - verschlüsselt mit public key von C
C überbringt	Aber davor ließt er die Nachricht und verschlüsselt es mit B´s public key um keinen Verdacht zu erwecken
	B entschlüsselt „geheime Botschaft" mit private key
A ← B	„habe es erhalten" - verschlüsselt mit private key (authentifiziert)

Problem: CA kann nicht feststellen, ob die Anfrage nach einem Schlüssel echt ist oder von einer dritten Partei gefälscht ist, daran kann nichts ändern

Versuch 4

...	
A → CA	„Wie lautet B´s Schlüssel?" - verschlüsselt mit CAs public key
C überbringt	
A ← CA	„Public key **von B:** BOT" – verschlüsselt mit CAs private key die CA sendet nicht nur den Schlüssel, sondern in der gleichen Nachricht auch die Information, zu wem dieser Schlüssel passt
...	

Zertifizierung - Die digitale Unterschrift

Public key mit Namen des Eigentümers, Gültigkeitsdauer + anderen Informationen

Authentifizierung: Mit private key
Zeitstempel: wann die Nachricht gesendet wurde
Fingerabdruck: für jeden Schlüssel eindeutiger Prüfcode

1. Die Bank generiert ein Schlüsselpaar (private and public key)
2. Die Bank bringt public-key persönlich zu einer CA und weis Identität nach
3. Die CA verschlüsselt den public-key der Bank mit der CA-private-key = ZERTFIKAT
4. Die Bank verteilt die mit dem CA-Private-key verschlüsselten

Erweiterung:

- Kommunikation A zu B ist immer geheim, aber **nicht authentifiziert**
- Kommunikation B zu A **ist offen**, aber authentifiziert

A → CA	„Wie lautet B´s Schlüssel?" - verschlüsselt mit CAs public key
C überbringt	Kann ohne private key nicht entschlüsseln
A ← CA	„Public key: BOT" – verschlüsselt mit CAs private key (authentifiziert)
C überbringt	
A → B	„Geheime Botschaft, **Schlüssel für symmetrische Verschlüsselung**" - verschlüsselt mit public key
C überbringt	
	B entschlüsselt „geheime Botschaft" mit private key
A ← B	„habe es erhalten" - verschlüsselt mit private key (authentifiziert), und **symmetrischer Verschlüsselung**

- ab diesem Zeitpunkt ist die Kommunikation sicher
- asymmetrische Verschlüsselung wird nur zu Beginn verwendet, um einen Schlüssel für ein symmetrisches Verfahren sicher auszutauschen, dieser wird pro Sitzung gewechselt

Sichere Verschlüsselungsverfahren müssen die statistischen Besonderheiten der Dateien verschleiern:

- Schemata der Darstellung von Daten auf Webseiten von Anbietern (Banken, OnlineGeschäfte, etc.)
- einige Methoden, die das verhindern: DES (Data Encryption Standard), AES (Advanced Encryption Standard), RC4 (Ron's Cipher 4)
- Die Länge des Schlüssels ist entscheidend für die Sicherheit

asymmetrische
Das ~~symmetrische~~ RSA-Verfahren

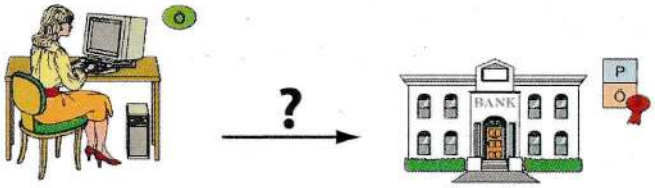
Weil durch alle möglichen Primzahlen dividiert werden muss ist es sehr sicher

Verschleierung
DES — Data Encryption Standard
AES — Advanced Encryption Standard
RC4 — Ron's Cipher 4
Huffman Codierung
Symmetrische Verschlüsselung
RSA Verfahren

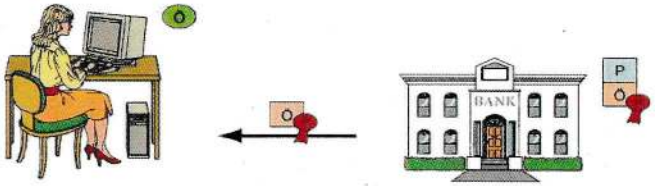
Zusammenfassung

Wie kann ein symmetrischer Schlüssel ausgetauscht werden, damit der Kunde (Jennifer) sicher mit der Bank Überweisungen austauschen kann?

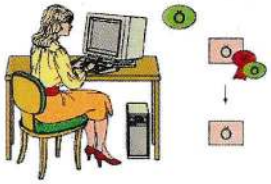
1. Jennifer ist lediglich in Besitz des öffentlichen CA-Schlüssels



2. Jennifer fragt die Bank (oder auch eine CA) nach dem Bank-Zertifikat

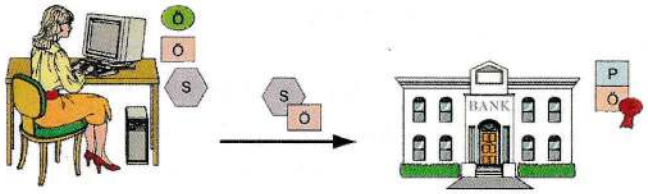


3. Die Bank schickt Jennifer ihr Zertifikat



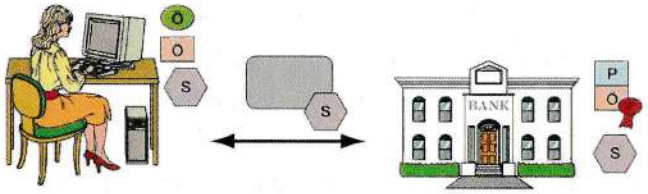
4. Jennifer nutzt den öffentlichen Schlüssel der CA, um das Zertifikat zu entschlüsseln und damit die Echtheit zu überprüfen

5. Jennifer erzeugt zufällig einen Schlüssel für ein symmetrisches Verfahren



6. Jennifer verschlüsselt den symmetrischen Schlüssel mit dem öffentlichen Schlüssel der Bank und verschickt ihn

7. Nur die Bank kann die Nachricht mit ihrem privaten Schlüssel entziffern und damit den symmetrischen Schlüssel akquirieren



8. Da beide Parteien jetzt in Besitz des symmetrischen Schlüssels sind, können sie nun über diesen sicher Nachrichten (zum Beispiel Überweisungen und Kontoauszüge) austauschen

Fehlertoleranz

Experiment

Ein Feld aus X und O - feste Regel: jede Zeile und Spalte hat eine gerade Anzahl an X.

Je mehr Zeilen und Spalten desto leichter kann man entscheiden welche Fehler man ausbessert

Gezielte Redundanz

- Erweiterung unter einer bestimmten Regel
- Informationsgehalt bleibt gleich
- Verbessert Erkennbarkeit der Fehler

notwendig für Datenübertragung und Speicherung vor allem bei Systemen die sehr große Datenmengen auf engem Raum speichern z.B. Blu-Rays

Beispiele

Prüfziffern

Redundanzprüfung zur Erkennung von Übertragungsfehlern

Fehlerarten wie zB das Vertauschen von Ziffern lassen sich am besten korrigieren

Regel bei Kreditkarten: abwechselnd wird jede Ziffer mit 2 und 1 multipliziert → Summe muss ein Vielfaches von 10 sein

Hamming Distanz bei Codierung

Codierung

Darstellung einer Informationseinheit durch eine Folge von Symbolen (z.B. Binärsystem)

011 000 010 111 = DACH

Hier ist jedes einzelne Bit relevant, keine redundanten Daten

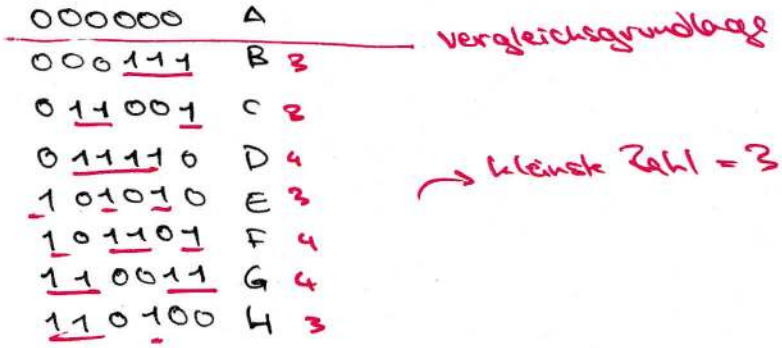
Hamming Distanz

Maß für Unterschiedlichkeit von Codes

011110 000000 011001 110100 = DACH

Doppelt so viele Ziffern

unterscheiden sich alle Buchstaben-Codes an mindestens 3 Stellen heißt das gibt es eine Hamming-Distanz von 3.



Die Grenzen des Computers

Das Affenpuzzle
in der Theorie schon ab 5x5 ohne Computer kaum lösbar.
Es existiert kein Lösungsalgorithmus - Klasse NP

Problem skaliert nicht
manche Probleme beanspruchen bei Verdopplung der Aufgabengröße unermesslich viel mehr Lösungszeit, wie bei Brückenbau

Branch and Bound Technik der Menschen

In Wirklichkeit brauchen Menschen mit dem Branch and Bound Algorithmus weniger Zeit und sind effektiver als der Computer

Branch man verzweigt zu allen potentiellen Lösungen, um die korrekte zu finden
Bound man bricht ab wenn man einen falschen Zweig erwischt hat

Domino-Problem - Problem des Fliesenlegers

Puzzelteile bei denen man unendlich weiterpuzzeln kann, aber nie ein Rechteck findet, das sich periodisch wiederholt

unendliche nichtperiodische Kachelung → ein Programm würde nie fertig werden
Ein Computer kann nicht berechnen, dass dieses Puzzle unendlich ist → **nicht entscheidbares Problem**

Das Halteproblem

Terminierung = Abschließen der Berechnung

nicht terminierende Programme → Endlosschleife

Validierung = Beweis von Fehlerfreiheit eines Teilprogrammes

Es gibt aber teilautomatische Programmbeweiser: Computer muss irgendwann Teilproblem an den Benutzer weitergeben

Kontrollprogramm – HÄLTS?

Entscheidungen: „terminiert es irgendwann" oder „terminiert nicht immer"

Programm

```
1. Falls HÄLTS?(Programm) gehe zu Zeile 1
2. Ende
```

Ist sie wahr verliert sie sich in eine Endlosschleife, ist sie falsch terminiert sie

Angenommen: HÄLTS? (Programm) = JA

→ Programm geht bei Zeile 1 in eine Endlosschleife

Widerspruch: Programm hält nicht an

Angenommen: HÄLTS? (Programm) = NEIN

→ Programm überspringt Zeile 1 und hält bei Zeile 2 an

Widerspruch: Programm hält an

Beide Thesen falsch

Es können keine Programme entwickelt werden können, die bestimmen ob eine Software terminiert oder nicht. Es ist nicht Entscheidbar!

Es können auch keine Programme entwickelt werden können, die eine Software validieren.