

1. Du kannst das Modell beschreiben, mit dem die Komplexität von Algorithmen mit der O -Notation gemessen wird.
2. Du kannst die (minimale) Laufzeitkomplexität $O(f(n))$ aufgrund einer gegebenen Zeitfunktion $T(n)$ durch Umformen und Vereinfachen ermitteln.
3. Du kannst allgemein die Laufzeitkomplexität einer Summe und eines Produkts von k Zeitfunktionen $T_1(n), T_2(n), \dots, T_k(n)$ bestimmen.
4. Asymptotische Laufzeitkomplexität bedeutet, dass eine Zeitfunktion $T_1(n)$ erst ab einer bestimmten Problemgrösse n_0 für alle weiteren n eine höhere (oder tiefere) Laufzeitkomplexität als eine Zeitfunktion $T_2(n)$ hat. Du kannst diesen kritische Problemgrösse n_0 für geeignete $T_1(n), T_2(n)$ ermitteln.
5. Du kannst aus bekannten Laufzeiten und Laufzeitkomplexitäten die Laufzeiten für neue Problemgrössen extrapolieren (schätzen).
6. Du kannst die Laufzeitkomplexitäten einfacher Python-Programme abschätzen, wobei für jeden elementaren Rechenschritt dieselbe Verarbeitungszeit verwendet wird.
7. Du kannst die häufig vorkommenden (handhabbaren) Laufzeitkomplexitäten $O(1)$, $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$, $O(n^3)$ nach aufsteigender Grösse ordnen und jeweils einen Beispiyalgorithmus dafür angeben.