Trainingsaufgaben Hashing

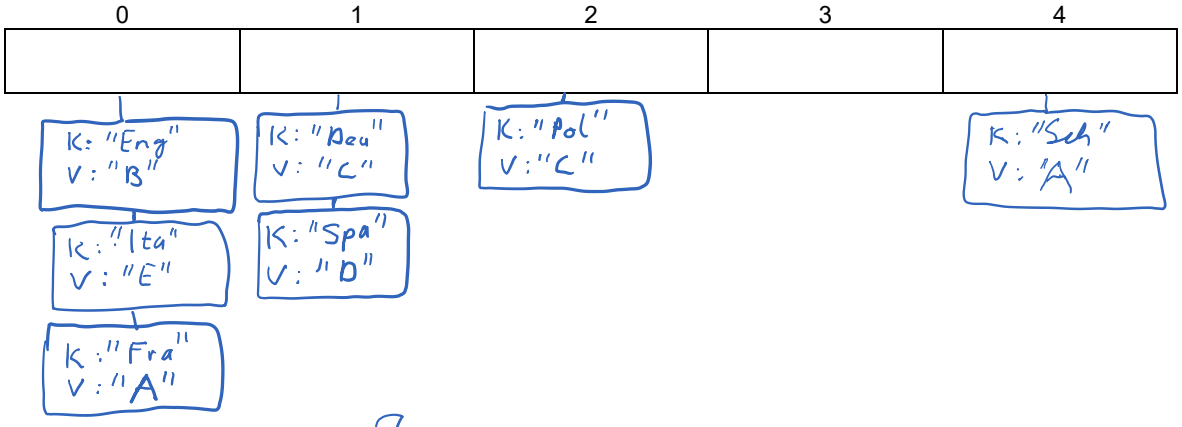
1. Separate Chaining

Bei einer Fussball EM werden die Teams in verschiedene Gruppen eingeteilt. Dazu soll in einer Map zu jeder Mannschaft hinterlegt werden, welcher Gruppe diese zugeordnet ist.

Key	*hashCode* des Keys	Value
England	5	Gruppe **B**
Polen	12	Gruppe **C**
Schweiz	9	Gruppe **A**
Italien	0	Gruppe **E**
Deutschland	6	Gruppe **C**
Frankreich	5	Gruppe **A**
Spanien	31	Gruppe **D**

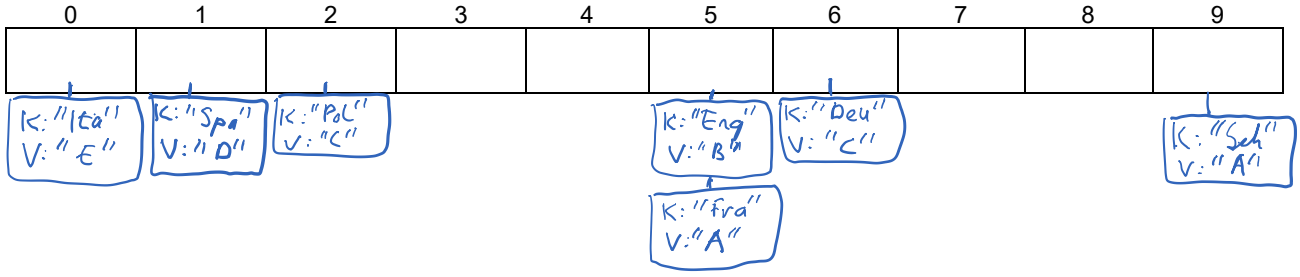
Verwenden Sie in aus Platzgründen vom Key die ersten 3 Buchstaben und für den Value nur den Gruppenbuchstaben (fett markierte Teile).

Tragen Sie die oben aufgelisteten Key-Value Zuordnungen in die Hash-Tabelle unten ein. Verwenden Sie als Strategie zur Kollisionsbehandlung Separate Chaining:



Berechnen Sie den LoadFactor: $\frac{7}{5}$

Aufgrund des LoadFactors entscheiden Sie sich, die Grösser der HashTabelle zu verdoppeln. Erledigen Sie dies und stellen Sie sicher, dass alle Elemente korrekt platziert werden.



2. Open Addressing mit Linear Probing

Als Ausgangslage verwenden wir wieder die Zuweisungen der Fussball-Teams in Gruppen. Tragen Sie die in der Tabelle abgebildeten Key-Value Zuordnungen in die untenstehende HashTabelle der Grösse 8 ein. Verwenden Sie als Strategie zur Kollisionsbehandlung Linear Probing und sondieren Sie **rückwärts**.

Key:	Eng	Pol	Sch	Ita	Deu	Fra	Spa
`hashCode(Key):`	`5`	`12`	`9`	`0`	`17`	`8`	`31`
Value:	B	C	A	E	C	A	D
Start-Index: % 8	5	4	1	0	1	0	7
Kollisionen					2	2	4

Hash-Tabelle:

0	1	2	3	4	5	6	7
K: "Ita" V: "E"	K: "Sch" V: "A"		K: "Spa" V: "D"	K: "Pol" V: "C"	K: "Eng" V: "B"	K: "Fra" V: "A"	K: "Deu" V: "C"

Geben Sie den Load-Factor an: $\frac{7}{8}$

3. Open Addressing mit Double Hashing

Tragen Sie die in der Tabelle abgebildeten Key-Value Zuordnungen in die untenstehende HashTabelle der Grösse 7 ein. Verwenden Sie als Strategie zur Kollisionsbehandlung Double Hashing und sondieren Sie **vorwärts**.

Die Formel für die Schrittgrösse lautet: (`0x7FFFFFFF` steht für `Integer.MAX_VALUE`)

```
step = 1 + (hashCode(Key) & 0x7FFFFFFF) % (size - 2);
```

Key	Eng	Pol	Sch	Ita	Deu	Fra	Spa
hashCode(Key)	5	12	9	0	4	11	33
Value	B	C	A	E	C	A	D
Start-Index: % 7	5	5	2	0	4	4	5
step	1	3	5	1	5	2	4
Kollisionen		1				1	3

Hash-Tabelle (vorwärts sondiert):

0	1	2	3	4	5	6
K: "Ita" V: "E"	K: "Pol" V: "C"	K: "Sch" V: "A"	K: "Spa" V: "D"	K: "Deu" V: "C"	K: "Eng" V: "B"	K: "Fra" V: "A"

Geben Sie den Load-Factor an: $\frac{7}{7} = 1$

Tragen Sie die Zuweisungen nun nochmals in die untenstehende Hash-Tabelle ein. Verwenden Sie wieder Double Hashing, sondieren aber diesmal **rückwärts**. Zählen Sie wieder die Anzahl Kollisionen für jeden Eintrag:

Key	Eng	Pol	Sch	Ita	Deu	Fra	Spa
Kollisionen		1	1		1	4	1

Hash-Tabelle (rückwärts sondiert):

0	1	2	3	4	5	6
K: "Ita" V: "E"	K: "Spa" V: "D"	K: "Pol" V: "C"	K: "Fra" V: "A"	K: "Sch" V: "A"	K: "Eng" V: "B"	K: "Deu" V: "C"

4. Implementation von hashCode und equals

Implementieren Sie für die Klasse `Spiel` die `hashCode()` und `equals()` Methoden. Die Klasse Spiel speichert die Informationen zu einem einzelnen Fussball-Spiel. Die Identität eines Fussallspiels ist definiert durch die beiden Teams und den Timestamp, welche alle nicht null sein dürfen. Gehen Sie davon aus, dass die Klasse `Team` die Methoden `hashCode()` und `equals(…)` korrekt implementiert.

```
public class Spiel {

	private final Team teamA;
	private final Team teamB;
	private final String stadion;
	private final long timestamp;
	private byte goalsTeamA = 0;
	private byte goalsTeamB = 0;


	@Override
	public int hashCode() {
		final int prime = 31;
		int result = teamA.hashCode();
		result = prime * result + teamB.hashCode();
		result = prime * result + (int) (timestamp ^ (timestamp >>> 32));
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (obj instanceof Spiel) {
			Spiel other = (Spiel) obj;
			return other.teamA.equals(teamA)
					&& other.teamB.equals(teamB)
					&& other.timestamp == timestamp;
		}
		return false;

	}
}
```