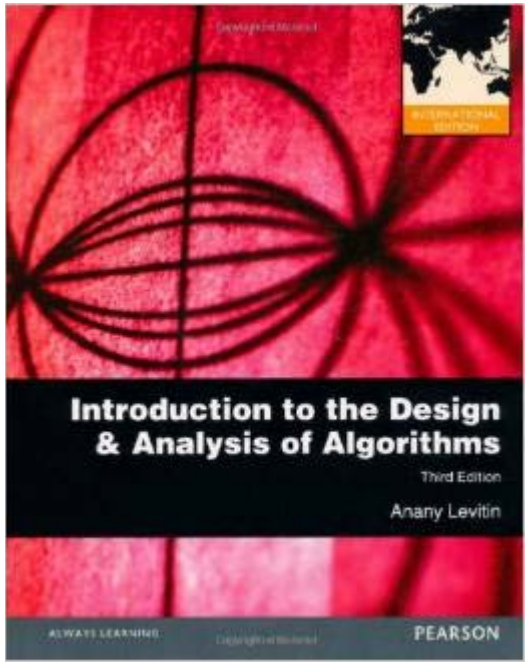
Algorithmen & Datenstrukturen

Greedy-Algorithmen

Literaturangaben

Diese Lerneinheit basiert größtenteils auf dem Buch „The Design and Analysis of Algorithms“ von Anany Levitin.

In dieser Einheit behandelte Kapitel:
9 Greedy Technique
9.4 Huffman Trees and Codes



Greedy-Algorithmen: Grundidee

- Konstruieren **schnell**, **einfach** und **Schritt für Schritt** Lösungen für **Optimierungs-probleme**
- **Hauptmerkmale**
  - **Zulässig**: Zwischenlösungen erfüllen immer die Randbedingungen des Problems
  - **Lokal optimal**: Wähle (gierig) das Element, das momentan am vielversprechendsten erscheint
  - **Unwiderruflich**: Eine getroffene Wahl wird nie rückgängig gemacht
- **Beschränkungen**
  - Greedy-Techniken können einige Problemtypen **optimal lösen**, für andere liefern sie **schnell Näherungslösungen**

Anwendung der Greedy-Technik

- **Optimal lösbare Probleme**
  - Geldwechsel-Algorithmen (normale Stückelungen)
  - Minimalgerüste von Graphen
  - Kürzeste Wege in einem Graphen (single source)
  - Einfache Scheduling-Probleme
  - Rucksackproblem (gebrochener Rucksack)
  - Datenkomprimierung mit Huffman-Codes
- **Näherungsweise lösbare Probleme**
  - Problem des Handelsreisenden
  - Rucksackproblem (0-1-Rucksack)
  - Weitere kombinatorische Optimierungsprobleme

Codierungsproblem

- **Binäre Codierung**: Repräsentation der Buchstaben eines Alphabets durch Codewörter (Bitfolgen)
- Wichtige **Merkmale**
  - **Codewortlänge**
    - Fixe Länge (z. B. ASCII) F = 01000110
    - Variable Länge (z. B. Morsecode) F = ··–·
  - **Präfix-Freiheit**: Kein Codewort ist Präfix eines anderen Codeworts
- **Problem**: Falls Buchstaben **unterschiedlich häufig** auftreten, was ist die effizienteste (= platz-sparendste) binäre, präfix-freie Codierung?

Morse-Code

Morse Code Table

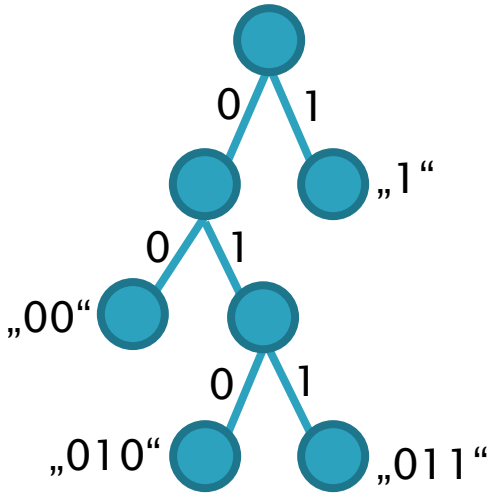
A	.-	N	-.	1	.----	.	.-.-.-
B	-...	O	---	2	..---	,	--..--
C	-.-.	P	.--.	3	...--	?	..--..
D	-..	Q	--.-	4	....-	(	-.--.
E	.	R	.-.	5	.....	)	-.--.-
F	..-.	S	...	6	-....	-	-....-
G	--.	T	-	7	--...	"	.-..-.
H	....	U	..-	8	---..	_	..--.-
I	..	V	...-	9	----.	'	.----.
J	.---	W	.--	0	-----	:	---...
K	-.-	X	-..-	/	-..-.	;	-.-.-.
L	.-..	Y	-.--	+	.-.-.	$	...-..-
M	--	Z	--..	=	-...-		

Variable Codewortlänge

Code präfix-frei?

Huffman-Codes

- Kennzeichne alle **„linken“ Kanten** eines Binärbaums mit 0 und alle **„rechten“ Kanten** mit 1
- Für alle Blätter: Betrachte den Pfad von der Wurzel zum jeweiligen Blatt und notiere die dabei entstehende **Folge von Nullen und Einsen**



- Die Bitfolgen aller Blätter sind einzigartig und **präfix-frei** „0101101010011“

Huffman-Algorithmus

- **Initialisierung**:
  - Erstelle n aus einem Knoten bestehende Bäume mit den Zeichen des Alphabets
  - Gebe jedem Baum die Wahrscheinlichkeit des zugehörigen Buchstabens
- **Wiederhole n-1 mal**:
  - Vereine die beiden Bäume mit den kleinsten Wahrscheinlichkeiten zu einem neuen Baum (mit neuer Wurzel)
  - Der neue Baum erhält die Wahrscheinlichkeit, die der Summe der Wahrscheinlichkeiten der vereinten Teilbäume entspricht
- **Ermittle Codewörter**
  - Alle Buchstaben des Alphabets befinden sich in den Blättern des Baums. Ermittle die zugehörigen Codewörter aufgrund er zu durchlaufenden Kanten.

Huffman-Algorithmus: Beispiel

- Buchstaben: A B C D _
- Häufigkeit: 0,35 0,1 0,2 0,2 0,15
- Codewort: 11 100 00 01 101
- Durchschnittliche Anzahl Bits pro Zeichen: 2,25
- Durchschnittlich Anzahl Bits bei fixer Codierung: 3
- Kompressionsrate: (3-2,25)/3 * 100% = 25%