# Dokumentation zum Aufgabenblatt a05

Implementiert werden sollten Algorithmen, die die n-te Zeile des Pascalschen Dreiecks berechnen und ausgeben sollen. Gefordert war ein iterativer, ein rekursiver, sowie ein möglichst schneller Algorithmus. Weiterhin sollten Zähler eingebaut werden, um den Aufwand zu messen.

Beim Implementieren der Algorithmen stellten sich Probleme durch die Grenzen der genutzten Datentypen dar. Um möglichst viele Zeilen darstellen zu können, ohne dass es zu einem Überlauf des Zahlenbereichs kommt, wurde der Datentyp long verwendet. Auch in der Klasse für die Berechnung der Fakultät gibt es verschiedene Implementierungen. Mittels des Datentyps long kann die Fakultät bis maximal 20! berechnet werden. Um Weitaus höhere Fakultäten zu berechnen, wurde eine Methode geschrieben, die intern mit BigInteger rechnet. Allerdings geht diese Variante zu Lasten der Laufzeit.

Implementiert wurde ein iterativer Algorithmus, dieser berechnet das Dreieck bis zur maximal 66. Zeile.
Eine weitere iterative Methode wurde implementiert, die statt mit einem gängigen 2D-Array mit zwei 1D-Arrays arbeitet, in denen sich jeweils nur die aktuelle und die letzte Zeile gemerkt werden. Diese Methode verbraucht wesentlich weniger Speicher (embedded geeignet), als die analoge Implementierung. Allerdings vergeht mehr Laufzeit, da die Arrays pro Iteration verworfen und neu angelegt werden müssen.

Der rekursive Algorithmus kann Zeilen bis n = 33 berechnen, was allerdings extrem lange dauert.
Um dem Benutzer die Übergabe einer Wunschzeile (n) zu ermöglichen, wurde die rekursive Methode überladen. In der Methode mit dem Übergabeparameter n läuft dann eine for-Schleife, die die rekursive Methode mit entsprechenden Zeilen und Spaltenparametern aufruft.

Für die schnelle Berechnung (Aufgabe 3) wurden zwei Methoden mit Binomialkoeffizienten implementiert. Aufgrund der Zahlenbereichsbeschränkung des Datentyps long kann die erste Binomial-Methode nur bis n = 20 rechnen, für größere n erfolgt bei der Fakultätsberechnung ein Überlauf des Zahlenbereichs. Um diesem Problem vorzubeugen wurde eine weitere Methode implementiert, die mit der BigInteger Implementierung der Fakultätsberechnung arbeitet. Diese Methode kann bis max. n = 136 rechnen, ist allerdings bei der Zeitmessung langsamer.
Weiterhin wurde eine Formel implementiert, die die n-te Zeile mit einem Aufwand von O(N) berechnen kann. Die Formel berechnet die Zeile bis max. n = 60.

Die geforderte Auswertung und Darstellung der Ergebnisse in Form einer Tabelle und Diagrammen befindet sich im Anschluss an diese Dokumentation.

**Martin Slowikowski**
Matrikelnummer: 199 91 66

**Tell Mueller-Pettenpohl**
Matrikelnummer: 198 99 82

**Teamname**: Tugend und Laster

**Quellenangaben**

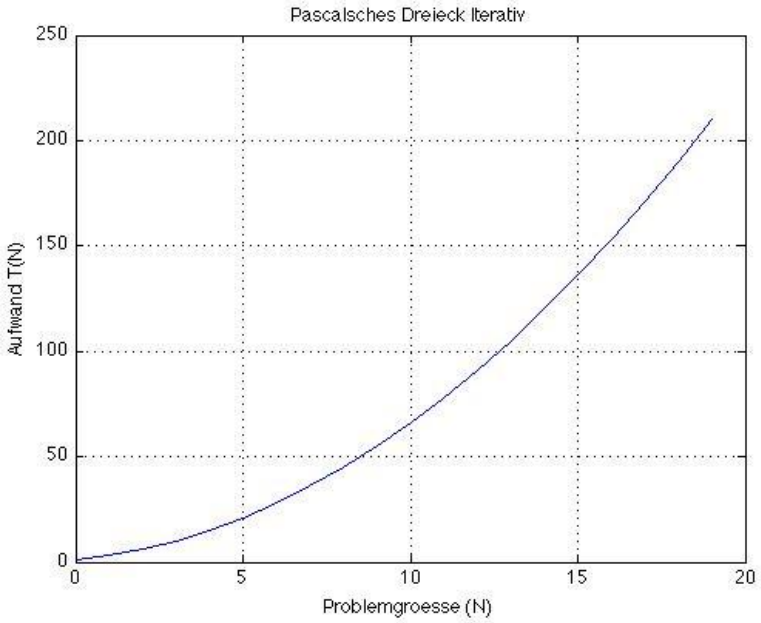
TIB3-AD-skript.pdf
Letzte Änderung 22.03.2011

Das Pascalsche Dreieck
Michael Holzapfel
http://www.michael-holzapfel.de/themen/pascaldreieck/pascaldreieck.htm
abgerufen am 30.04.2011

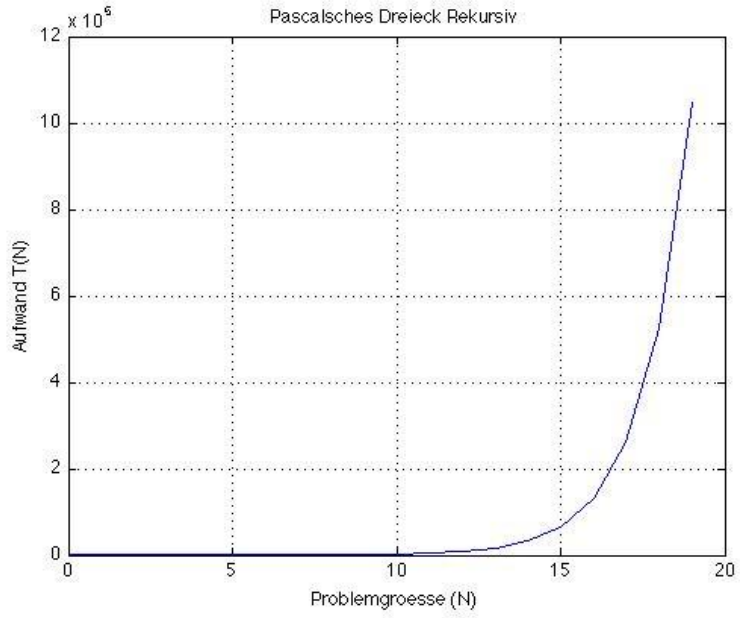
Pascal's Triangle
Ynnorj
http://www.dreamincode.net/forums/topic/23942-pascals-triangle-so-close-need-help-finishing/page__view__findpost__p__847607
27.11.2009
abgerufen 30.04.2011

## Auswertung und Darstellung

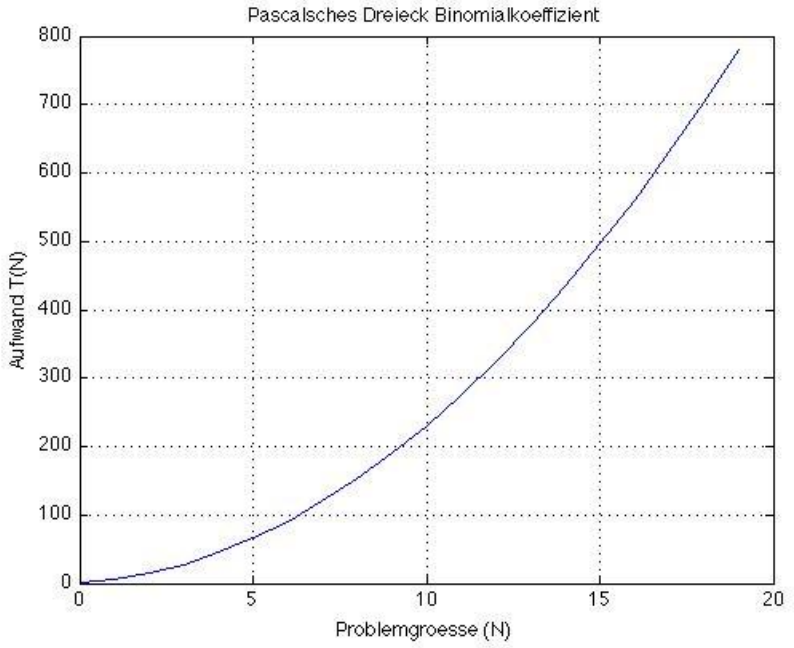
Es folgen die grafischen Auswertungen der eingebauten Zähler der einzelnen Algorithmen.



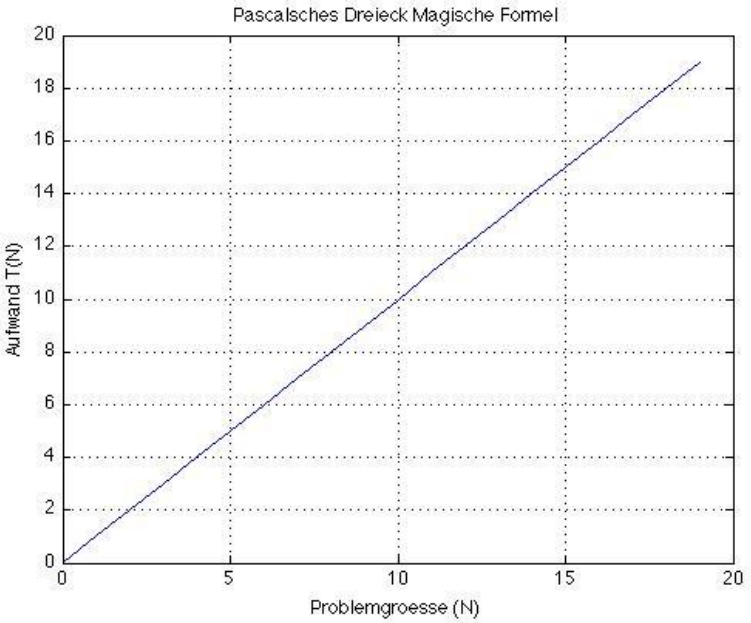

Bei der iterativen Berechnung der n-ten Zeile des Pascalschen Dreiecks ist ein quadratischer Aufwand zu erkennen, der sich aus den zwei geschachtelten for-Schleifen ergibt, Aufwand $O(N^2)$.
In der Methode wird über die beiden geschachtelten for-Schleifen das komplette Dreieck in einem 2D-Array gespeichert und zurückgegeben. Über eine Ausgabemethode wird dann nur die letzte Zeile ausgegeben. Diese Implementierung ist bereits im direkten Vergleich (Laufzeit) mit den anderen Methoden sehr schnell bei der Berechnung der n-ten Zeile.



Bei der rekursiven Berechnung der n-ten Zeile des Pascalschen Dreiecks ist ein exponentieller Aufwand erkennbar, Aufwand $O(2^n)$.
In der Methode, in der das zu berechnende n übergeben wird, wird in einer for-Schleife jede Position der gesuchten Zeile berechnet. Die Berechnung erfolgt durch den Aufruf der überladenen Methode. An diese wird die Zeile und die Position übergeben. Hier findet dann die Rekursion rückwärts bis zur ersten Zeile statt. Die extrem lange Dauer der Berechnung folgt aus dem return-statement, in dem die Rekursion aufgerufen wird. Hier wird die Rekursion zwei Mal aufgerufen für die Addition der beiden Elemente aus n-1 Zeile.




Bei der Berechnung der n-ten Zeile des Pascalschen Dreiecks mittels Binomialkoeffizienten ist wieder ein quadratischer Aufwand zu erkennen, $O(N^2)$.
In der äußeren for-Schleife wird die Methode zur Berechnung der entsprechenden Binomialkoeffizienten für die n-te Zeile aufgerufen. Hier wird dann der entsprechende Wert nach der Formel $\frac{n!}{k!(n-k)!}$ ausgerechnet. Die Fakultäten werden mittels einer for-Schleife jeweils berechnet.
Damit ergibt sich eine zweifach geschachtelte for-Schleife, also quadratischer Aufwand, da die for-Schleifen für die Fakultät nicht weiter geschachtelt sind, sondern nach und nach abgearbeitet werden. Dieser Algorithmus arbeitet bei der Zeitmessung schneller als die iterative Berechnung, obwohl mehr Operationen anfallen.




Bei der Berechnung der n-ten Zeile mit der „magischen Formel“ ist ein komplett linearer Aufwand erkennbar, O(N).
Im inneren der Methoden gibt es lediglich eine for-Schleife, die die einzelnen Werte der n-ten Zeile berechnet.
Wie genau die Formel arbeitet kann hoffentlich im Praktikum geklärt werden.