FACHHOCHSCHULE **TRIER**
Hochschule für Technik, Wirtschaft und Gestaltung
University of Applied Sciences
Informatik

Benutzerhandbuch

Werkzeug zur statistischen Analyse von Zufallsfolgen

Timo Scheit

Bachelor Abschlussarbeit

Betreuer: Prof. Dr. Ing. Alfred Scheerhorn

Trier, 05.12.2007

Inhaltsverzeichnis

1

Einleitung

Im Rahmen des Abschlussprojekts “Design und Entwicklung eines Werkzeugs zur statistischen Analyse von Zufallsfolgen” wurde ein Java-Prgramm zur Generierung und Bewertung pseudozufälliger Bitfolgen implementiert.

Zufallsfolgen werden in der Informatik für verschiedene Aufgaben benötigt. Da ein Rechner normalerweise komplett deterministisch ist und daher keine echten zufälligen Ereignisse erzeugen kann, werden mit Hilfe bestimmter Algorithmen Pseudozufallsfolgen erzeugt. Diese Folgen sollten bestimmte Eigenschaften erfüllen, damit sie als zufällig gelten können. Es existieren verschiedene Tests um dies zu prüfen. Das Programm stellt zum Einen eine Reihe solcher Tests zur Verfügung, zum Anderen enthält es auch einige Zufallsgeneratoren.

Dieses Dokument dient als Benutzerhandbuch zu dem erstellten Programm. Es ist weitgehend identisch mit der Hilfedatei. Neben dem allgemeinen Umgang, werden auch die einzelnen Generatoren und Testverfahren vorgestellt.

2

Starten des Programms

Das Java SE Development Kit 5.0 von Sun wurde zur Implementierung des Programms genutzt. Um das Programm ausführen zu können, muss eine entsprechende Runtime Environment ab Version 5.0 installiert sein. Insofern diese verfügbar ist, sollte das Programm auf allen gängigen Betriebssystemen lauffähig sein. Die Runtime wird unter http://java.sun.com für verschiedene Betriebssysteme angeboten.

Das Programm kann über die Jar-Datei `Zufallsfolgen.jar` gestartet werden. In der Konsole ist dazu folgender Befehl einzugeben:

```
java -jar Zufallsfolgen.jar
```

3

Hauptprogramm

3.1 Umgang mit Tabs

Um die Ergebnisse der verschiedenen Analysen übersichtlich darstellen zu können, nutzt das Programm Tabs. Für jede Analyse wird ein eigener Tab geöffnet. Links neben der Überschrift eines Tabs befindet sich ein Icon, welches den Status des Tabs anzeigt (siehe Abbildung 3.1).

- Grünes Icon: Der Tab zeigt das Ergebnis einer erfolgreich durchgeführten Analyse an
- Rotes Icon: Der Tab zeigt einen Fehler an, der während der Durchführung einer Analyse aufgetreten ist.
- Graues Icon: Der Tab zeigt ein Analyseergebnis oder einen Fehler an, der sich nicht mehr auf die aktuelle Zufallsfolge bezieht.

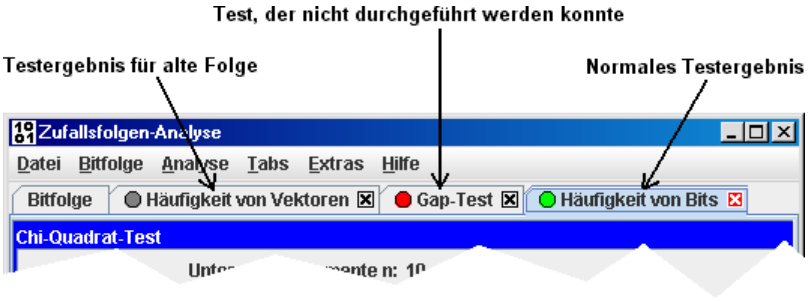

Abbildung 3.1. Icons verschiedener Tabs

Lediglich ein Tab hat kein solches Icon: der Bitfolge-Tab, welcher Informationen über die Bitfolge und deren Generator bereit hält. Dieser ist auch der einzige Tab, der nicht geschlossen werden kann. Lädt oder generiert man eine neue Bitfolge, wird die alte automatisch überschrieben und der Bitfolge-Tab entsprechend aktualisiert. Die Analyse-Tabs können per “Tabs” → “Derzeit gewählter Tab schließen” oder durch Klicken auf den im Reiter befindlichen Schließen-Button, einzeln geschlossen werden. Ein Klick auf “Tabs” → “Alle Analyse-Tabs schließen” schließt alle Tabs bis auf den Bitfolge-Tab. Falls man aus Versehen einen Tab geschlossen

hat, kann man ihn per “Tabs” → “Zuletzt geschlossener Tab wieder öffnen” wieder herstellen. Dies ist nur möglich, wenn der Tab einzeln geschlossen wurde.

3.2 Bitfolge bearbeiten

3.2.1 Erzeugen einer Bitfolge

Es gibt zwei Wege, um eine Bitfolge zu erhalten. Über den Menüpunkt “Bitfolge” kann eine Folge mit Hilfe eines Pseudozufallsgenerators erzeugt werden. Es stehen verschiedene Generatoren zur Verfügung, die mit unterschiedlichen Parametern ausgeführt werden können. Als zweite Möglichkeit kann eine Folge aus einer Binärdatei geladen werden (“Datei” → “Bitfolge öffnen...”). Dadurch ist es möglich, auch Folgen von externen Zufallsgeneratoren zu untersuchen.

3.2.2 Der Bitfolge-Tab

Sobald man zum ersten mal eine Bitfolge erzeugt hat, öffnet sich der Bitfolge-Tab (Abbildung 3.2). Dort sieht man noch einmal, wie die Bitfolge erzeugt wurde und kann sich einen Ausschnitt der Folge anzeigen lassen (Checkbox “Bits x bis y anzeigen”). Über den “Optionen”-Button kann der Ausschnitt verändert werden und vom binären in den hexadezimalen Anzeigemodus gewechselt werden.

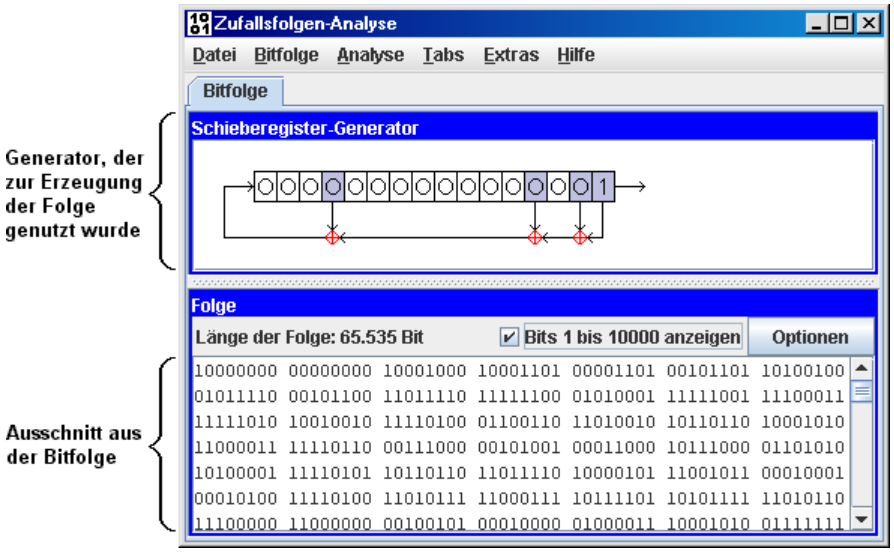

Abbildung 3.2. Bitfolge-Tab

3.2.3 Byte- und Bitorder

In der Informationstechnik haben sich zwei verschiedene Standards zum Speichern von natürlichen Zahlen etabliert. Beim Big-Endian-First-Byte-System ist das erste

Byte das höchstwertige Byte. Das Little-Endian-First-Byte-System beginnt hingegen mit dem niederwertigsten Byte. Entsprechend wäre auch die Reihenfolge der Bits eines einzelnen Bytes jeweils anders zu interpretieren. Das Programm nutzt Big-Endian-First-Byte. Beim Anfügen eines Bytes zur Zufallsfolge wird entsprechend zunächst das MSB übernommen. Sollten dadurch Fehlinterpretationen z.B. beim Laden aus einer Datei entstehen, kann dies mit Hilfe der Funktion "Bitfolge" → "Wörter spiegeln..." behoben werden.

3.2.4 Bitfolge speichern

Falls man eine erzeugte Zufallsfolge speichern möchte, kann man dies über den Menüpunk "Datei" → "Bitfolge speichern..." tun. Die Folge wird in einer Binärdatei gespeichert. Falls das letzt Byte nicht komplett belegt ist, wird es mit '0'-Bits aufgefüllt.

3.3 Durchführen von Analysen

3.3.1 Verschiedene Möglichkeiten

Das Analyse-Menü wird erst verfügbar, sobald eine Bitfolge geladen oder erzeugt wurde. Es stehen eine Reihe von Analysen zur Verfügung, die entweder einzeln oder per automatischem Ablauf ausgeführt werden können.

3.3.2 Durchführung einer einzelnen Analyse

Um eine einzelne Analyse durchzuführen, klickt man im Menü "Analyse" auf den Namen des durchzuführenden Tests. Benötigt der Test zusätzliche Parameter (zu erkennen an den drei Punkten hinter dem Namen), erscheint ein Dialogfenster, um diese abzufragen. Wurden alle Parameter angegeben und mit "OK" bestätigt, wird der Test durchgeführt und es öffnet sich ein neuer Tab mit einer Übersicht über das Testergebnis (siehe Abbildung 3.3). Oben links sieht man (falls vorhanden) noch einmal die verwendeten Testparameter, rechts daneben sind die Parameter und das Resultat der Chi-Quadrat-Analyse (siehe Abschnitt 6.2) aufgeführt. Genauere Analysespezifische Informationen werden darunter tabellarisch und/oder grafisch dargestellt.

3.3.3 Automatischer Ablauf

Wählt man im Analyse-Menü den Punkt "Automatischer Ablauf", werden mehrere Analysen hintereinander ausgeführt und für jedes Testergebnis ein eigener Tab erzeugt. Es werden dabei vom Benutzer keinerlei Test-Parameter abgefragt, sondern Standard-Parameter verwendet. Über den Menüpunkt "Extras" → "Standard-Analysen ändern..." gelangt man zu dem in Abbildung 3.4 dargestelten Dialogfenster, in dem angegeben werden kann, welche Analysen durchgeführt werden sollen und welche Parameter dabei genutzt werden.

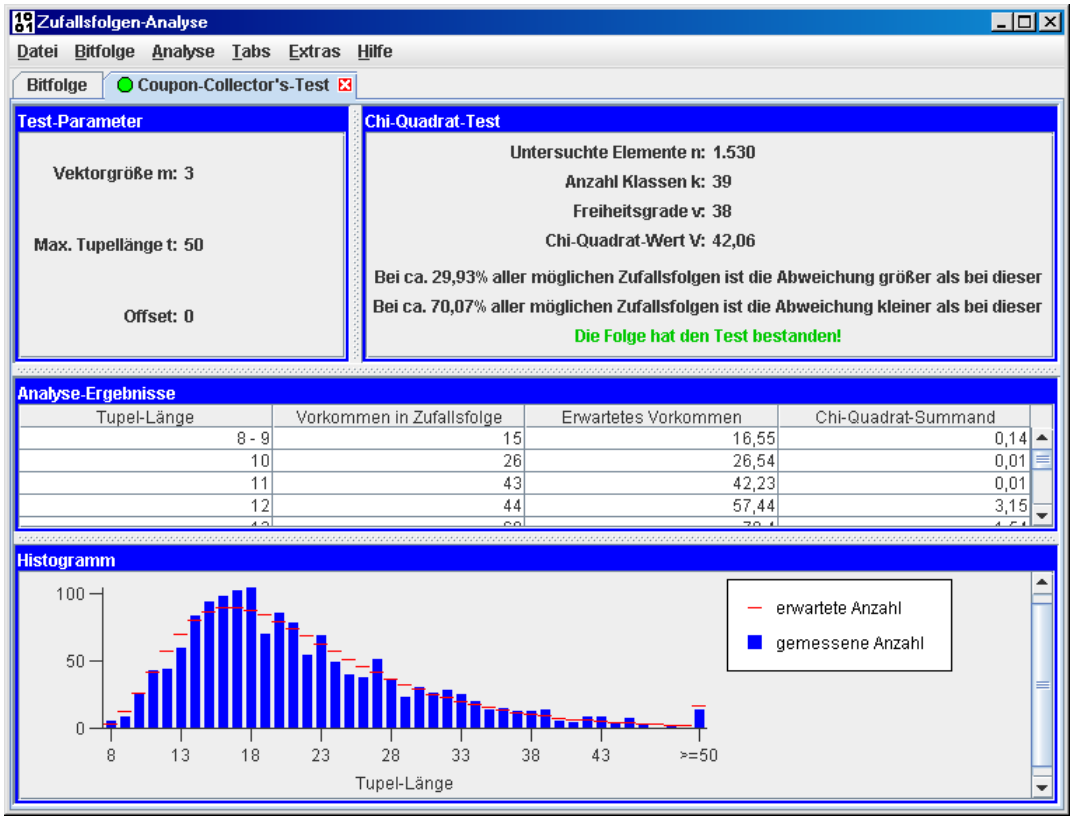

Abbildung 3.3. Testergebnis eines Coupon-Collector's-Tests

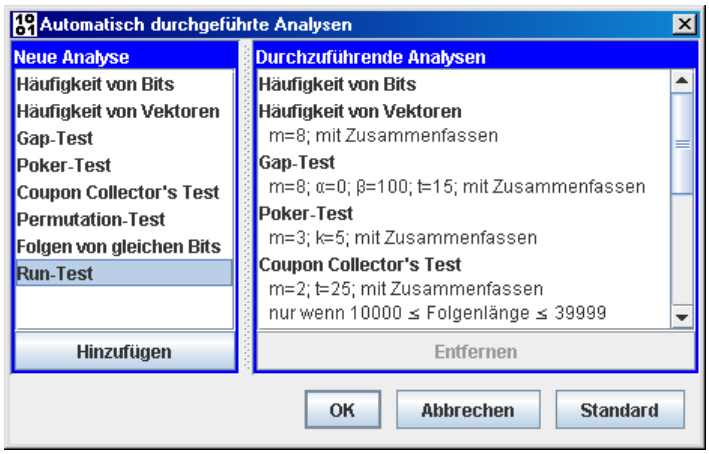

Abbildung 3.4. Dialogfenster für automatisch durchzuführende Analysen

Um eine Analyse hinzuzufügen, wählt man in der linken Liste eine der Verfügbaren Analysen aus und klickt auf "Hinzufügen". Neben den Test-Parametern werden auch die Anforderungen an die Bitfolge erfragt. Liegt die Länge der Bitfolge nicht in dem hier angegebenen Bereich, wird der Test beim automatischen Ablauf übersprungen. Die rechte Liste enthält eine Übersicht über alle auszuführenden Analysen. Überflüssige Analysen können ausgewählt und per Klick auf "Entfernen" gelöscht werden. Alle Änderungen werden erst nach einem Klick auf "OK" übernommen.

3.3.4 Speichern von Analyseergebnissen

Es besteht die Möglichkeit, Analyseergebnisse als Textdatei zu speichern. Über den Menüpunkt "Datei" → "Einzelnes Ergebnis speichern..." wird nur das Ergebnis des gerade gewählten Tabs gespeichert. Wählt man hingegen "Datei" → "Alle aktuellen Ergebnisse speichern...", werden alle Ergebnisse von geöffneten Tabs mit grünem Icon gespeichert. In der resultierenden Textdatei sind dann alle Einzelergebnisse nacheinander aufgeführt.

3.3.5 Mindesterwartungswert

Der Chi-Quadrat-Test kann nur verlässliche Ergebnisse liefern, wenn die Anzahl untersuchter Werte (n) hoch genug ist. Man sollte n so wählen, dass der Erwatungswert für jede Klasse mindestens 5 beträgt. Besser wäre ein noch höherer Wert. Falls die Länge der Folge nicht ausreicht um dies zu erzielen, können Klassen mit geringem Erwartungswert automatisch zusammengefasst werden, so dass sie zusammen das Kriterium erfüllen. Ob die automatische Zusammenlegung von Klassen geschehen soll, kann während der Eingabe der Testparameter bestimmt werden. Das Dialogfeld enthält eine entsprechende Checkbox "Klassen mit geringem Erwartungswert zusammenfassen". Standardmäßig werden die Klassen so zusammengelegt, dass hinterher jede Klasse einen Erwartungswert von mindestens 5 hat. Dieser Wert kann unter "Extras" → "Mindest-Erwartungswert setzen" angepasst werden.

3.3.6 Offset

Sehr viele Tests unterteilen die zu untersuchende Bitfolge in Vektoren fester Länge, und untersuchen dann Eigenschaften dieser Vektoren. Bei allen diesen Test ist es möglich, einen Offset anzugeben. Der Wert gibt an, ob einige Bits am Anfang der Bitfolge übersprungen werden sollen oder ob der erste Vektor mit dem ersten Bit der Folge beginnen soll. Abbildung 3.5 zeigt ein Beispiel mit einem Offset von 4 und einer Vektorlänge von 8. Die Folge wird in diesem Fall quasi so untersucht, als würde sie erst ab dem fünften Bit beginnen.

Abbildung 3.5. Beispiel mit einem Offset von 4

4

Generatoren

4.1 Linearer Kongruenzgenerator

Der lineare Kongruenzgenerator errechnet aus einem ganzzahligen Anfangswert X_0 iterativ mit Formel 4.1 eine Folge ganzzahliger Werte im Bereich von 0 bis $m-1$.

$$X_{n+1} = (aX_n + c) \mod m \tag{4.1}$$

m: Modul

X_n: Ergebnis der vorangegangenen Rechenoperation

a: Faktor

c: Inkrement

Es kann maximal eine Periodenlänge von m ganzzahligen Werten erreicht werden. Um diese zu erzielen, müssen die Parameter folgende Bedingungen erfüllen:

- c und m sind teilerfremd
- $a-1$ ist ein Vielfaches von jedem Primfaktor von m
- wenn m ein vielfaches von 4 ist, dann ist auch $a-1$ ein Vielfaches von 4

Die durch den Generator erzeugte Folge ganzer Zahlen muss abschließend in eine Bitfolge umgewandelt werden. Dazu wird die binäre Darstellung der Zahlen genutzt. Bei dieser sind jedoch je nach Modul nicht immer alle Bits hinreichend zufällig. So haben bei $m = 2^x$ die hinteren (niederwertigen) Bits eine geringe Periodenlänge, das LSB z.B. kann bestenfalls immer zwischen 0 und 1 hin und her springen. Dem kann man entgegen wirken, indem man nicht alle Bits in die Zufallsfolge übernimmt, sondern einige der hinteren Bits ignoriert. Bei m knapp über 2^x und deutlich unter 2^{x+1} ist das MSB in den meisten Fällen 0. Daher kann man dieses Bit ignorieren oder die Zahl mit der Formel 4.2 so umrechnen, dass sie nicht mehr zwischen 0 und $m-1$ liegt, sondern zwischen 0 und $2^x - 1$. Dadurch wird das MSB gleichverteilt.

$$X_n^* = \left\lfloor X_n * \frac{2^x}{m} \right\rfloor \tag{4.2}$$

4.2 Linear rückgekoppeltes Schieberegister

Beim linear rückgekoppelten Schieberegister wird der Register-Inhalt schrittweise um ein Bit verschoben.

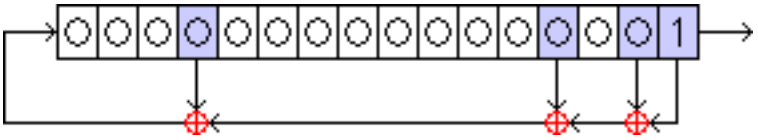

Abbildung 4.1. 16-Bit-Schieberegister

Das letzte Bit (in der Abbildung 4.1 ganz rechts), welches nach dem Schiebevorgang aus dem Register herausfällt, wird in die Zufallsfolge übernommen. Der Bitwert für die frei werdende erste Bitposition wird durch Addition ohne Übertrag (XOR) der Bitwerte fest definierter Registerpositionen ermittelt (in der Abbildung farblich gekennzeichnet).

Die Periodenlänge der erzeugten Bitfolge hängt von der Wahl der Rückkopplungen ab. Für jedes Schieberegister der Länge n ist eine maximale Periodenlänge von $2^n - 1$ möglich. Das Programm schlägt immer passende Rückkopplungen vor, so dass die maximale Periodenlänge erreicht wird. Die Registerlänge ist auf maximal 63 Bit beschränkt.

4.3 AES im Output Feedback Mode

Blockchiffren können, wenn sie im Output Feedback (OFB) Mode betrieben werden, eine Pseudozufallsfolge erzeugen. Das Programm nutzt dazu den Advanced Encryption Standard (AES) als Block-Chiffre. An Parametern erhält der Generator einen Schlüssel (K), sowie einen Initialisierungsvektor (IV), beide 128 Bit lang.

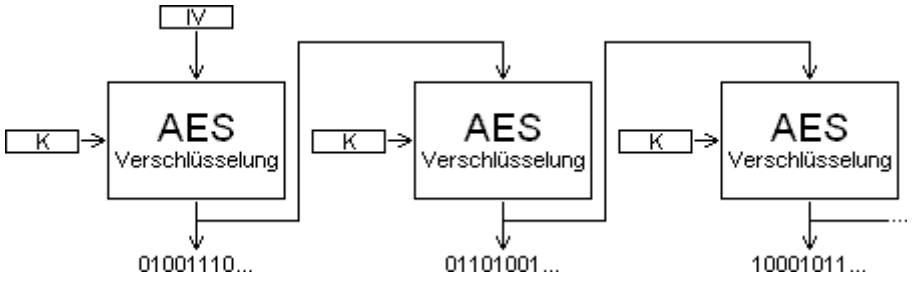

Abbildung 4.2. Funktionsweise des Generators

Abbildung 4.2 veranschaulicht das Funktionsprinzip des Generators. Zunächst wird der Initialisierungsvektor verschlüsselt. Die resultierenden 128 Bit werden in die Zufallsfolge übernommen und dienen zugleich als nächster zu verschlüsselnder Vektor. Dies kann so oft wiederholt werden, bis die benötigte Anzahl an Bits erzeugt wurde.

5

Testverfahren

5.1 Häufigkeit von Bits

Dieser Test zählt lediglich die Anzahl von Null- und Eins-Bits in der Bitfolge. Zu erwarten wäre eine Gleichverteilung, also dass bei n Bits jeweils ungefähr $n/2$ 0- und 1-Bits vorkommen.

5.2 Häufigkeit von Vektoren

Dieser Test ist eine Erweiterung des Tests "Häufigkeit von Bits". Anstatt nur einzelne Bits zu untersuchen, wird die Bitfolge in Vektoren mit jeweils m Bits aufgeteilt. Anschließend wird gezählt, wie oft die einzelnen Vektoren vorkommen. Es sind 2^m verschiedene Vektorwerte möglich. Wiederum wird eine Gleichverteilung erwartet.

5.3 Gap-Test

Der Gap-Test untersucht die Lückengröße zwischen zwei Werten aus einem bestimmten Wertebereich. Dazu wird die Bitfolge zunächst in Vektoren der Länge m aufgeteilt. Die Bitbelegung der einzelnen Vektoren wird als binäre Darstellung eines ganzzahligen Wertes interpretiert. Auf diese Weise erhält man eine Folge zufälliger ganzer Zahlen im Bereich von 0 bis $2^m - 1$. Der Wertebereich wird durch die Grenzen α und β definiert. Für alle Werte U_j im Wertebereich gilt: $\alpha \leq U_j < \beta$. Die Lückengröße ist die Anzahl an Werten außerhalb des Wertebereichs, die zwischen zwei Werten innerhalb des Wertebereichs liegen. Als Ergebnis liefert der Test für verschiedene Lückengrößen die Anzahl vorgekommener Lücken. Es sollten vor allem eher kurze Lücken auftreten. Je nach Wertebereich ist dieses Verhalten unterschiedlich stark ausgeprägt.

5.4 Poker-Test

Dieser Test ist der Wertung beim Poker-Spiel nachempfunden. Ähnlich einer Hand voll Karten, wird ein Tupel aus mehreren Vektoren untersucht. Dazu wird die

Bitfolge zunächst in Vektoren der Länge m aufgeteilt. Jeder Vektor hat einen von 2^m möglichen Werten. Die Folge von Vektoren wird wiederum in Tupel von k Vektoren aufgeteilt. Klassischerweise haben die Tupel eine Länge von 5. Wie bei einem Pokerspiel wird dann untersucht, ob bei den 5 Vektoren (Analog zu 5 Karten auf der Hand) Vektoren mit dem selben Wert vorkommen. Das Tupel kann dann in verschiedene Klassen eingeteilt werden:

- alle fünf Werte unterschiedlich
- zwei Vektoren gleich (ein Paar)
- zwei mal zwei Vektoren gleich (zwei Paare)
- drei Vektoren gleich (Drilling)
- ein Drilling und ein Paar (Full-House)
- vier Vektoren gleich (Vierling)
- fünf Vektoren gleich

Um den Test jedoch gerade im Hinblick auf andere Werte für k zu vereinfachen, wird lediglich die Anzahl unterschiedlicher Vektoren im Tupel unterschieden. Dadurch fallen einige der oben genannten Klassen zusammen, der Test ist aber immer noch hinreichend aussagekräftig. Als Ergebnis liefert der Test für jede Anzahl unterschiedlicher Vektorwerte die Anzahl vorgekommener Tupel. In den meisten Fällen kommen dabei Tupel mit einer sehr geringen Zahl unterschiedlicher Vektorwerte nur sehr selten vor.

5.5 Coupon-Collector's-Test

Dieser Test bildet das Sammeln von Sammelmarken (Coupons) ab. Zunächst wird die Bitfolge in Bit-Vektoren der Länge m aufgeteilt. Dadurch entstehen 2^m mögliche Vektorwerte. Die Vektoren stehen stellvertretend für die Sammelmarken. Es wird gemessen, wie viele Vektoren eingelesen (gesammelt) werden müssen, bis jeder Vektorwert mindestens einmal vorgekommen ist. Die Anzahl eingelesener Vektoren wird als Tupelgröße r bezeichnet. Als Ergebnis liefert der Test für verschiedene Tupelgrößen die Anzahl vorgekommener Tupel. Schon bei geringen Vektorlängen kann es sehr lange dauern, bis ein Tupel komplett ist.

5.6 Permutation-Test

Dieser Test untersucht die Reihenfolge in einem Tupel von Vektoren. Zunächst wird die Bitfolge in Vektoren der Länge m aufgeteilt. Die Bitbelegung der einzelnen Vektoren wird als binäre Darstellung eines ganzzahligen Wertes interpretiert. Die resultierende Folge ganzzahliger Werte wird wiederum in Tupel der Länge t aufgeteilt. Der Test untersucht die Sortierung der Werte innerhalb des Tupels. Jede Reihenfolge bildet eine eigene Klasse, der mithilfe des fakultätsbasierten Zahlensystems eine spezielle Permutations-Nummer zugeordnet wird (siehe Dokumentation). Der Fall, dass zwei Vektoren den selben Wert haben können, wird dabei

vernachlässigt. Als Ergebnis liefert der Test für jede Permutations-Nummer die Anzahl vorgekommener Tupel. Statistisch gesehen sollte jede Sortierung ungefähr gleich oft vorkommen.

5.7 Folge von gleichen Bits

Dieser Test untersucht, wie viele Bits mit dem gleichen Wert aufeinander folgen. Ein Block bestehend aus gleichen Bits wird als Run (Reihe, Serie) bezeichnet (siehe Abb. 5.1).

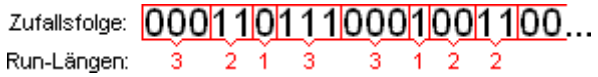

Abbildung 5.1. Beispiel einer in Runs zergliederten Zufallsfolge

Man kann auswählen, ob nur die '0'-Runs (also Runs bestehend aus '0'-Bits), nur die '1'-Runs oder alle Runs untersucht werden sollen. Als Ergebnis liefert der Test für verschiedene Run-Längen die Anzahl vorgekommener Runs. Zu erwarten wäre, dass ca. die Hälfte aller Runs die Länge 1 hat. Pro zusätzliches Bit halbiert sich der Erwartungswert.

5.8 Run-Test

Der Run-Test untersucht die Anzahl an direkt aufeinander folgenden monoton steigenden oder monoton fallenden Werten. Dazu wird die Bitfolge zunächst in Vektoren der Länge m aufgeteilt und die Bitbelegung der einzelnen Vektoren wie bei Permutation- und Gap-Test als binäre Darstellung eines ganzzahligen Wertes interpretiert, wodurch eine Folge natürlicher Zahlen entsteht. Der Test kann entweder "Runs Up" oder "Runs Down" untersuchen, also entweder monoton steigende oder monoton fallende Ausschnitte der Folge.

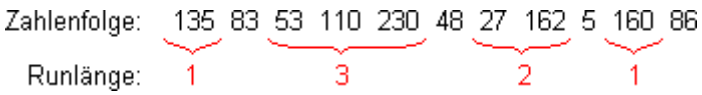

Abbildung 5.2. Runs Up in einer Folge natürlicher Zahlen

Abbildung 5.2 zeigt ein Beispiel, bei dem Runs Up untersucht wurden. Ein Run Up endet, sobald ein Wert erreicht wird, der kleiner als der Vorgängerwert ist (z.B. $83 < 135$). Der nächste Run beginnt dann nicht etwa mit dem Wert, der den vorherigen Run unterbrochen hat, sondern erst mit dem Wert danach (im Beispiel also mit 53 statt 83). Andernfalls wären die Runs nicht unabhängig voneinander und das Ergebnis daher schwer auszuwerten. Als Ergebnis liefert der Test für verschiedene Runlängen die Anzahl vorgekommener Runs.

6

Auswerten der Testergebnisse

6.1 Auswertungs-Grundlage

Allen vorgestellten Tests ist gemeinsam, dass sie eine Reihe von endlichen Klassen bilden und für jede Klasse ein Erwartungswert sowie die wirkliche Anzahl an Vorkommen, die zu der jeweiligen Klasse gehören, ermittelt wird. Durch den Vergleich dieser Soll- und Ist-Verteilung kann berechnet werden, wie viel Prozent aller möglichen Zufallsfolgen eine niedrigere oder höhere Abweichung besitzen. Um dies zu berechnen, bietet sich der Chi-Quadrat-Test an.

6.2 Chi-Quadrat-Test

Der Chi-Quadrat-Test ermittelt für jede Klasse die Abweichung von der Anzahl an Elementen zum Erwartungswert. Diese Abweichung wird quadriert und durch den Erwartungswert geteilt. Auf diese Weise entsteht für jede Klasse ein, je nach Erwartungswert gewichteter, Chi-Quadrat-Summand. Addiert man diese alle zusammen, ergibt sich der Chi-Quadrat-Wert V (siehe Formel 6.1). Der Erwartungswert E_s entspricht $n * p_s$, wobei n für die Anzahl untersuchter Elemente und p_s für die Wahrscheinlichkeit, dass ein Element zur Klasse s gehört, steht. Der Erwartungswert sollte für jede Klasse einen bestimmten Wert nicht unterschreiten. Ein üblicher Mindestwert ist 5. Sollte eine Klasse einen niedrigeren Erwartungswert haben, kann sie mit einer anderen Klasse vereinigt werden.

$$V = \sum_{s=1}^{k} \frac{(Y_s - E_s)^2}{E_s} \tag{6.1}$$

Y_s: gemessene Anzahl für die Klasse s

E_s: Erwartungswert für die Klasse s

k: Anzahl Klassen

Der Chi-Quadrat-Wert kann zusammen mit der Anzahl an Freiheitsgraden ν, welche in allen vorgestellten Tests um eins geringer als die Anzahl an Klassen, also $\nu = k - 1$ ist, dazu verwendet werden, das Akzeptanzlevel zu bestimmen.

6.3 Ergebnis einordnen

Das Akzeptanzlevel ist ein Prozentualer Wert, der angibt, wie viele Folgen gleicher Länge (bzw. gleicher Anzahl an Runs/Tupel/Gaps) eine geringere Abweichung von den Erwartungswerten haben. Ideal wäre ein Akzeptanzlevel von ca. 50%. Werte über 95% oder unter 5% lassen darauf schließen, dass die Folge Mängel bezüglich ihrer Zufälligkeit hat. Liegt das Akzeptanzlevel gar über 99% oder unter 1%, dann ist es sehr wahrscheinlich, dass die Folge nicht hinreichend zufällig ist.