

## ADP Aufgabe 4 (Vorlagen für die Aufgaben in der mitgelieferten Exceltabelle)

### 4.A

1. Geben Sie für eine Klasse, deren Objekte genau eine Instanzvariable haben, eine `hashCode` Implementierung an, die beim Einfügen von N unterschiedlichen Schlüsseln in eine Hashtabelle, die mit Verkettung arbeitet, zu quadratischem Laufzeitverhalten führt. Zeigen Sie mittels einer Berechnung, wie es zu dem quadratischen Verhalten kommt.
2. Wie sieht der Inhalt der Tabellen beim Verfahren mit linearer Sondierung aus, wenn Sie die `hashCode` Implementierung aus 4.A.1 verwenden?
3. Geben Sie für eine Klasse, deren Objekte genau eine Instanzvariable vom Typ `double` haben eine Implementierung an, die eine ungleichmäßige Verteilung der Schlüssel erzeugt. Vergleichen Sie die Performance Ihrer Implementierung mit einer, die die `hashCode` Berechnung für `double` in Java verwendet.
4. Neben linearer Sondierung gibt es noch andere Strategien zur Auflösung von Kollisionen bei Hashtabellen mit offener Adressierung. Geben Sie ein Beispiel.
5. Konstruieren Sie eine Folge von Einfüge- und Löschoperationen, die für Hashing mit linearer Sondierung, eine maximale Anzahl an Größenänderungen auslöst. Lösen Sie diese Aufgabe nach 4.A.6 zuerst am Beispiel und verallgemeinern Sie dann ihre Beobachtung.

6. Gegeben die folgenden Schlüssel mit Hashwerten für interne Tabellengrößen 4, 8 und 16.

Key	Hash -4	Hash-8	Hash-16	Key	Value
A	1	1	1	W	1
B	2	2	2	A	2
E	1	5	5	G	3
G	3	7	7	E	4
N	2	6	14	N	5
R	2	2	2	B	6
U	1	5	5	A	7
W	3	7	7	U	8
				E	9
				R	10

sowie die Eingabe:

Zeigen Sie, wie Hashing mit linearer Sondierung für die Eingabe arbeitet, wenn die interne Hashtabelle anfangs die Größe 4 hat. Tragen Sie die Key-Value Paare untereinander in die Hashtabelle ein. Markieren Sie in der Tabelle jeweils die Größe der internen Hashtabelle. Wenn sich die Größe der internen Hash-Tabelle ändert, markieren sie die Zeile mit der jeweiligen Operation. Erläutern Sie kurz den Grund für die Größenänderung und tragen Sie dann das Ergebnis in nur zwei Zeilen ein.

Führen Sie abschließend auf dem Endergebnis die Löschoperationen durch.

Verwenden Sie die entsprechende Tabelle in der Excelvorlage.

#### 4.B.1 Schlüsselindiziertes Zählen LSD MSD

1. Was liefert für ein Alphabet die Methode *IgR*?
2. Für welche Eingaben ist schlüsselindiziertes Zählen besonders gut geeignet?
3. Skizzieren Sie mit eigenen Worten den MSD Algorithmus für das Sortieren von Zeichenketten unterschiedlicher Länge.
4. Wie stellt das MSD-Verfahren den korrekten Umgang mit kürzeren Zeichenketten sicher?
5. Für welche Eingaben ist das MSD-Verfahren besonders ungeeignet?
6. Was ist der größte Nachteil des MSD-Verfahrens? Welche Größe spielt dabei eine wesentliche Rolle?
7. Für welche Eingaben würden Sie das MSD-Verfahren dann dennoch empfehlen?
8. Welches alternative Verfahren kennen Sie für das Sortieren von Zeichenketten unterschiedlicher Länge? Wie verhält sich dieses Verfahren bzgl. der Zeichenvergleiche zu einem vergleichbaren Standardsortierverfahren?
9. Sortieren Sie die gezeigte Eingabe nach den LSD Verfahren. Beschriften Sie die Tabelle mit der korrekten Anzahl an Einträgen, den Positionen der Zeichenketten, nach denen sortiert wird, und den einzelnen Phasen des Sortierens in jedem Schritt. Füllen Sie dann die Tabellen entsprechend aus. Sie können 0-wertige Einträge und die Werte, die sich zwischen mehreren Positionen nicht ändern, auslassen. Verwenden Sie die entsprechende Vorlage der Excelschablone.

#### 4.B.1 Schlüsselindiziertes Zählen LSD MSD

1. Was liefert für ein Alphabet die Methode  $lgR$ ?

Anzahl der Bits um einen Index (ein Zeichen) zu repräsentieren.

2. Für welche Eingaben ist schlüsselindiziertes Zählen besonders gut geeignet?

Schlüsselindiziertes Zählen ist bei Anwendungen mit kleinen Zahlen als Schlüssel ein extrem effektives Sortierverfahren.

3. Skizzieren Sie mit eigenen Worten den MSD Algorithmus für das Sortieren von Zeichenketten unterschiedlicher Länge.

Eingabe [a]

0	A	P	F	E	L			
1	A	N						
2	B	L	A	U	B	E	E	R
3	B	A	N	A	N	E		
4	A	P	R	I	K	O	S	E

Häufigkeiten  
zählen  
Indexberechnung

0	0	0	0	
1	A	0	A	0
2	B	6	B	6
3	E	3	E	9
4	F	5	F	14
-	I	1	I	15
5	K	1	K	16
7	L	1	L	17
8	N	2	N	19
9	O	3	O	22
10	P	1	P	23
11	R	2	R	25
12	S	2	S	27
13	U	1	U	28
74		1		29

Verteilungsf

0	0	0
1	A	6
2	B	9
3	E	14
4	F	15
5	I	16
6	K	17
7	L	19
8	N	22
9	O	23
10	P	25
11	R	27
12	S	28
13	U	29
74		29

0	A	N							
1	A	P	F	E	L				
2	A	P	R	I	K	O	S	E	E
3	B	A	N	A	N	E			
4	B	L	A	U	B	E	E	R	

4. Wie stellt das MSD-Verfahren den korrekten Umgang mit kürzeren Zeichenketten sicher?

```
if (hi <= lo + M) {Insertion.sort(a,lo,hi,d); return;} Wechselt zu Insertionsort für kleine Teilarrays
```

Das MSD-Verfahren wechselt zu Insertionsort, wenn untere Grenze (lo) + Grenzwert (M)  $\geq$  obere Grenze (hi) ist.

5. Für welche Eingaben ist das MSD-Verfahren besonders ungeeignet?

MSD-Radix-Sort verhält sich ineffizient bei Eingaben mit vielen gleichen Schlüsseln oder Schlüsseln mit gemeinsamen langen Präfixen, da immer alle Zeichen miteinander verglichen werden müssen.

Bei gleichen Schlüsseln bleibt das Intervall [lo..hi] konstant und die Bedingung für den Wechsel zu Insertionsort wird evtl. nicht erreicht.

6. Was ist der größte Nachteil des MSD-Verfahrens? Welche Größe spielt dabei eine wesentliche Rolle?

Größtes Problem: nicht zufällig verteilte Schlüssel

- lange gemeinsame Abschnitte
- Daten stammen nur aus einem kleinen Wertebereich (unnötig große count[]-Arrays lassen sich nur, wenn der Wertebereich bekannt ist, durch ein passendes Alphabet reduzieren.)  
dann erzeugt MSD-Radix-Sort eine große Anzahl an leeren Teilarrays.

noch:

7. Für welche Eingaben würden Sie das MSD-Verfahren dann dennoch empfehlen?

Im besten Fall: Alle Zeichenketten unterscheiden sich im ersten Zeichen.  
Dann benötigt MSD nur einen Durchgang.

8. Welches alternative Verfahren kennen Sie für das Sortieren von Zeichenketten unterschiedlicher Länge? Wie verhält sich dieses Verfahren bzgl. der Zeichenvergleiche zu einem vergleichbaren Standardsortierverfahren?

Bei Eingaben mit vielen gleichen Schlüsseln oder Schlüsseln mit langem gemeinsamen Präfix werden immer alle Zeichen untersucht. 3-Wege-Quick-Sort für Zeichenketten hingegen untersucht immer nur die nachfolgenden Zeichen bei gemeinsamen Präfix.

9. Sortieren Sie die gezeigte Eingabe nach den LSD Verfahren. Beschriften Sie die Tabelle mit der korrekten Anzahl an Einträgen, den Positionen der Zeichenketten, nach denen sortiert wird, und den einzelnen Phasen des Sortierens in jedem Schritt. Füllen Sie dann die Tabellen entsprechend aus. Sie können 0-wertige Einträge und die Werte, die sich zwischen mehreren Positionen nicht ändern, auslassen. Verwenden Sie die entsprechende Vorlage der Exceltabelle.

		Count[]															Phase				
		Index																			
i		d=2	d=1	d=0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Schlüssel	
0	god	god	get	are		a	b	d	c	g	i	k	l	n	o	r	t	u	x	Zählung	
1	oki	old	bin	axe				3	2	1				1			2			Indexberechnung	
2	bin	grd	oki	bin		0	0	3	3	5	5	6	6	6	7	7	7	9	9	Verteilung	
3	old	axe	old	but				3	5	6		7			9					Zählung	
4	axe	axe	god	get						1		1	1	1	1	2		1	1	Indexberechnung	
5	grd	OKi	grd	god		0	0	0	0	1	1	2	3	4	4	5	7	7	8	9	Verteilung
6	are	bin	are	grd							1	2	3	4	5	7	8			Zählung	
7	get	get	but	oki																Indexberechnung	
8	but	but	axe	old		2	2			3					2					Verteilung	
					0	2	4	4	4	7	7	7	7	7	9	9	9	9	9	Alphabet	
					a	b	d	e	g	i	k	l	n	o	r	t	u	x	Index		

#### 4.C Spezielle Symboltabellen für String-Schlüssel (RST und TST)

1. Konstruieren Sie für die Eingabe `an 0, ant 1, and 2, her 3, friend 4, searching 5, for 6, healthy 7, food 8` aus Schlüssel-Wert Paaren den zugehörigen TST. Sie können die langen Schlüssel ab der Stelle abkürzen, ab der es keine Verzweigung mehr gibt. Schreiben Sie dazu nur den Reststring und den Wert in den TST.
2. Geben Sie ein Beispiel für eine Löschoperationen, die die Struktur des Baumes verändert / nicht verändert.
3. Die Alternative zu einem TST wäre der RST. Stellen Sie die Vorteile und Nachteile beider Implementierungen gegenüber und geben Sie eine Empfehlung für die Verwendung der jeweiligen Implementierung. Vergleichen Sie diese Spezialformen auch mit einer allgemeinen Symboltabellen Implementierung.

#### 4.D Teilstring-Suche

1. KMP:
  - a. Erzeugen Sie für das Phantasie-Muster ALACALAX den DFA des KMP-Algorithmus.
  - b. Zeigen Sie dann für den Text BALAXCALACALACALACALAX wie der KMP-Algorithmus nach dem Muster sucht. Zeigen Sie dazu die Werte des Suchzeigers im Text sowie die Zustand im Automaten, indem Sie die zugehörige Vorlage der Exceltabelle ausfüllen.
2. Boyer Moore: Führen Sie für das Muster B A D E S P A S S und den Text D I E B A D E S A I S O N I S S S P A S S I M N A S S B A D E S P A S S Y E S eine Teilstring-Suche nach dem Verfahren von Boyer-Moore durch. Dokumentieren Sie dabei den Text und Musterindex und zeigen Sie die Berechnung des Folgeindex für i. Füllen Sie die entsprechende Vorlage in der Exceltabelle aus.

#### 4.E Kompression

1. **Lauflängen:** Sie sollen die Bitfolge 1111111 1111111 111 0000000 00 11 00 1 0 1 0 1 00 1111 mit 3 Bits für die Lauflängen komprimieren.
2. **Huffman:** Erstellen Sie für die Text „SONNE MORGEN OHNE SORGEN“ einen präfixfreien Huffman-Code. Schreiben Sie den Huffman Trie binär (sie müssen die Zeichen nicht binär schreiben) und kodieren Sie das erste Wort des Textes. Wenn Sie den Huffman Trie konstruieren, vereinfachen Sie das Einfügen und Löschen in die/der MinPQ, indem Sie immer die ersten beiden Elemente der Queue als Minima annehmen und beim Einfügen das Element mit Häufigkeit X an das Ende aller Elemente mit Häufigkeit x einfügen. Das ergibt zwar nicht die optimale Lösung, aber einen präfixfreien Code. Sie dürfen selbstredend auch die exakte Lösung entwickeln.
3. **LZW:**
  - a. Führen Sie für die Eingabe „SONNE MORGEN OHNE SORGEN“ eine LZW Kompression mit 7 Bit für die Zeichen und 8 Bit für die Codewörter durch. Verwenden Sie die entsprechende Vorlage der Exceltabelle.
  - b. Dekodieren Sie Codefolge: **53 4F 81 83 84**

#### 4.E Kompression

1. **Lauflängen:** Sie sollen die Bitfolge

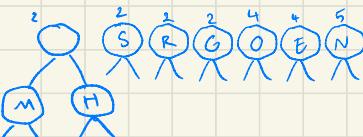
1111111 1111111 111 0000000 00 11 00 1 0 1 0 1 00 1111 mit 3 Bits für die Lauflängen komprimieren.

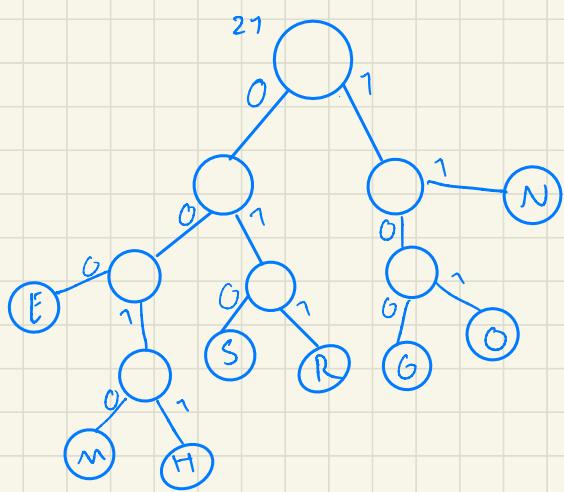
7 Einsen	111	
7 Einsen	111	
3 Einsen	011	
7 Nullen	111	111111011111010010010001001001001001010100
2 Nullen	010	
2 Einsen	010	
2 Nullen	010	
1 Eins	001	
1 Null	001	
1 Eins	001	
1 Null	001	
1 Eins	001	
2 Nullen	010	
4 Einsen	100	

2. **Huffman:** Erstellen Sie für die Text „SONNE MORGEN OHNE SORGEN“ einen präfixfreien Huffman-Code. Schreiben Sie den Huffman Trie binär (sie müssen die Zeichen nicht binär schreiben) und kodieren Sie das erste Wort des Textes. Wenn Sie den Huffman Trie konstruieren, vereinfachen Sie das Einfügen und Löschen in die/der MinPQ, indem Sie immer die ersten beiden Elemente der Queue als Minima annehmen und beim Einfügen das Element mit Häufigkeit X an das Ende aller Elemente mit Häufigkeit x einfügen. Das ergibt zwar nicht die optimale Lösung, aber einen präfixfreien Code. Sie dürfen selbstredend auch die exakte Lösung entwickeln.



Die beiden Bäumen mit den kleinsten Schlüsseln werden zu einem neuen Baum zusammengefügt und wieder eingefügt





3. **LZW:**

- Führen Sie für die Eingabe „SONNE MORGEN OHNE SORGEN“ eine LZW Kompression mit 7 Bit für die Zeichen und 8 Bit für die Codewörter durch. Verwenden Sie die entsprechende Vorlage der Exceltabelle.
- Dekodieren Sie Codefolge: **53 4F 81 83 84**

Eingabe

s

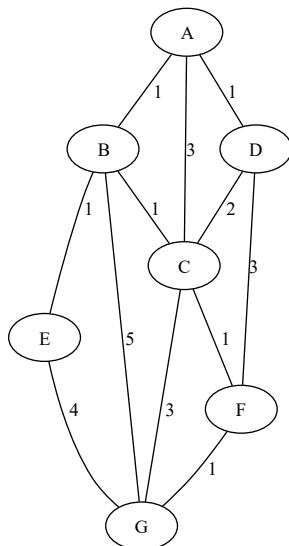
Übereinstimmung

Ausgabe

## 4.F Graphen Dijkstra-A\*-Minimale Spannbäume

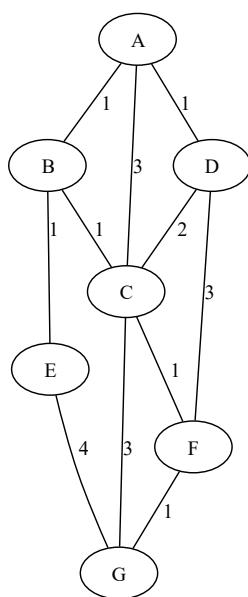
## 1. Dijkstra

Gegeben der folgende Graph. Berechnen Sie nach dem Verfahren der Vorlesung den kürzesten Weg von A nach G. Verwenden Sie die entsprechende Vorlage der Exceltabelle.



2. A\*: Gegeben der folgende Graph sowie eine Heuristik, die die Entfernung eines Knoten zum Zielknoten G schätzt. Zeigen Sie nach dem Verfahren der Vorlesung, welchen Weg A\* vom A nach G berechnet. Verwenden Sie dazu gerne die entsprechende Vorlage der Exceltabelle.

$h(A)=4$
$h(B)=3$
$h(C)=3$
$h(D)=6$
$h(E)=1$
$h(F)=5$
$h(G)=0$



3. Minimale Spannbäume: Gegeben der folgende Graph. Bestimmen Sie mit den Algorithmen nach **Kruskal und Prim** jeweils einen minimalen Spannbaum. Ordnen Sie Knoten und Kanten wie in Skript gezeigt alphabetisch an. Beginnen Sie mit Knoten R beim Algorithmus von Prim. Verwenden Sie dazu gerne auch die entsprechende Vorlage der Exceltabelle.

