Anton Eichstädt, Joshua Storost Theoretische Informatik I

Abgabe Aufgabenblatt 1

Übungsgruppe: Gruppe 8 Wintersemester 2023/24

Bonusaufgabe B4:

Eingabe:
Ein Array A der Länge n

Ausgabe:
Ein Element e aus A, welches Mehrheitselement ist oder nil

Algorithmusbeschreibung:
Der Algorithmus guckt sich jedes Element aus A an und behält dabei in jedem Schritt einen Mehrheitsindex und einen Mehrheitscounter, der mit 0 initialisiert ist.
In jedem Schritt überprüft der Algorithmus, ob das aktuelle Element gleich zum Element am Mehrheitsindex ist. Wenn ja, wird der Mehrheitscounter um 1 erhöht. Wenn nicht, wird der Mehrheitscounter um 1 reduziert.
Wenn der Mehrheitscounter 0 ist, wird der Mehrheitsindex auf das aktuelle Element und der Mehrheitscounter auf 1 gesetzt.
Nachdem der Algorithmus einmal durch das ganze Array A durch ist, finden wir das Element e, auf das der aktuelle Mehrheitsindex zeigt und gehen das Array erneut durch und zählen dabei, wie oft e im Array A ist. Wenn e mindestens $\lceil \frac{n}{2} \rceil$ mal vorkommt, geben wir e zurück, sonst geben wir nil zurück.

Korrektheitsbedingung:
Der Algorithmus ist korrekt, wenn er e zurückgibt, wenn e mindestens $\lceil \frac{n}{2} \rceil$ vorkommt oder wenn es kein solches e gibt, gibt der Algorithmus nil zurück.

Korrektheitsbeweis:
Es gibt 2 Fälle:
Wenn ein Element e eine absolute Mehrheit im Array A besitzt kann der Mehrheitscounter für das Element am Ende des Arrays nicht 0 werden, da der Mehrheitscounter für e mindestens $\lceil \frac{n}{2} \rceil$ mal erhöht und maximal $\lceil \frac{n}{2} \rceil - 1$ verringert wird. Damit muss der Mehrheitscounter für dieses Element am Ende des Arrays am Ende mindestens 1 sein und damit zeigt der Mehrheitsindex auf unser Element. Dann überprüft der Algorithmus am Schluss nochmal, dass dieses Element wirklich eine absolute Mehrheit hat und gibt e zurück.
Wenn es kein Element e mit einer absoluten Mehrheit im Array A zeigt der Mehrheitsindex am Ende auf ein Element f im Array A. Dann geht der Algorithmus nochmal das Array durch und zählt wir oft f im Array A vorkommt. Da f nach Fallunterscheidung keine absolute Mehrheit besitzt, gibt der ALgorithmus nil zurück.

Laufzeit:
Der Algorithmus geht das Array 2 mal durch. Im ersten Durchgang wird für jeden

Schritt ein Vergleich und eine Veränderung des counters und möglicherweise eine Veränderung eines Indizes gemacht. All diese Operationen haben eine konstante Laufzeit und damit hat der erste Durchgang eine Laufzeit von $O(n)$. Der zweite Durchgang zählt, wie oft ein bekanntes Element in einem Array vorkommt, was auch eine Laufzeit von $O(n)$ hat. Damit hat der gesamte Algorithmus eine Laufzeit von $O(n)$.