Labor II: Algorithmen und Datenstrukturen

Dokumentation Gruppe 27

# Ergebnissdokumentation

1.

Theoretisch größter Binomialkoeffizient:

**33 über 16 = 1.166.803.110 < 231-1**

**34 über 17 = 2.333.606.220 > 231­­-1**

33 über 16 ist nur dann der größtmögliche Koeffizient wenn die Berechnung für jedes k gültig ist.

K muss n/2 sein um den größtmöglichen Koeffizienten von n zu erhalten.

2.

1. Bei der Berechnung des Binomialkoeffizienten besteht eine Fakultät über n. Zu bestimmen ist nun der maximal Wert für n des Typs Integer. Ein Integer besteht aus 32 Bits. Da es in Java leider keine Möglichkeit eines „unsigned int“ gibt, ist der Wertebereich nicht 232 wie zuerst vermutet sondern nur die Hälfte. Somit ist der positive Wertebereich 0 <= x <= 2.147.483.647. Den maximalen Wertebereich haben wir nun mittels einer Methode fakultaet() durch ausprobieren auf 12 eingrenzen können. Bei !13 ist das Ergebnis aufgrund eines Überlaufs nicht mehr korrekt.

|  |  |  |
| --- | --- | --- |
| Fakultät !n | Fakultaet(long n) | Fakultaet(int n) |
| !0 | 1 | 1 |
| !1 | 1 | 1 |
| !2 | 2 | 2 |
| !3 | 6 | 6 |
| !4 | 24 | 24 |
| !5 | 120 | 120 |
| !6 | 720 | 720 |
| !7 | 5040 | 5040 |
| !8 | 40320 | 40320 |
| !9 | 362880 | 362880 |
| !10 | 3628800 | 3628800 |
| !11 | 39916800 | 39916800 |
| !12 | 479001600 | 479001600 |
| !13 | 6227020800 | 1932053504 |
| !14 | 87178291200 | 1278945280 |

Bei der Iterativen Berechnung ist es wichtig zwischen Worst Case und Best Case zu unterscheiden. Die Methode iterativ() entspricht dem Worst Case da nicht gekürzt wird.

Der Worst Case in Abhängigkeit von k wäre k = n. Somit wäre der Wertebereich wieder bei 12.

Mithilfe der Verbesserung der Methode durch Kürzen ist es möglich den maximalen Binomialkoeffizienten (Aufgabe a) ) zu berechnen, d.h. n = 33.

Bei der Rekursiven Variante ist der Wertebereich ebenfalls n = 33.

1. Um den Wertebereich zu vergrößern ist es nötig den Wertebereich von Nenner und Zähler zu minimieren. Wir erreichen dies durch Kürzen mit dem ggT. Der ggT lässt sich am einfachsten durch den Euklidschen Algorithmus finden. Die Multiplikationen im Nenner und Zähler wird vor dem ausmultiplizieren in einem Array gespeichert und mit der Mehtode kürzen(), gekürzt. Nach dem Kürzen werden die Werte in den Arrays ausmultipliziert.
2. Komplexitäten:

Berechnung über Fakultäten:

Θ **(n)**

Berechnung Iterative Berechnung:

Θ(1), falls k = n

Θ(1), falls k = 0

sonst Θ(k)

Rekursive Berechnung:

Abhängig vom Binomialkoeffizienten und daher nicht vorhersagbar.

Θ(2n)