Universität des Saarlandes
FR 6.2 Informatik
Prof. Dr. Hans-Peter Lenhof
Dipl. Inform. Andreas Hildebrandt

Programmierung II, SS 2003
Musterlösung zur 6. Übung

Aufgabe 1: Faire Münzen (10 Punkte)
Offensichtlich ist jeder Münzwurf ein von den anderen Würfen unabhängiges Ereignis. Daher werfen wir die Münze zwei mal hintereinander, und sehen uns das Ergebnis an. Bezeichnen wir die (unbekannte) Wahrscheinlichkeit für Kopf mit p und die für Zahl mit $q = 1 - p$, so kommen nach zweimaligem Werfen die möglichen Ereignisse mit folgenden Wahrscheinlichkeiten vor:

Kopf Kopf	$p \times p$
Zahl Zahl	$q \times q$
Kopf Zahl	$p \times q$
Zahl Kopf	$q \times p$

Da die Multiplikation in $\mathbb{R}$ praktischerweise kommutativ ist, sind die Wahrscheinlichkeiten für die letzten beiden Ereignisse, also Kopf-Zahl und Zahl-Kopf, offensichtlich gleich!
Dies können wir ausnutzen, indem wir solange unsere Münze zwei mal hintereinander werfen, bis sich die Ergebnisse zweier aufeinanderfolgender Würfe unterscheiden, also auf Zahl Kopf oder auf Kopf Zahl folgt. Den Fall Kopf-Zahl nennen wir einfach *Kopf* und den Fall Zahl-Kopf *Zahl*, und da beide mit gleicher Wahrscheinlichkeit auftreten, haben wir uns einen fairen Münzwurf gebastelt.

Wie sieht nun der Erwartungswert der benötigten Anzahl X von Münzwürfen aus? Dazu definieren wir für zwei aufeinanderfolgende Münzwürfe die beiden neuen Elementarereignisse e_1 und e_2: e_1 bedeutet, dass wir zwei unterschiedliche Ergebnisse geworfen haben, e_2 steht für zwei gleiche. Damit ergibt sich für die elementaren Wahrscheinlichkeiten:

$$p(e_1) = pq + qp = 2pq, p(e_2) = p^2 + q^2$$

Wieviele dieser Elementarereignisse müssen wir nun im Mittel durchführen, bevor wir zu einem Ergebnis gelangen? Wir benötigen genau dann n solcher Münzwurfpaare, wenn $n-1$ mal e_2 eintritt und dann einmal e_1. Da diese Ereignisse alle voneinander unabhängig sind, ergibt sich für den Erwartungswert der benötigten Anzahl an Münzwurfpaaren Y:

$$
\begin{aligned}
E[Y] &= \sum_{n\in\mathbb{N}} np(e_2)^{n-1}p(e_1) \\
&= p(e_1)\sum_{n=0}^{\infty} np(e_2)^{n-1} \\
&= p(e_1)\sum_{n=0}^{\infty} \frac{d}{dp(e_2)}p(e_2)^n \\
&= p(e_1)\frac{d}{dp(e_2)}\sum_{n=0}^{\infty} p(e_2)^n \\
&= p(e_1)\frac{d}{dp(e_2)}\left(\frac{-1}{p(e_2)-1}\right) \\
&= \frac{p(e_1)}{(p(e_2)-1)^2} \\
&= \frac{2p(1-p)}{(p^2+(1-p^2)-1)^2} \\
&= \frac{1}{2pq}
\end{aligned}
$$

Und da jedes dieser Paare aus 2 Münzwürfen besteht, gilt für die Anzahl benötigter Würfe: $E[X] = 2E[Y] = \frac{1}{pq}$
Nehmen wir z.B. an, das die Münze schon ursprünglich fair war, so gilt $E[X] = \frac{1}{\frac{1}{4}} = 4$.

Aufgabe 2: Cauchy-Schwarz für Erwartungswerte (10 Punkte)
Zu zeigen ist die Cauchy-Schwarz-Ungleichung für Erwartungswerte:

$$(E[XY])^2 \leq E[X^2]E[Y^2]$$

- **zu Zeigen:** für $p(\{Y=0\}) = 1$ ist Cauchy-Schwarz (sogar mit Gleichheit) erfüllt. **Beweis:** Es gilt: $(E[XY])^2 = \left(\sum_{e\in S} p(e)X(e)\cdot Y(e)\right)^2$. Diese Summe zerfällt in zwei Teile: in eine Summe über Ereignisse e, für die $Y(e) = 0$ gilt, und über solche mit $Y(e) \neq 0$. Wir wissen, dass $p(Y=0) = 1$, dass also für alle Ereignisse e mit $Y(e) \neq 1$ gilt: $p(e) = 0$. Damit erhalten wir für die beiden Summen:

  $$(E[XY])^2 = \left(\sum_{e\in S, Y(e)=0} p(e)X(e)\cdot \underbrace{Y(e)}_{=0} + \sum_{e\in S, Y(e)\neq 0} \underbrace{p(e)}_{=0} X(e)\cdot Y(e)\right)^2 = 0$$

  Analog können wir zeigen, dass $E[Y^2] = 0$, und damit reduziert sich für den Fall $p(\{Y=0\}) = 1$ die Cauchy-Schwarz Ungleichung auf die triviale wahre Aussage $0 \leq 0$. ■

- Sei $P(\{Y=0\}) \neq 1$, $h(a) := E[(X-aY)^2]$

(a) **zu Zeigen:** $h(a) \geq 0$
Beweis: $h(a) = E[(X - aY)^2] = \sum_{e \in S} \underbrace{p(e)}_{\geq 0} \underbrace{(X(e) - aY(e))^2}_{\geq 0}$ ist eine Summe von nichtnegativen Termen und damit natürlich selbst nichtnegativ $\Rightarrow$ Behauptung.■

(b) **zu Zeigen:** $h(a)$ hat bei $a^* := \frac{E[XY]}{E[Y^2]}$ ein Minimum.
Beweis: Betrachte

$$\begin{aligned}
\frac{dh(a)}{da} &= \frac{d}{da} \sum_{e \in S} p(e)(X(e) - aY(e))^2 \\
&= \sum_{e \in S} p(e) \frac{d}{da} (X(e) - aY(e))^2 \\
&= \sum_{e \in S} p(e)\, 2\, (X(e) - aY(e))(-Y(e)) \\
&= E[2\,(X - aY)(-Y)] \\
&\underset{\text{E linear}}{=} 2(aE[Y^2] - E[XY])^2
\end{aligned}$$

An der Stelle $a^* = \frac{E[XY]}{E[Y^2]}$ haben wir damit $\frac{dh(a)}{da}|_{a^*} = 2\left(\frac{E[XY}{E[Y^2]} E[Y^2] - E[XY]\right) = 2(E[XY] - E[XY]) = 0 \Rightarrow a^*$ ist Extremum von $h(a)$. Und mit $\frac{d^2h(a)}{da^2}|_{a^*} = 2E[Y^2] \geq 0$ folgt, dass a^* Minimum von $h(a)$ ist.■

- **zu Zeigen:** für $P(\{Y = 0\}) \neq 1 (\Rightarrow E[Y^2] \neq 0)$ gilt $(E[XY]^2) \leq E[X^2]E[Y^2]$.
Beweis: Nach Teil (b) gilt:

$$\begin{aligned}
h(a^*) &\geq 0 \\
\Leftrightarrow E[\left(X - \frac{E[XY]}{E[Y^2]} E[Y]\right)^2] &\geq 0 \\
\Leftrightarrow E[X^2 + \frac{E[XY]^2}{E[Y^2]^2} Y^2 - 2XY \frac{E[XY]}{E[Y^2]}] &\geq 0 \\
\Leftrightarrow E[X^2] + \frac{E[XY]^2}{E[Y^2]^2} E[Y^2] - \frac{2E[XY]^2}{E[Y^2]} &\geq 0 \\
\Leftrightarrow E[X^2] - \frac{E[XY]^2}{E[Y^2]} &\geq 0 \\
\Leftrightarrow E[X^2] &\geq \frac{E[XY]^2}{E[Y^2]} \\
\Leftrightarrow E[X^2]E[Y^2] &\geq E[XY]^2
\end{aligned}$$

■

Damit ist die Cauchy-Schwarz Ungleichung bewiesen!

Aufgabe 3: Stochastik (10 Punkte)

(a) Wir partitionieren den Ereignisraum in die Ereignisse K^+ und $K-$. Dann können wir nach dem Satz von Bayes die gesuchte Wahrscheinlichkeit $P(K^+|T^+)$ folgendermaßen berechnen:

$$\begin{aligned} P(K^+|T^+) &= \frac{P(T^+|K^+)P(K^+)}{P(T^+|K^+)P(K^+) + P(T^+|K^-)P(K^-)} \\ &= \frac{P(T^+|K^+)P(K^+)}{P(T^+|K^+)P(K^+) + (1 - P(T^-|K^-))(1 - P(K^+))} \end{aligned}$$

(b) Setzen wir die gegebenen Werte in $P(K^+|T^+)$ ein, so ergibt sich folgendes Resultat:

$$\begin{aligned} P(K^+|T^+) &= \frac{0.997 \times 0.0005}{0.997 \times 0.0005 + (1 - 0.97) \times (1 - 0.0005)} \\ &\approx 0.016 \end{aligned}$$

Die Wahrscheinlichkeit, dass der Patient tatsächlich erkrankt ist, liegt also bei ca. 1.6%! Das dieser positive Vorhersagewert trotz der eigentlich auf den ersten Blick recht gut aussehenden Daten von ELISA so erstaunlich schlecht ausfällt, liegt an der zum Glück sehr geringen Prävalenz. Da nur ein winziger Teil der Gesamtbevölkerung erkrankt ist, gewinnen die falsch positiven Ergebnisse gegenüber den falsch negativen an Gewicht. Wird z.B. ein Patient aus einer Riskogruppe mit höherer Prävalenz getestet, verbessert sich auch der positive Vorhersagewert.

Warum verwendet man überhaupt einen Test, der eine so miserable Vorhersagequalität besitzt? Dazu muss man sich nochmal vor Augen führen, welche zwei Fehler ein Test machen kann: er kann falsch positive oder falsch negative Vorhersagen machen. Offensichtlich ist die Rate der falsch positiven Ergebnisse bei ELISA sehr hoch (obwohl aufgrund der geringen Prävalenz nur sehr selten überhaupt positive Resultate vorkommen). Dafür ist die Rate der falsch negativen Ergebnisse äußerst gering! Der *negative Vorhersagewert* $P(K^-|T^-)$ liegt, wie man sich leicht ausrechnen kann, bei ca. 0.99! Aufgrund der hohen Ansteckungsgefahr bei HIV ging es den Entwicklern von ELISA in erster Linie darum, einen schnellen und billigen Test zu entwickeln, der äußerst selten einen kranken Patienten für gesund hält, dafür jedoch häufiger "falschen Alarm´´ schlägt. Daher wird auch jeder Patient, der positiv getestet wurde, mit noch mindestens einem anderem (teureren und aufwendigeren) Test mit besserem positivem Vorhersagewert getestet.

Aufgabe 4: Die Binomialverteilung (10 Punkte)
Die *Binomialverteilung* ist definiert als:

$$P(\{X = k\}) := \binom{n}{k} p^k q^{n-k}$$

Bestimmen wir den Erwartungswert $E[X]$ einer binomialverteilten Zufallsvariablen X:

$$
\begin{aligned}
E[X] &= \sum_{k=1}^{n} k \binom{n}{k} p^k q^{n-k} \\
&= np \sum_{k=1}^{n} \binom{n-1}{k-1} p^{k-1} q^{n-k} \\
&= np \sum_{k=0}^{n-1} \binom{n-1}{k} p^k q^{(n-1)-k} \\
&= np \underbrace{(p+q)}_{} = 1^{n-1} \\
&= np
\end{aligned}
$$

Aufgabe 5. (STL, Funktionsobjekte Bonuspunkte)
10

```
#include <algorithm>
#include <vector>
#include <iostream>

using namespace std;

// Die Klasse, die unseren Algorithmus zur Simulation einer fairen Muenze enthaelt
class FaireMuenze
{
  public:
    // Der Konstruktor. Wir erlauben hier _keinen_ Defaultkonstruktor
    // ohne Argumente, um sicherzustellen, dass wir von Anfang an
    // einen Zufallszahlengenerator zur Verfuegung haben
    FaireMuenze( double (*f)(), double min, double max );
    ~FaireMuenze();

    bool operator() ();

  protected:
    // Der Zufallszahlengenerator
    double (*f_)();
    // Die Schranken des Wertebereichs von f
    double min_;
    double max_;
    // Der Wert, der spaeter ueber "Kopf oder Zahl" entscheidet
    double mid_;
};

// Im Konstruktor ist nicht viel zu tun, wir m"ussen nur die
```

```
// "ubergebenen Werte kopieren und das Mittel von min und max bilden
FaireMuenze::FaireMuenze( double (*f)(), double min, double max )
  : min_(min),
    max_(max)
{
  f_ = f;
  mid_ = (max_ - min_)/2.;
}

// Der Destruktor ist auch ziemlich einfach... :-)
FaireMuenze::~FaireMuenze()
{
}

// Hier kommt endlich unser Algorithmus aus Aufgabe 1
bool FaireMuenze::operator() ()
{
  // Wir brauchen eine Zufallszahl
  double z;

  // Und wir muessen uns die letzten Wuerfe merken
  bool erster_wurf=false, wurf=false;

  // Zuerst generieren wir mal einen zuf"alligen Wurf mit der
  // unfairen Muenze. Wenn der Generator einen Wert > mid
  // zuruecklliefert, war's ein Kopf, sonst eine Zahl
  // und dann werfen wir so lange, bis sich die Ergebnisse
  // von zwei Wuerfen unterscheiden

  while (erster_wurf == wurf)
  {
    z = f_();
    if (z > mid_)
    {
      erster_wurf = true;
    }
    else
    {
      erster_wurf = false;
    }
    z = f_();

    if (z > mid_)
    {
      wurf = true;
    }
    else
    {
```

```
      wurf = false;
    }
  }

  // erster_wurf und wurf unterscheiden sich jetzt. Da wir vereinbart hatten,
  // dass das Ergebnis des gesamten Wurfs gleich dem Ergebnis des ersten Wurfs
  // ist, sind wir auch schon fertig
  return erster_wurf;
}


// Ein paar tests
int main(int argc, char **argv)
{
  // ein vector mit Wuerfen
  vector<bool> wuerfe(1000);

  // Unser Generator
  srand48(42);

  // Und ein erster Test... Hier sollte die Muenze schon von Anfang
  // an recht fair sein...
  generate(wuerfe.begin(), wuerfe.end(), FaireMuenze(drand48, 0., 1.));

  // Zaehlt die Anzahl an Koepfen die geworfen werden
  int koepfe=0;

  for (int i=0; i<wuerfe.size(); i++)
  {
    if (wuerfe[i] == true)
      koepfe++;
  }

  cout << "Wahrscheinlichkeit fuer Kopf: " << koepfe /(float) wuerfe.size() << endl;

  // ein zweiter Test... Wir werfen unserem Generator eine ziemlich
  // unfaire Muenze vor!
  generate(wuerfe.begin(), wuerfe.end(), FaireMuenze(drand48, 0., 0.5));

  koepfe=0;

  for (int i=0; i<wuerfe.size(); i++)
  {
    if (wuerfe[i] == true)
      koepfe++;
  }

  cout << "Wahrscheinlichkeit fuer Kopf: " << koepfe /(float) wuerfe.size() << endl;
```

```
    return 0;
}
```
