

Prof. Dr. Pascal Laube Fakultät Informatik

8.11.21

Algorithmen und Datenstrukturen Übung 4 (AIN2)

In dieser Übung soll die Laufzeit des **Square-Multiply** Verfahrens zum Potenzieren experimentell untersucht werden.

1. Implementieren Sie zunächst eine Klasse PowerFunctions mit den im folgenden angegebenen statischen Methoden zur Berechnung von $y = x^n$. Später sollen diese Methoden in anderen Klassen verfügbar gemacht werden.

```
public static double power (double x, int n) soll x^n = 1 \cdot x \cdot x \cdot \dots x mit Hilfe einer Schleife berechnen. Dabei werden n Multiplikationen benötigt.
```

public static double fastPower(double x, int n) soll x^n unter Verwendung des **Square-Multiply** Verfahrens berechnen. Dazu sind nur $O(\log n)$ Multiplikationen erforderlich.

2. Ergänzen Sie die Klasse PowerFunctions um zwei private statische Member-Variablen countPower und countFastPower, mit deren Hilfe die Anzahl der von power bzw. fastPower ausgeführten Multiplikationen gezählt werden kann. Prinzipiell könnten Sie dazu int-Variablen benutzen, aber zur Übung sollten Sie Objekte der Klasse Counter aus Übung 1 verwenden.

Erweitern Sie die Berechnungsmethoden, so dass bei jeder Multiplikation der entsprechende Zähler erhöht wird.

3. Implementieren Sie folgenden zusätzlichen Methoden für den externen Zugriff auf die Zähler:

```
public static void resetCounters()
setzt beide Zähler auf 0 zurück.

public static int getCountPower()
liefert den Zählerstand von countPower

public static int getCountFastPower()
liefert den Zählerstand von countFastPower
```

4. Implementieren Sie schließlich eine neue Klasse PowerFunctionsTest mit einem Hauptprogramm, das jeweils 10000 Berechnungen von x^n für zufällige Argumente $x \in [0,1)$ und $n \in [0,1000)$ mit den beiden Berechnungsverfahren durchführt.

Anschließend soll der durchschnittliche verwendete Exponent n und die durchschnittliche Anzahl notwendiger Multiplikationen in power bzw. fastPower ausgegeben werden.

Hinweise: Mit der Anweisung double x=Math.random() können Sie eine reelle Zufallszahl im Intervall [0,1) erzeugen.

Brauchen Sie z.B. einen ganzzahligen Wert aus [0,99] erhalten Sie ihn mit int y = (int)(100*x).