Informatik II

Übung 1

Andreas Bärtschi, Andreea Ciuprina, Felix Friedrich, Patrick Gruntz,
Hermann Lehner, Max Rossmannek, Chris Wendler

FS 2018

Heutiges Programm

1. Administratives
2. Wiederholung Theorie
   - Problem und Algorithmus
   - Asymptotische Laufzeit
3. Programmierübung
   - Unfair Würfeln

Unser Angebot

- Bearbeitung der wöchentlichen Übungsserien $\rightarrow$ Bonus von maximal 0.25 Notenpunkten für die Prüfung.
- Bonus proportional zur erreichten Punktzahl von **speziell markierten Bonus-Aufgaben**. Volle Punktzahl $\widehat{=}$ 0.25.
- Für die **Zulassung** zur Bonusaufgabe 1 benötigt man 180 Punkte auf die ersten drei Übungsaufgaben.
- Grundgedanke: Man soll die ersten zwei Übungen ernsthaft angeschaut haben, bevor die Bonusaufgabe gemacht wird.
- Die Bonusaufgabe wird freigeschaltet, sobald Sie die nötigen 180 Punkte haben, nicht jedoch früher als zur dritten Woche.

Warm-up

- Was ist ein Problem?

Warm-up

- Was ist ein Problem?
- Was ist ein Algorithmus?

Warm-up

- Was ist ein Problem?
- Was ist ein Algorithmus?
  - ➜ wohldefinierte Berechnungsvorschrift, welche aus Eingabedaten (input) Ausgabedaten (output) berechnet.

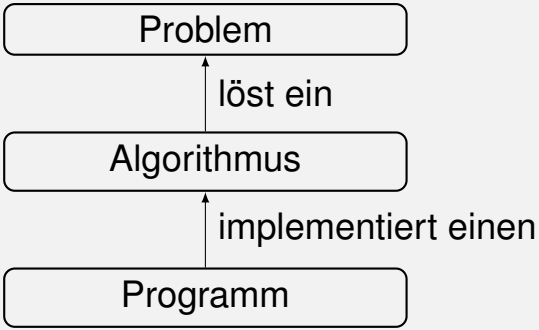
Warm-up

- Was ist ein Problem?
- Was ist ein Algorithmus?
  - ➜ wohldefinierte Berechnungsvorschrift, welche aus Eingabedaten (input) Ausgabedaten (output) berechnet.
- Was ist ein Programm?

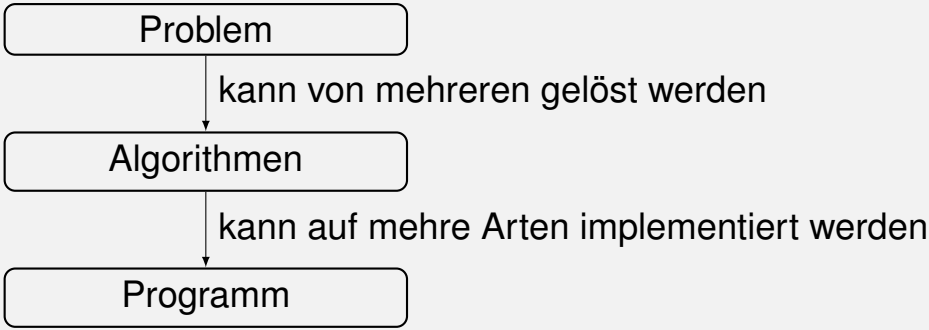
Warm-up

- Was ist ein Problem?
- Was ist ein Algorithmus?
  - ➜ wohldefinierte Berechnungsvorschrift, welche aus Eingabedaten (input) Ausgabedaten (output) berechnet.
- Was ist ein Programm?
  - ➜ Konkrete Implementation eines Algorithmus.

Warm-up


Problem
löst ein
Algorithmus
implementiert einen
Programm

Warm-up


Problem
kann von mehreren gelöst werden
Algorithmen
kann auf mehre Arten implementiert werden
Programm

Effizienz

Problem	Komplexität	Minimale (asymptotische) Kosten über alle Algorithmen, die das Problem lösen.
Algorithmus	Kosten	Anzahl Elementaroperationen
Programm	Laufzeit	Messbarer Wert auf einer konkreten Maschine.

Effizienz

Problem	Komplexität	Minimale (asymptotische) Kosten über alle Algorithmen, die das Problem lösen.
Algorithmus	Kosten	Anzahl Elementaroperationen
Programm	Laufzeit	Messbarer Wert auf einer konkreten Maschine.

➜ Abschätzen von *Kosten* oder *Laufzeit* abhängig von der Eingabegrösse n.

Asymptotisches Verhalten

- Was sind $\Omega(g(n))$, $\Theta(g(n))$, $\mathcal{O}(g(n))$?

Asymptotisches Verhalten

- Was sind $\Omega(g(n))$, $\Theta(g(n))$, $\mathcal{O}(g(n))$?

➜ Mengen von Funktionen!

Asymptotisches Verhalten

- Was sind $\Omega(g(n))$, $\Theta(g(n))$, $\mathcal{O}(g(n))$?

➜ Mengen von Funktionen!

Wiederholung, Mengen A, B:

Teilmenge	$A \subseteq B$
echte Teilmenge	$A \subsetneq B$
Schnittmenge	$A \cap B$

Asymptotisches Verhalten

Gegeben Funktion $f : \mathbb{N} \to \mathbb{R}$.

Definition:

$$\mathcal{O}(g) = \{f : \mathbb{N} \to \mathbb{R} | \exists c > 0, n_0 \in \mathbb{N} : 0 \leq f(n) \leq c \cdot g(n) \; \forall n \geq n_0\}$$
$$\Omega(g) = \{f : \mathbb{N} \to \mathbb{R} | \exists c > 0, n_0 \in \mathbb{N} : 0 \leq c \cdot g(n) \leq f(n) \; \forall n \geq n_0\}$$
$$\Theta(g) = \mathcal{O}(g) \cap \Omega(g)$$

Nützliches für Aufgabenblatt

Theorem

1. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f \in \mathcal{O}(g), \mathcal{O}(f) \subsetneq \mathcal{O}(g)$.
2. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = C > 0$ *(C konstant)* $\Rightarrow f \in \Theta(g)$.
3. $\frac{f(n)}{g(n)} \underset{n\to\infty}{\to} \infty \Rightarrow g \in \mathcal{O}(f), \mathcal{O}(g) \subsetneq \mathcal{O}(f)$.

Nützliches für Aufgabenblatt

Theorem

1. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f \in \mathcal{O}(g), \mathcal{O}(f) \subsetneq \mathcal{O}(g)$.
2. $\lim_{n\to\infty} \frac{f(n)}{g(n)} = C > 0$ *(C konstant)* $\Rightarrow f \in \Theta(g)$.
3. $\frac{f(n)}{g(n)} \underset{n\to\infty}{\to} \infty \Rightarrow g \in \mathcal{O}(f), \mathcal{O}(g) \subsetneq \mathcal{O}(f)$.

Beispiel

1. $\lim_{n\to\infty} \frac{n}{n^2} = 0 \Rightarrow n \in \mathcal{O}(n^2), \mathcal{O}(n) \subsetneq \mathcal{O}(n^2)$.
2. $\lim_{n\to\infty} \frac{2n}{n} = 2 > 0 \Rightarrow 2n \in \Theta(n)$.
3. $\frac{n^2}{n} \underset{n\to\infty}{\to} \infty \Rightarrow n \in \mathcal{O}(n^2), \mathcal{O}(n) \subsetneq \mathcal{O}(n^2)$.

Quiz

$1 \in \mathcal{O}(15)$?

Quiz

$1 \in \mathcal{O}(15)$? ✓ besser $1 \in \mathcal{O}(1)$

Quiz

$1 \in \mathcal{O}(15)$?

✓ besser $1 \in \mathcal{O}(1)$

$2n + 1 \in \Theta(n)$?

Quiz

$1 \in \mathcal{O}(15)$? ✓ besser $1 \in \mathcal{O}(1)$

$2n + 1 \in \Theta(n)$? ✓

Quiz

$1 \in \mathcal{O}(15)$?	✓ besser $1 \in \mathcal{O}(1)$
$2n + 1 \in \Theta(n)$?	✓
$\sqrt{n} \in \mathcal{O}(n)$?	

Quiz

$1 \in \mathcal{O}(15)$?	✓ besser $1 \in \mathcal{O}(1)$
$2n + 1 \in \Theta(n)$?	✓
$\sqrt{n} \in \mathcal{O}(n)$?	✓

Quiz

$1 \in \mathcal{O}(15)$?	✓ besser $1 \in \mathcal{O}(1)$
$2n + 1 \in \Theta(n)$?	✓
$\sqrt{n} \in \mathcal{O}(n)$?	✓
$\sqrt{n} \in \Omega(n)$?	

Quiz

$1 \in \mathcal{O}(15)$?	✓ besser $1 \in \mathcal{O}(1)$
$2n + 1 \in \Theta(n)$?	✓
$\sqrt{n} \in \mathcal{O}(n)$?	✓
$\sqrt{n} \in \Omega(n)$?	✗

Quiz

$1 \in \mathcal{O}(15)$?	✓ besser $1 \in \mathcal{O}(1)$
$2n + 1 \in \Theta(n)$?	✓
$\sqrt{n} \in \mathcal{O}(n)$?	✓
$\sqrt{n} \in \Omega(n)$?	✗
$n \in \Omega(\sqrt{n})$?	

Quiz

$1 \in \mathcal{O}(15)$?	✓ besser $1 \in \mathcal{O}(1)$
$2n + 1 \in \Theta(n)$?	✓
$\sqrt{n} \in \mathcal{O}(n)$?	✓
$\sqrt{n} \in \Omega(n)$?	✗
$n \in \Omega(\sqrt{n})$?	✓

Quiz

$1 \in \mathcal{O}(15)$?	✓ besser $1 \in \mathcal{O}(1)$
$2n + 1 \in \Theta(n)$?	✓
$\sqrt{n} \in \mathcal{O}(n)$?	✓
$\sqrt{n} \in \Omega(n)$?	✗
$n \in \Omega(\sqrt{n})$?	✓
$\sqrt{n} \notin \Theta(n)$?	

Quiz

$1 \in \mathcal{O}(15)$?	✓ besser $1 \in \mathcal{O}(1)$
$2n + 1 \in \Theta(n)$?	✓
$\sqrt{n} \in \mathcal{O}(n)$?	✓
$\sqrt{n} \in \Omega(n)$?	✗
$n \in \Omega(\sqrt{n})$?	✓
$\sqrt{n} \notin \Theta(n)$?	✓

Quiz

$1 \in \mathcal{O}(15)$?	✓ besser $1 \in \mathcal{O}(1)$
$2n + 1 \in \Theta(n)$?	✓
$\sqrt{n} \in \mathcal{O}(n)$?	✓
$\sqrt{n} \in \Omega(n)$?	✗
$n \in \Omega(\sqrt{n})$?	✓
$\sqrt{n} \notin \Theta(n)$?	✓
$\mathcal{O}(\sqrt{n}) \subset \mathcal{O}(n)$?	

Quiz

$1 \in \mathcal{O}(15)$?	✓ besser $1 \in \mathcal{O}(1)$
$2n + 1 \in \Theta(n)$?	✓
$\sqrt{n} \in \mathcal{O}(n)$?	✓
$\sqrt{n} \in \Omega(n)$?	✗
$n \in \Omega(\sqrt{n})$?	✓
$\sqrt{n} \notin \Theta(n)$?	✓
$\mathcal{O}(\sqrt{n}) \subset \mathcal{O}(n)$?	✓

Quiz

$1 \in \mathcal{O}(15)$?	✓ besser $1 \in \mathcal{O}(1)$
$2n + 1 \in \Theta(n)$?	✓
$\sqrt{n} \in \mathcal{O}(n)$?	✓
$\sqrt{n} \in \Omega(n)$?	✗
$n \in \Omega(\sqrt{n})$?	✓
$\sqrt{n} \notin \Theta(n)$?	✓
$\mathcal{O}(\sqrt{n}) \subset \mathcal{O}(n)$?	✓
$2^n \notin \mathcal{O}(\exp(n))$?	

Quiz

$1 \in \mathcal{O}(15)$?	✓ besser $1 \in \mathcal{O}(1)$
$2n + 1 \in \Theta(n)$?	✓
$\sqrt{n} \in \mathcal{O}(n)$?	✓
$\sqrt{n} \in \Omega(n)$?	✗
$n \in \Omega(\sqrt{n})$?	✓
$\sqrt{n} \notin \Theta(n)$?	✓
$\mathcal{O}(\sqrt{n}) \subset \mathcal{O}(n)$?	✓
$2^n \notin \mathcal{O}(\exp(n))$?	✗

Quiz

$1 \in \mathcal{O}(15)$?	✓ besser $1 \in \mathcal{O}(1)$
$2n + 1 \in \Theta(n)$?	✓
$\sqrt{n} \in \mathcal{O}(n)$?	✓
$\sqrt{n} \in \Omega(n)$?	✗
$n \in \Omega(\sqrt{n})$?	✓
$\sqrt{n} \notin \Theta(n)$?	✓
$\mathcal{O}(\sqrt{n}) \subset \mathcal{O}(n)$?	✓
$2^n \notin \mathcal{O}(\exp(n))$?	✗

Zeitbedarf

Annahme: 1 Operation = $1\mu s$.

Problemgrösse	1	100	10000	10^6	10^9
$\log_2 n$	$1\mu s$				
n	$1\mu s$				
$n \log_2 n$	$1\mu s$				
n^2	$1\mu s$				
2^n	$1\mu s$				

Zeitbedarf

Annahme: 1 Operation = $1\mu s$.

Problemgrösse	1	100	10000	10^6	10^9
$\log_2 n$	$1\mu s$	$7\mu s$	$13\mu s$	$20\mu s$	$30\mu s$
n	$1\mu s$				
$n \log_2 n$	$1\mu s$				
n^2	$1\mu s$				
2^n	$1\mu s$				

Zeitbedarf

Annahme: 1 Operation = $1\mu s$.

Problemgrösse	1	100	10000	10^6	10^9
$\log_2 n$	$1\mu s$	$7\mu s$	$13\mu s$	$20\mu s$	$30\mu s$
n	$1\mu s$	$100\mu s$	$1/100s$	$1s$	17 Minuten
$n \log_2 n$	$1\mu s$				
n^2	$1\mu s$				
2^n	$1\mu s$				

Zeitbedarf

Annahme: 1 Operation = $1\mu s$.

Problemgrösse	1	100	10000	10^6	10^9
$\log_2 n$	$1\mu s$	$7\mu s$	$13\mu s$	$20\mu s$	$30\mu s$
n	$1\mu s$	$100\mu s$	$1/100s$	$1s$	17 Minuten
$n \log_2 n$	$1\mu s$	$700\mu s$	$13/100\mu s$	$20s$	8.5 Stunden
n^2	$1\mu s$				
2^n	$1\mu s$				

Zeitbedarf

Annahme: 1 Operation = $1\mu s$.

Problemgrösse	1	100	10000	10^6	10^9
$\log_2 n$	$1\mu s$	$7\mu s$	$13\mu s$	$20\mu s$	$30\mu s$
n	$1\mu s$	$100\mu s$	$1/100s$	$1s$	17 Minuten
$n \log_2 n$	$1\mu s$	$700\mu s$	$13/100\mu s$	$20s$	8.5 Stunden
n^2	$1\mu s$	$1/100s$	1.7 Minuten	11.5 Tage	317 Jahrhund.
2^n	$1\mu s$				

Zeitbedarf

Annahme: 1 Operation = $1\mu s$.

Problemgrösse	1	100	10000	10^6	10^9
$\log_2 n$	$1\mu s$	$7\mu s$	$13\mu s$	$20\mu s$	$30\mu s$
n	$1\mu s$	$100\mu s$	$1/100s$	$1s$	17 Minuten
$n \log_2 n$	$1\mu s$	$700\mu s$	$13/100\mu s$	$20s$	8.5 Stunden
n^2	$1\mu s$	$1/100s$	1.7 Minuten	11.5 Tage	317 Jahrhund.
2^n	$1\mu s$	10^{14} Jahrh.	$\approx \infty$	$\approx \infty$	$\approx \infty$

Eine gute Strategie?

... dann kaufe ich mir eben eine neue Maschine!

Eine gute Strategie?

... dann kaufe ich mir eben eine neue Maschine! Wenn ich heute ein Problem der Grösse n lösen kann, dann kann ich mit einer 10 oder 100 mal so schnellen Maschine...

Komplexität	(speed $\times 10$)	(speed $\times 100$)
$\log_2 n$		
n		
n^2		
2^n		

Eine gute Strategie?

... dann kaufe ich mir eben eine neue Maschine! Wenn ich heute ein Problem der Grösse n lösen kann, dann kann ich mit einer 10 oder 100 mal so schnellen Maschine...

Komplexität	(speed $\times 10$)	(speed $\times 100$)
$\log_2 n$	$n \to n^{10}$	$n \to n^{100}$
n		
n^2		
2^n		

Eine gute Strategie?

... dann kaufe ich mir eben eine neue Maschine! Wenn ich heute ein Problem der Grösse n lösen kann, dann kann ich mit einer 10 oder 100 mal so schnellen Maschine...

Komplexität	(speed $\times 10$)	(speed $\times 100$)
$\log_2 n$	$n \to n^{10}$	$n \to n^{100}$
n	$n \to 10 \cdot n$	$n \to 100 \cdot n$
n^2		
2^n		

Eine gute Strategie?

... dann kaufe ich mir eben eine neue Maschine! Wenn ich heute ein Problem der Grösse n lösen kann, dann kann ich mit einer 10 oder 100 mal so schnellen Maschine...

Komplexität	(speed $\times 10$)	(speed $\times 100$)
$\log_2 n$	$n \to n^{10}$	$n \to n^{100}$
n	$n \to 10 \cdot n$	$n \to 100 \cdot n$
n^2	$n \to 3.16 \cdot n$	$n \to 10 \cdot n$
2^n		

Eine gute Strategie?

... dann kaufe ich mir eben eine neue Maschine! Wenn ich heute ein Problem der Grösse n lösen kann, dann kann ich mit einer 10 oder 100 mal so schnellen Maschine...

Komplexität	(speed $\times 10$)	(speed $\times 100$)
$\log_2 n$	$n \to n^{10}$	$n \to n^{100}$
n	$n \to 10 \cdot n$	$n \to 100 \cdot n$
n^2	$n \to 3.16 \cdot n$	$n \to 10 \cdot n$
2^n	$n \to n + 3.32$	$n \to n + 6.64$

Asymptotische Laufzeiten mit Θ

```
void run(int n){
  for (int i = 1; i<n; ++i)
    for (int j = 1; j<n; ++j)
      op();
}
```

Wie oft wird `op()` aufgerufen?

Asymptotische Laufzeiten mit mit $\ominus$

```
void run(int n){
  for (int i = 1; i<n; ++i)
    for (int j = i; j<n; ++j)
      op();
}
```

Wie oft wird `op()` aufgerufen?

Asymptotische Laufzeiten mit Θ

```
void run(int n){
  for (int i = 1; i<n; ++i){
    op();
    for (int j = i; j<n; ++j)
      op();
    }
}
```

Wie oft wird `op()` aufgerufen?

Asymptotische Laufzeiten mit Θ

```
void run(int n){
  for(int i = 1; i <= n; ++i)
    for(int j = 1; j*j <= n; ++j)
      for(int k = n; k >= 2; --k)
        op();
}
```

Wie oft wird `op()` aufgerufen?

3. Programmierübung

Unfairer Würfel

Würfel simulieren

Gegeben: Simulation uniformverteilter Zufallsvariable `Math.Random()` $\in [0, 1)$.

Gesucht: Simulation eines *fairen* Würfels

Würfel simulieren

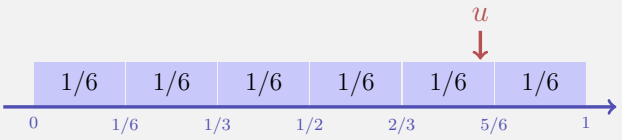
`Math.Random()` gibt ein $U \in [0, 1)$ zurück mit

$$\mathbb{P}(U \in [l, r)) = r - l.$$

`Dice()` soll ein $Y \in \{1, \ldots, 6\}$ zurückgeben, so dass

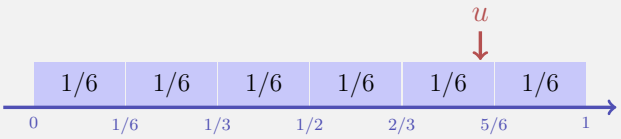
$\mathbb{P}(Y = k) = 1/6$ für jedes $k \in \{1, \ldots, 6\}$

Umständlich, aber korrekt




```
static int Dice(){
  double u = Math.random();
  if (u<1.0/6) return 1;
  else if (u<1.0/3) return 2;
  else if (u<1.0/2) return 3;
  else if (u<2.0/3) return 4;
  else if (u<5.0/6) return 5;
  else return 6;
}
```

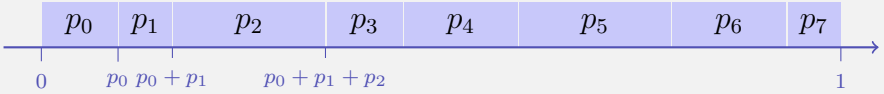
Einfacher




```
static int Dice(){
  double u = Math.random();
  return (int)(u*6+1);
}
```

Unfair Würfeln

Gegeben: Wahrscheinlichkeitsvektor $p = (p_0, \ldots, p_{n-1})$ mit $\sum_{i=0}^{n-1} p_i = 1$ und $p_i \geq 0$ $(0 \leq i < n)$.

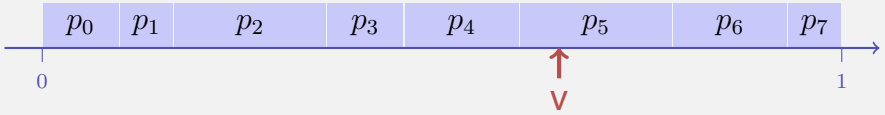


Gesucht: `Sample(p)` soll ein j $(0 \leq j < n)$ zurückgeben mit Wahrscheinlichkeit p_j.

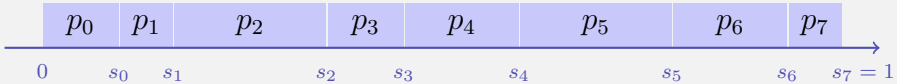
Formal

Ziehe mit einem vorhandenen Zufallszahlengenerator eine Zufallszahl v gleichverteilt aus dem Intervall $[0, 1)$. Aus dieser Zahl v erzeugt man dann eine ganze Zahl $0 \leq S(p, v) < n$ mit folgender Regel

$$S(p, v) = \min\{0 \leq i < n : \sum_{k=0}^{i} p_k > v\}$$



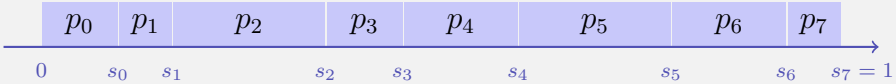
Unfair Würfeln




```
static int Sample(double[] p){
  double u = Math.random();
  if (u<p[0]) return 0;
  if (u<p[0]+p[1]) return 1;
  if (u<p[0]+p[1]+p[2]) return 2;
  if (u<p[0]+p[1]+p[2]+p[3]) return 3;
  ...
}
```

Zu umständlich: wir brauchen eine Schleife!

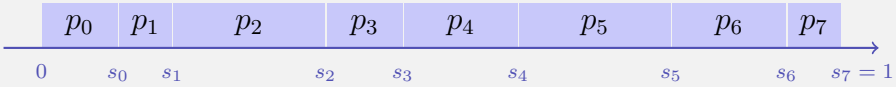
Ein kleiner Trick




```
static int Sample(double[] p){
  double u = Math.random();
  if (u<p[0]) return 0;
  u -= p[0];
  if (u<p[1]) return 1;
  u -= p[1];
  if (u<p[2]) return 2;
  ...
}
```

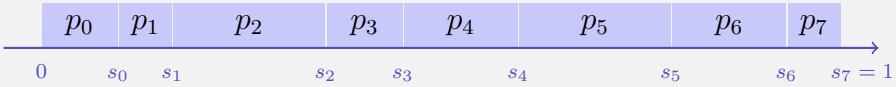
So müssen wir Summe der p_i nicht mitrechnen.

In eine Schleife




```
static int Sample(double[] p){
  double u = Math.random();
  for (int k = 0; k < p.length−1; ++k){
    if (u<p[k]) return k;
    u −= p[k];
  }
  return p.length−1;
}
```

Noch kompakter




```
static int Sample(double[] p){
  double u = Math.random();
  int k=0;
  while (k < p.length && u>0){
    u −= p[k++];
  }
  return k−1;
}
```

Fragen oder Anregungen?