Namen: Christian Gurski [4067886], Florian Ryll [4068296]

P1L2A03

Zum Abschätzen der Laufzeit nutze ich die O-Notationen, gebe also die Laufzeitkomplexität an:

Algorithmus A hat aufgrund der von n abhängigen Schleife eine O-Notation von O(n):

f(n)= O(n) + O(1) = **O(n)**, da O(n) dominiert.

Algorithmus B hat keine Schleifen, sondern nur einmal ausgeführte vom Parameter i abhängige Methodenaufrufe und somit eine O-Notation von O(1):

f(i)=2* O(1) + 2 * O(1) = **O(1)** , da O(1) dominiert.

Algorithmus C hat eine Dreifachschleife, wobei nur bei 2 Schleifen von n abhängig sind und die übrig bleibende Schleife eine konstante Anzahl von 10 Durchläufen hat. Somit ergibt das eine O-Notation von $O(n^2)$:

f(n)=O(1) + O(1) + O(n) * O(10) * O(n+1) + O(1) = **$O(n^2)$**, da $O(n^2)$ dominiert.

Algorithmus D hat eine Doppelschleife und eine einfache Schleife. Weil die Doppelschleife dominiert, ergibt sich eine O-Notation von $O(n^2)$:

f(n)=O(1) + O(n)* O(n) + O(n) = **$O(n^2)$**, da $O(n^2)$ dominiert.

Algorithmus E hat eine Doppelschleife, von der nur eine Schleife von n abhängig ist. Dadurch ergibt sich eine O-Notation von O(n):

f(n)=O(1) + O(1) +O(n)* O(3) = **O(n)**, da O(n) dominiert.

In Algorithmus F wird einmal im Worst-Case *return algoE(i-2)+algoE(i-1)* d.h. zweimal eine Funktion mit der O-Notation O(n) aufgerufen, wobei i und n durch die Parameterübergabe dann die gleiche Zahl sind. Dadurch ergibt sich für algoF im Worst Case und im Average Case die O-Notation O(i):

Best Case: f(i)=O(1) + O(1) = O(1), da O(1) dominiert.

Worst Case: f(i)=O(1) + O(1) + (O(i-2) + O(i-1))

= O(1) + O(1) + (O(1) + O(1) +O(n)* O(3)) + (O(1) + O(1) +O(n)* O(3)) = **O(i)**,

da O(i) dominiert.

Das ergibt folgenden Average Case: **O(i)**