Zusammenfassung wichtiger Begriffe

Zur Vorbereitung für das Aufnahmeverfahren für die Bachelorstudien Informatik und Wirtschaftsinformatik

Literatur: J. Gallenbacher, Abenteuer Informatik: IT zum Anfassen - von Routenplaner bis Online-Banking, 3. Auflage. Berlin Heidelberg: Springer-Spektrum, 2012.

Wichtige Begriffe....1
Begriffe im Buch:....1
[Summary] Bubble-Sort (vgl. S. 53f.)....2
[Summary] Selection- Sort (vgl. S. 50f.)....2
[Summary] Tournament- Sort (vgl. S. 62f.)....2
[Summary] Proxmap- Sort (vgl. S. 75f.)....2
[Summary] IP - Adresse (vgl. S. 168f.)....2
[Summary] DNS- Server (Domain Name Service) (vgl. S. 173f.)....3
[Summary] OSI- Modell: Open Systems Interconnection Reference Model (vgl. S. 176f.)....3
[Summary] Sequentielle Suche (vgl. S. 226f.)....3
[Summary] Binäre Suche (vgl. S. 228f.)....3
[Summary] Modulo - Mod (vgl. S. 231f.)....4
[Summary] Hashing (vgl. S. 233f.)....4
[Summary] Kollision und Kollisionsbehandlung beim Hashing (keine Gleichverteilung) (vgl. S. 233f.)....4
[Summary] (Pseudo-)Zufallszahlen (vgl. S. 245)....4
[Summary] Transpositionsschiffre (vgl. S. 254)....4
[Summary] Substitutionsschiffre (vgl. S. 254f.)....5
[Summary] Verschlüsselungsmethoden & ihre Probleme (vgl. S. 259ff.)....5
[Summary] Sicherheitslücken (vgl. S. 283ff.)....5
[Summary] Symmetrische Verschlüsselung (vgl. S. 260ff.)....5
[Summary] Asymmetrische Verschlüsselung (vgl. S. 261ff.)....5
[Summary] RSA- Verfahren (vgl. S. 279ff.)....6
[Summary] Steganographie (vgl. S. 287ff.)....6
[Summary] Fehlerkorrektur (vgl. S. 323ff.)....6
[Summary] Probleme und ihre Lösbarkeit (vgl. S. 339f.)....6
[Summary] Terminierung (vgl. S. 350)....6

Wichtige Begriffe

Im Folgenden findet ihr Kurzzusammenfassungen [Summary] zu wichtigen Konzepten oder Begriffen. Ansonsten ist es empfehlenswert sich die Begriffsdefinitionen in den blauen Kästen im Buch anzusehen. Stand Sommer 2017. Zusammengestellt von einem Studi, also keine Garantie, dass hier keine Fehler drin sind.

Begriffe im Buch:

Übersicht über die Begriffskästen im Buch
[Buch] Methode der Abstraktion (S. 3)
[Buch] Methode der Gleichformung (S. 6)
[Buch] Algorithmus (S. 14)
[Buch] Die Problemgröße (S. 43)

[Buch] Das Prinzip „divide et impera“ (S. 45)
[Buch] Elementaroperationen (S. 56)
[Buch] Die Aufwandsabschätzung (S. 61)
[Buch] Dynamisches Programmieren (S. 85)
[Buch] Schleifen: (S. 98)
[Buch] Vollständige Induktion (S. 101)
[Buch] Codierung (S. 121) *(2017 nicht Prüfungsstoff)*
[Buch] Binärcode (S. 122) *(2017 nicht Prüfungsstoff)*
[Buch] Präfix (S. 129) *(2017 nicht Prüfungsstoff)*
[Buch] Verlustfreie Komprimierung (S. 132) *(2017 nicht Prüfungsstoff)*
[Buch] Lauflängencodierung (S. 136) *(2017 nicht Prüfungsstoff)*
[Buch] Der Huffman’sche Algorithmus (S. 138ff.) *(2017 nicht Prüfungsstoff, aber für Kapitel 9. - Verschlüsslung - relevant)*.
[Buch] Komprimierung (S. 157) *(2017 nicht Prüfungsstoff)*
[Buch] Paradigma (S. 182)
[Buch] Die Invarianz (S. 188)
[Buch] Kryptographie (S. 257)
[Buch] Redundanz (S. 325)
[Buch] NPC - nicht praktisch berechenbares Problem (S. 341)
[Buch] Branch and Bound (S. 343)

[Summary] Bubble-Sort (vgl. S. 53f.)

In 1 Durchgang wird durch den Vergleich von jeweils 2 Karten immer die höhere in Richtung Ende verschoben, bis die höchste Karte dort liegt, dann verschiebt sich der Sortierbereich in Richtung Anfang. Die Ordnung des Aufwandes ist O (n^2).

[Summary] Selection- Sort (vgl. S. 50f.)

In jedem Durchgang wird von allen Karten die jeweils niedrigste ermittelt & an den Anfang gesetzt. Der Sortierbereich verschiebt sich danach in Richtung Ende. Die Ordnung des Aufwandes ist O (n^2).

[Summary] Tournament- Sort (vgl. S. 62f.)

Auf mehreren Ebenen treten nach dem Sortierkriterium immer 2 Karten gegeneinander an, bis auf der höchsten Ebene eine Siegerkarte feststeht. Diese wird einsortiert. Auf leere Felder rücken immer Karten nach & treten gegeneinander an, bis alle auf dem Siegerfeld angekommen und damit einsortiert sind. Die Ordnung des Aufwandes ist O (nlogn). [h Ebenen, n ~ 2^h = > log2 n ~ h

[Summary] Proxmap- Sort (vgl. S. 75f.)

Vor dem Sortierverfahren wird der Kartenbereich (und mögliche Häufungen) abgeschätzt, danach wird abgeschätzt wo sich die Karte jeweils ungefähr befindet. Bei Kollisionen (es liegt bereits eine Karte da) wird mit anderen Sortierverfahren nachsortiert. Die Ordnung des Aufwandes ist O (n).

[Summary] IP - Adresse (vgl. S. 168f.)

IP (Internet-Protokoll) ist eine Identifikationsnummer für Teilnehmer in einem Netzwerk. Sie besteht aus vier nummern zwischen 0 und 255, die meist mit einem Punkt getrennt sind. Ein Kommunikator, oder Router besitzt immer so viele IP-Adressen wie er Netzwerke miteinander

verbindet. Ein Computer muss die eigene IP-Adresse kennen, die Zuordnungsnummer des eigenen Netzwerkes (erste 3 Zahlen) und die IP-Adresse des angeschlossenen Routers. Die Router besitzen Tabellen anhand denen sie bestimmen wohin sie eine Nachricht weiterleiten (allgemeine und spezielle Regeln). Die IP-Adressen werden von der IANA (Internet Assigned Numbers Authority) in meist Blöcken vergeben (Class A-Adressen: 13.*.*.* = ca. 16. Mio Computer, Class B-Adressen: 130.83.*.* = ca. 65.000 Computer, usw.)

[Summary] DNS- Server (Domain Name Service) (vgl. S. 173f.)

Die Domain Name Servive sind Namensserver, die eine Art Telefonbuch für IP-Adressen und IP-Namen sind. Wenn ein Computer eine bestimmte Internetseite aufrufen will (www.abenteuerinformatik.de), dann muss er, um die IP- Adresse herauszufinden, bei einem DNS- Server nachfragen. Mit der IP-Nummer, kann dann die Seite aufgerufen werden. Die ICANN (Internet Cooperation for Assigned Names and Numbers) vergibt Domains (Namensbereich) an Personen und Organisationen (Beispielsweise: .de)

[Summary] OSI- Modell: Open Systems Interconnection Reference Model (vgl. S. 176f.)

Das OSI-Modell ist eine siebenschichtige hierarchische Unterteilung der Internetkommunikation. OSI-Layer 1: ist der Physical Layer (physikalische Schicht) und beschreibt das physikalisch vorhandene Übertragungsmedium (Papier und Tinte, elektronische Signale in Netzwerkkabeln). OSI-Layer 2: ist der Data-Link Layer (Datenverbindungsschicht) und sorgt für die zuverlässig Übertragung von Nachrichten (korrekte Adressierung auf dem Brief, IP-Adressen, Prüfnummern). OSI-Layer 3: ist der Network Layer (Netzwerk- oder Paketschicht) und in diesem wird die prinzipielle Vermittlung von Nachrichten im Internet geregelt (Notwendigkeit eines Postmeisters, Router). OSI-Layer 4: ist der Transport Layer (Transportschicht) und regelt die Vermittlung von Datenpaketen von Ursprung bis Ziel (Nachrichten gesendet nach Priorität, Vorfahrtsregeln). OSI-Layer 5: ist der Session Layer (Sitzungsschicht) und hält virtuelle Verbindungen zwischen zwei Computern im Internet aufrecht (Zusammengehörigkeit von Datenpaketen, Kennzeichnung der Zugehörigkeit der Pakete zur Sitzung). OSI-Layer 6: ist der Presentation Layer (Darstellungsschicht) hier werden die Nachrichten codiert, komprimiert und decodiert (die Zahlenkommunikation der Computer in von Menschen lesbare Zeichen umgewandelt und Komprimierung). OSI-Layer 7: ist der Application Layer (Anwendungsschicht) regelt die Kompatibilität verschiedener Computersprachen (HTTP, SMTP (Mail), FTP (Dateien), NTP (Zeitansage)...).

[Summary] Sequentielle Suche (vgl. S. 226f.)

In der gesamte Datenmenge werden alle Elementenacheinander durchgesehen. Bedingung: gesuchte Datei befindet sich im Speicher. Durchschnittlich muss die Hälfte des Speichers durchsucht werden (O (n/2)

[Summary] Binäre Suche (vgl. S. 228f.)

bei eine bereits sortierte Datenmenge (von klein nach groß aufsteigend) wird die Suche in der Mitte begonnen. Ist das Element in der Mitte nicht das gesuchte und hat eine größer Wertigkeit, dann kann der Suchbereich auf die linke Hälfte verkleinert werden (hat sie eine kleinere Wertigkeit, wird

der Suchbereich auf die rechte Hälfte verkleinert). Die Suche wird im neuen Suchbereich wieder in der Mitte begonnen und dasselbe Verfahren wiederholt bis die Karte gefunden wurde. Bedingung: gesuchte Datei befindet sich im Speicher. Hier werden wesentlich weniger Vergleiche benötigt als bei sequentieller Suche. (O (nlogn)

[Summary] Modulo - Mod (vgl. S. 231f.)

Eine eigene Berechnungsart des Rests (bei der Division mit Rest), die man bei der Informatik häufig verwendet. Beispiel: 27 mod 12 = 3 (der ist der Rest bei der Division von 27/12)

[Summary] Hashing (vgl. S. 233f.)

Hashing ist eine effiziente Sortierungmethode von Computern, die (nicht nach menschlichen Kriterien wie Größe der Elemente, sondern) nach einem Sortierkriterium, welches die absolute Position des jeweiligen Elements in der Sotiertabelle genau festlegt, sortiert. Zum Speichern bedarf es im Idealfall nur einen Schreibvorgang und zum Abrufen nur einen Lesevorgang (außer Kollisionen).

Jedes Element hat einen bestimmten Schlüssel (s), welcher der Wert oder die Zeichenfolge ist nach der sortiert wird. Auf diesen Schlüssel wird die jeweilige Hashfunktion (h) angewendet (bspw. mod, Quersummen usw.). Durch Anwendung der Hashfunktion ergibt sich die Speicherposition: Position = h (s).

Ein Beispiel für eine Hashfunktion ist die Sortierung nach Divisionsrest der Schlüsselzahl (meist ist der Divisor eine Primzahl). Beispiel: h (s) = s mod 103

Hashing bedarf eines relativ großen Speichers, der für ein effizientes Hashing nicht vollständig belegt sein sollte. 70-80% Belegung ergeben ein akzeptables Verhältnis von Ausnutzung des Speichers und Häufigkeit der Kollisionen.

[Summary] Kollision und Kollisionsbehandlung beim Hashing (keine Gleichverteilung) (vgl. S. 233f.)

Das Problem beim Hashing ist, dass es immer wieder dazu kommen kann, dass eine Speicherposition bereits belegt ist und es dadurch zu Kollisionen kommt. Das liegt daran, dass es bei (menschlichen) Daten oft zu Häufungen kommt die zusätzlich durch das Dezimalsystem verstärkt werden (Beispiel: Alter bei Krankenkassen). In der Regel wird bei einer Kollision einfach die nächste freie Speicherstelle genommen. Beispiel: hi (s) = (h0 (s) + i) mod 103.

[Summary] (Pseudo-)Zufallszahlen (vgl. S. 245)

Computer sind nicht in der Lage echt Zufallszahlen zu generieren, weil sie diese meist durch eine Funktion (meist durch Rekursion, also eine Rechnung, in der man die vorherige Lösung wieder einsetzt) berechnen. Genauso wie beim nur scheinbar chaotischen Hashing, sind auch die Zahlen die sich ergeben nur scheinbar zufällig. Allerdings ist die Zufallszahl 0 (Startwert) meist eine echte Zufallszahl. Beispiel: z (i) = (z (i-1) * 21 + 17) mod 40.

[Summary] Transpositionsschiffre (vgl. S. 254)

Geheimbotschaften in denen genau dieselben Zeichen enthalten sind wie im Klartext, aber in einer veränderten Reihenfolge. Auch Vertauschungs-Code.

[Summary] Substitutionsschiffre (vgl. S. 254f.)

Jeder Buchstabe des Klartextes wird durch einen anderen Buchstaben ersetzt, der im Alphabet eine entsprechende Zahl von Positionen später kommt. Ist man am Ende des Alphabetes angelangt, fängt man wieder von vorne an. Auch Ersetzungscode.

[Summary] Verschlüsselungsmethoden & ihre Probleme (vgl. S. 259ff.)

Gute Verschlüsslungsmethoden müssen drei Bedingungen erfüllen: die Nachricht darf bei der Übertragung nicht gelesen werden können, und weder verfälscht werden können noch durch eine völlig andere Nachricht ersetzt werden können, ohne dass das vom Empfänger bemerkt wird. Probleme bei der Übertragung: das Weiterleiten der Nachricht kann unterbunden, verfälscht oder vollständig ersetzt werden. Dagegen helfen folgende Methoden: Verschlüsslung, Authentifizierung (Methode zur Unverwechselbaren Identifikation des Urhebers einer Nachricht - mit privatem Schlüssel verschlüsselt) und Signierung durch Certification Authority (Zertifikatsinstanz = eine öffentliche Instanz, die die öffentlichen Schlüssel der Teilnehmer überprüft und archiviert. Sie stellt dann Zertifikate aus). Einer Signatur wird, wenn nötig auch noch ein Zeitstempel beigefügt (um zu verhindern, dass eine Nachricht/Anfrage/ Überweisung noch einmal geschickt werden kann). Grundsätzlich gilt: Kommunikation von Anbieter_innen ist unverschlüsselt aber authentifiziert, Kommunikation von Kund_innen ist verschlüsselt aber unauthentifiziert. Die Sicherheit hängt unter anderem von der Länge der Schlüssel ab.

[Summary] Sicherheitslücken (vgl. S. 283ff.)

Generell der Schlüsselaustausch - gelöst durch asymmetrische Verschlüsselung. Schwachstellen am Beispiel einer Bank: Sicherheit der Bank- Computersysteme (Ziel privater Schlüssel der Bank und Kund_innendaten), Sicherheit der Zertifizierungsinstanz (Ziel privater Schlüssel der CA um falsch Zertifikate auszustellen), Sicherheit asymmetrischer Verfahren (RSA kann rein theoretisch ausgerechnet werden, derzeit dauert das aber ewig), Sicherheit von symmetrischen Verfahren (ähnlich wie bei asymmetrischen Verfahren), Sicherheit des eigenen Computers (Trojaner kann Zertifikate am Computer manipulieren), menschliches Versagen (Reinfallen auf Trickbetrug).

[Summary] Symmetrische Verschlüsselung (vgl. S. 260ff.)

Sender und Empfänger benutzen beide dieselben Ver- und Entschlüsselungsmethode.

[Summary] Asymmetrische Verschlüsselung (vgl. S. 261ff.)

Hier werden Verschlüsselung und Entschlüsselung voneinander entkoppelt. Es wird also mit einem anderen Schlüssel verschlüsselt wie entschlüsselt wird. Oder: eine verschlüsselte Nachricht kann nur mit dem jeweiligen Gegenschlüssel entschlüsselt werden (und anders herum). Ein Schlüssel wird als öffentlicher Schlüssel und einer als privater Schlüssel verwendet. Über eine etablierte

Kommunikation mit Aysmmetrischer Verschlüsslung ist kann sich leicht eine symmetrische Verschlüsslung ausgemacht werden.

[Summary] RSA- Verfahren (vgl. S. 279ff.)

(Benannt nach den Erfindern: Ronald Rivest, Adi Shamir und Leonard Ableman). Beruht auf der Schwierigkeit eine große Zahl in Primfaktoren zu zerlegen (Multiplikation von zwei Primzahlen). Bei RSA werden aus zwei zufällig bestimmten sehr großen Primzahlen nach einem mathematischen Verfahren 2 Schlüssel abgeleitet, von denen man einen als öffentlichen und einen als privaten Schlüssel deklariert. Nachdem die Primzahlen und Zwischenrechnungen vernichtet sind, ist es sehr schwer den privaten aus dem öffentlichen Schlüssel zu berechnen.

[Summary] Steganographie (vgl. S. 287ff.)

Ist eine Methode Nachrichten in einem großen Datenmenge zu verstecken (Leerzeichen in Text, kleine Punkte in Bildern, winzige Pausen in Audiodateien). Das wird auch verwendet um die Urheberschaft mit einem digitalen Wasserzeichen festzuhalten.

[Summary] Fehlerkorrektur (vgl. S. 323ff.)

Um unbeabsichtigt auftretende Fehler zu beheben (Staubkorn auf DVD usw.) werden zusätzliche redundante (also nicht unbedingt notwendige) Daten als Prüfdaten gespeichert. Bei Kredikarten- oder Ausweisnummern gibt es für auftretende Fehler eine Prüfziffer. In Codes werden Fehler verhindert (bzw. leicht auffindbar), wenn die korrekten Symbolfolgen möglichst verschieden sind (Hamming- Distanz = Fehlertoleranz). Beim Erstellen eines Codes wird zunächst mit dem Huffmann-Verfahren Redundanz herabgesetzt und dann gezielt wieder hinzugefügt.

[Summary] Probleme und ihre Lösbarkeit (vgl. S. 339f.)

Ein praktisch nicht lösbares Problem, ist ein Problem, dessen Lösung unverhältnismäßig viel Zeit in Anspruch nehmen würde. Ein lösbares Problem kann durch das Vergrößen der Problemgröße um 1 schon zu einem unlösbaren werden. Das ist häufig bei nicht-skalierenden Probleme, der Fall, wo die Verdopplung des Problems weit mehr als doppelt so viel Zeit zur Lösung in Anspruch nimmt. Ein theoretisch (vom Computer) nicht berechenbares Problem ist eines, für das niemals eine Lösung gefunden werden kann, weil es nicht entscheidbar ist und dadurch einen endlosen Regress ergibt.

[Summary] Terminierung (vgl. S. 350)

Computerprogramme sollen im Normalfall irgendwann einmal mit ihrer Berechnung fertig werden und mit einem Ergebnis aufwarten. In der Fachsprache heißt das Terminierung. Ihnen bekannte Formen von nicht-terminierenden Programmen sind die berühmten “Abstürze” bei denen ein Computer “gar nichts mehr macht”. Effektiv macht der Computer doch noch etwas, aber er führt eventuell immer wieder den gleichen Code in einer Endlosschleife aus und reagiert nicht mehr auf ihre Eingaben - das Programm bzw. Betriebssystem terminiert nie. Ein wichtiger Aspekt der theoretischen Informatik ist die Betrachtung, ob ein Programm immer (also für jede mögliche Eingabe) terminiert. Dafür kann kein Programm geschrieben werden, welches bei anderen Programmen überprüft, ob sie bei allen möglichen Eingaben terminieren.