Martin Witte
Karl-Fabian Witte

21. Februar 2023

Abstract

Die Berechnung des Pascal'sche Dreieck soll in drei verschiedenen Algorithmen realisiert werden. Die Komplexität der Implementationen soll mittels einer Messung bestimmt werden. Die Algorithmen wurden rekursiv, iterativ und über den Binominalkoeffizienten für natürliche Zahlen realisiert.

Methoden

1. Rekursive-Methode

In der rekursiven Methode wird die N'te Zeile des Pascal'schen Dreiecks von der 0'ten Zeile an, Zeile für Zeile, durch Wiederaufrufen der gleichen Funktion errechnet.

```
void P(int N){
        for (int k = N; k >= 0; k--){    // N + 1 schritte
                recurP(N, k);
        }
}

int recurP(int N, int k){
        // Aufwand++
        if (N <= 1 || k==0 || k==N){
                        return 1;
                } else {
                        return recurP(N - 1, k - 1) + recurP(N - 1, k);
                }
}
```

$T_P(N) = T_{recurP}(N+1, N)$
$T_{recurP}(N,k) = T_{recurP}(N-1,k-1) + T_{recurP}(N-1,k))$
$= 2(T_{recurP}(N-1,k)) - 1$
$= 2^N - N$
$\rightarrow T_P(N) = 2^{N+1} - (N+1) = \mathcal{O}(2^N)$

2. Iterative-Methode

In der iterativen Methode wird die N'te Zeile des Pascal'schen Dreiecks von der 0'ten Zeile an, Zeile für Zeile, durch N-fachen Schleifenaufruf berechnet.

```
void P(int N){
        int[][] arr = new arr[N+1][N+1];
        arr[0][0] = 1;                                    // Aufwand++
        for (int i = 1; i <= N; i++) {
                for (int k = 0; k <= i; k++) {            // Aufwand++
                        if (k == 0)
                                arr[n][k] = 1;
                        else
                                arr[n][k] = arr[N -1][k - 1] + arr[N-1][k];
      }
    }
}
```

$T(N) = N * k(N) + 1 = \sum_{i=0}^{N}(i+1) + 1 = \frac{(N+2)(N+1)}{2} + 1 = \frac{N^2+3N+2}{2} = \mathcal{O}(n^2)$

3. Binominalkoeffizient-Methode

In der Binominalkoeffizient-Methode hier, FastPascal genannt, wird die N'te Zeile des Pascal'schen Dreiecks, direkt, mit Hilfe der Formel $\binom{N}{k} = \frac{N!}{k!*(N-k)!}$ für jede 'Spalte' der Zeile berechnet. «««< HEAD
Aufwand: $(\sum_{k=1}^{N-1}) + (\sum_{k=1}^{N-1}) = 2N$
Da jedoch auch alle Spalten berechnet werden müssen, wird der Aufwand wie folgt berechnet:
$T(N) = N * 2N = 2N^2 = \mathcal{O}(N^2)$

Es wurde jedoch auf einen noch angeblich schnelleren Algorithmus zurückgegriffen, welcher nur funtioniert, wenn N und k Elemente der Natürlichen zahlen sind. (Es ist dabei zu achten, dass die Berechnung in den Rationalen Zahlenbereich gehen kann.)

$\binom{N}{k} = \prod_{i=1}^{k} = \frac{N-k+i}{i}$

Die Implementation nutzt zudem die Symetrie des Dreieckes aus, sodass die Schleife über die Spalten nur bis höchstens $k = \frac{N}{2}$ (zur Mitte) läuft.

$\binom{N}{k} = \binom{N}{N-k}$

```
for(int k = 0; k <= N; k++){
        int res = 1; // Aufwand++
            int K = (2*k > N) ? N-k : k;

            for (int i=1;i <= K; i++){
              res *= (N-K+i);    // Aufwand++
              res /= i;
            }
            return res;
```

Die Aufwandsherleitung wird hier verwinfacht durch Abschätzung von $K(N) \leq N$:
$T(N) = (N+1)(1+K(N)) = (N+1)+2*\sum_{i=1}^{(N+1)/2}(i+1) = (N+1)+(\frac{N+1}{2}+2)*(\frac{N+1}{2}+1) = \frac{N^2}{2}+\cdots = \mathcal{O}(N^2)$

Die Messung

Auf Grund der Stackbelastung seitens der Recursion von der ersten Implementierung, wird mit dieser nur bis zu einem N = 32 gemessen. Das Diagramm **??** zeigt deutlich das exponentielle ansteigen der Aufwandsfunktion der Rekursion.

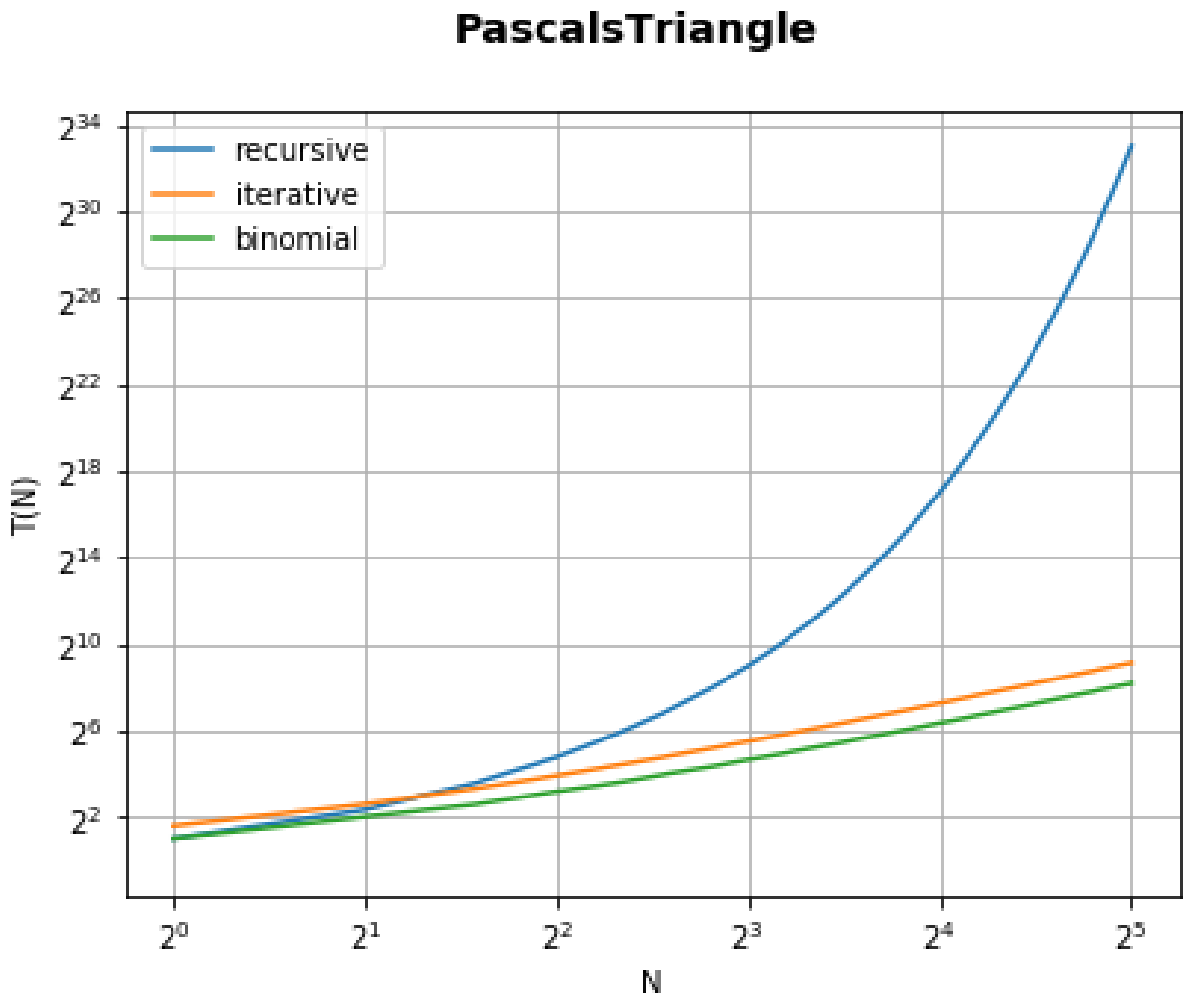

Abbildung 1: Die Aufwende der Implementationen des rekursiven, iterativen und die über den Binomialkoeffizienten Algorithmen sind gegen die Dreieckstiefe N aufgetragen.

In Abbildung **??** sind nur die iteration und der schnelle Algorithmus auf Basis des Binominal Koeffizienten. Beide Algorithmen haben die selbe Komplexität, jedoch hat die binominale Version allgemein einen kleineren Aufwand. Die Komplexität von $\mathcal{O}(N^2)$ kann man mit der Steigungsdreieckmethode im loglog-Diagramm nachzählen.

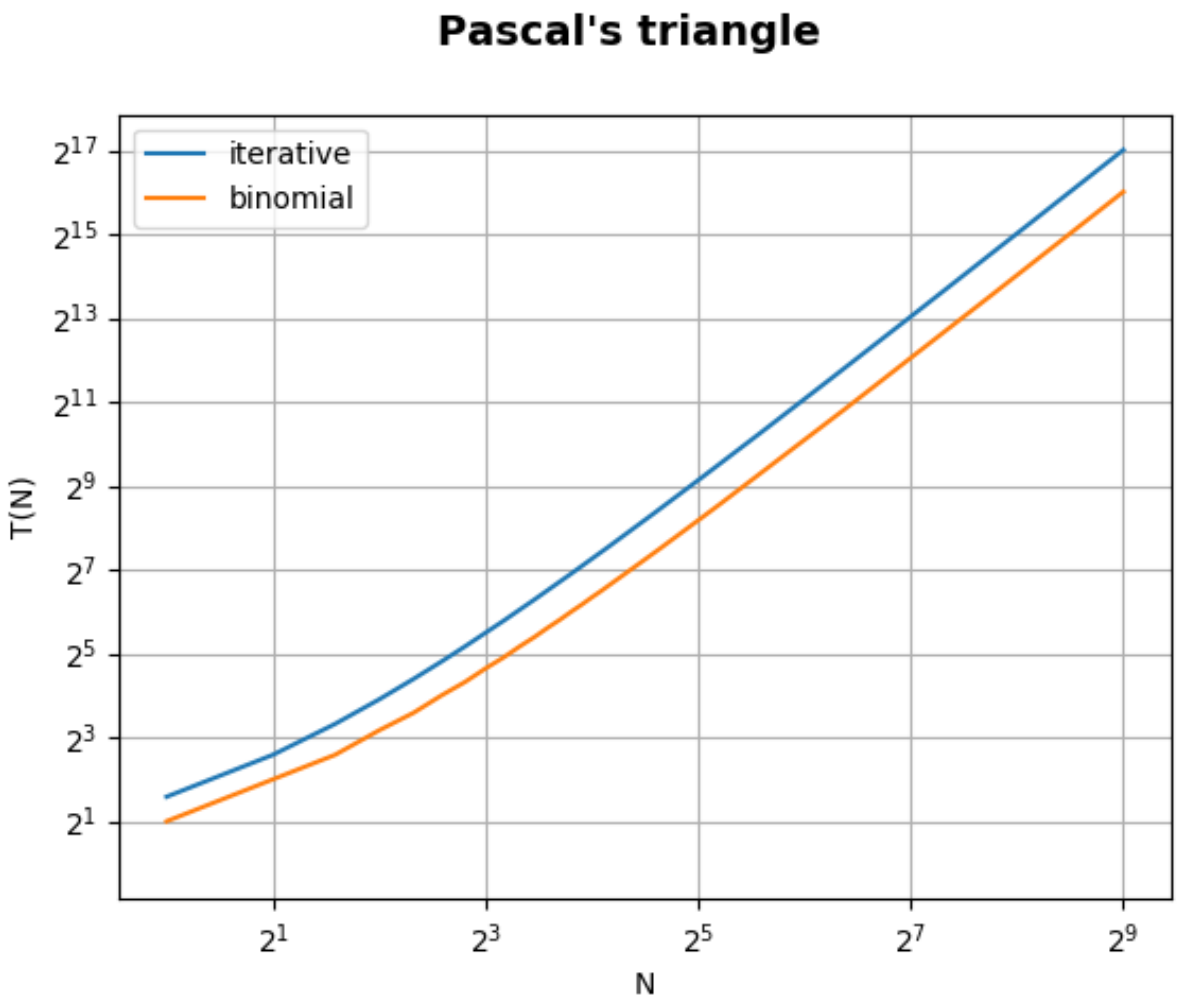

Abbildung 2: Die Aufwende des iterativen und die angeblich schnellere Implementation über den Binomialkooeffizienten sind gegen die Dreieckstiefe N aufgetragen.