## A.6:

1. Implementieren Sie in PYTHON eine Funktion , die eine positive Zahl in Binärdarstellung bitweise um 1 erhöht. Dabei wird b mit k ≥ 1 und bi ∈ {0, 1} als Binärdarstellung der Zahl aufgefasst.

Ein Bild, das Text, Monitor, Bildschirm, festlegen enthält.

Automatisch generierte Beschreibung

2. Führen Sie eine vereinfachte Laufzeitanalyse durch. Was kostet ein einzelner Aufruf im schlechtesten Fall, und was kosten n ≥ 1 Aufrufe von ?

Im schlechtesten Fall wird die for-Schleife m-mal durchlaufen, mit :

Für den n-fachen Aufruf gilt:

3. Führen Sie eine amortisierte Laufzeitanalyse durch, indem Sie n ≥ 1 Aufrufe von gemeinsam betrachten. Sie können vereinfachend annehmen, dass n eine 2er-Potenz ist. Bei wie vielen dieser n Aufrufe wird betrachtet, und wie ist es für usw.? Fassen Sie die Ergebnisse in einer Summe zusammen.

Bei -Aufrufen wird

* -mal überprüft
* -mal überprüft
* -mal überprüft
* -mal überprüft
* ….

Anzahl der Aufrufe als Summe mit :

Aufrufe 🡪 Zahlenwert in 🡪 viele Bits betrachtet =

Ergebnis:

* Man kommt nicht zum gleichen Ergebnis bei amortisiert und vereinfachter Analyse
* Amortisiert wir schauen uns den Durchschnitt an 🡪 Bei -Aufrufen haben wir Aufwand von
* Bei vereinfachter Analyse kommt Aufwand von raus

🡪 Amortisiert ist hier genauer und besser