Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. Peter Sanders
Hans-Peter Lehmann, Daniel Seemaier

# 5. Übungsblatt zu Algorithmen II im WS 2021/2022

http://algo2.iti.kit.edu/AlgorithmenII_WS21.php
{sanders, hans-peter.lehmann, daniel.seemaier}@kit.edu

**Aufgabe 1** (*Schlechter Zufall*)

Gegeben sei ein randomisierter Algorithmus badBit, der keine Eingabe liest und als Ausgabe zufällig mit Wahrscheinlichkeit $p$ die Zahl 0 und mit Wahrscheinlichkeit $q = 1 - p$ die Zahl 1 liefert. Es sei $0 < p < 1$; der konkrete Wert von $p$ sei aber unbekannt.
Entwerfen Sie einen randomisierten Algorithmus fairBit, der keine Eingabe liest und als Ausgabe immer zufällig eine der Zahlen 0 und 1 mit Wahrscheinlichkeit 1/2 liefert.
Was können Sie über die Laufzeit Ihres Algorithmus sagen?

**Aufgabe 2** (*Analyse: Speicherbandbreite (*)*)

Die Effizienz eines Algorithmus, der auf externem Speicher arbeitet, hängt von der gewählten Blockgröße $B$ in Zusammenspiel mit der maximalen Bandbreite $W_{max}$ und der durchschnittlichen Zugriffszeit $T_{seek}$ des externen Speichers ab.
Bestimmen Sie für die folgenden Fälle die Blockgröße, für die 90% der maximalen Bandbreite ausgereizt werden kann. Sie können davon ausgehen, dass ohne Unterbrechung auf ganze Blöcke in zufälliger Reihenfolge zugegriffen wird. Etwaige Berechnungen können als asynchron angenommen werden. Daher muss für diese keine Zeit berücksichtigt werden.

a) $W_{max} = 144$ MByte/$s$, $T_{seek} = 12ms$ (*Lesen von Festplatte*)

b) $W_{max} = 550$ MByte/$s$, $T_{seek} = 100\mu s$ (*Lesen von SSD*)

c) $W_{max} = 68$ MByte/$s$, $T_{seek} = 60s$ (*Lesen von LTO Streamer*)

**Aufgabe 3** (*Analyse: Externer Stack*)

In der Vorlesung wurde eine Implementierung von *Stack* als externe Datenstruktur vorgestellt. Eine äquivalente Implementierung besitzt folgende Struktur: Im Speicher wird ein Puffer $P$ der Größe $2B$ gehalten – $B$ sei die Blockgröße beim Zugriff auf externen Speicher. Der Puffer ist in Form eines (internen) Stacks organisiert und enthält die neuesten gespeicherten Elemente. Folgende Operationen sind für die externe Datenstruktur definiert:

`pop` Falls $P$ nicht leer, entferne das neueste Element aus $P$. Ansonsten, lese einen Block ein, um die Hälfte von $P$ zu füllen bevor `pop` auf $P$ ausgeführt wird.

`push` Falls $P$ nicht voll, füge das neue Element direkt zu $P$. Ansonsten, schreibe die ältere Hälfte von $P$ in den externen Speicher und verschiebe die aktuellere Hälfte an diese Stelle im Speicher. Anschließend führe ein `push` auf $P$ aus.

Für die Analyse können Sie davon ausgehen, dass ein Block $B$ Elemente des Stacks halten kann.

a) Zeigen Sie, dass die Operationen `push` und `pop` amortisiert $O(1/B)$ I/O Operationen benötigen.

b) Warum genügt es nicht, nur einen Puffer mit Größe $B$ zu verwenden?

**Aufgabe 4** (*Entwurf+Analyse: Telekommunikationsgesellschaft*)

Eine Telekommunikationsgesellschaft beauftragt Sie eine Anwendung zu schreiben, die monatlich die $k$ Kunden bestimmt, bei denen sich die Rechnung im Vergleich zum Vormonat am meisten verändert hat. Diese Kunden will sich die Telekommunikationsgesellschaft noch einmal genau anschauen, um ihnen eventuell einen neuen Vertrag anzubieten.
Die zu bearbeitenden Daten werden Ihnen auf (langsamen) Bandspeichern zur Verfügung gestellt. Sie erhalten eine Liste mit den aneinandergefügten Datensätzen jeder Zweigstelle ihres Auftraggebers für den aktuellen Monat. Außerdem haben Sie eine entsprechende Liste für den Vormonat zur Verfügung. Gespeichert sind jeweils Tupel $(Kundennummer, Kosten)$.

a) Geben Sie einen Algorithmus an, der die geforderte Aufgabe erfüllt. Geben Sie außerdem die Laufzeit Ihres Algorithmus an und begründen Sie diese. Sie können davon ausgehen, dass die $k$ zu bestimmenden Kunden in den Hauptspeicher passen.

b) Seien nun die $k$ zu bestimmenden Kunden zu groß, um im Hauptspeicher gehalten zu werden. Ändern Sie Ihren Algorithmus so ab, dass er mit der erhöhten Datenmenge zurecht kommt. Geben Sie die Laufzeit Ihres neuen Algorithmus an und begründen Sie diese.

**Hinweis:** Diese Aufgabe war ursprünglich für die Klausur vorgesehen.

**Aufgabe 5** (*Entwurf: Vorlesungs-Permutation*)

Für eine gegebene Algorithmen-Vorlesung speichert das Array $F$ an Position $F[i]$ den Foliensatz zu Kapitel $i$. Die Vorlesung besitzt $N$ Foliensätze, was zu viele sind, um diese im Hauptspeicher zu halten. Gegeben sei weiterhin ein Array $R$, welches die Reihenfolge angibt, in welcher die Kapitel tatsächlich gehalten wurden. Wird also beispielsweise zuerst Kapitel 3 gehalten, ist $R[0] = 3$. Array $R$ passt ebenfalls nicht in den Hauptspeicher. Auf der Vorlesungs-Website sollen nun die Foliensätze in der Reihenfolge zur Verfügung gestellt werden, in der sie gehalten wurden.
Geben Sie einen Algorithmus im External Memory Modell an, der mit $O(\frac{N}{B} \cdot \log(\frac{N}{M}) + \frac{N}{B})$ IO-Zugriffen die Foliensätze so permutiert, dass diese in der Reihenfolge sind, in der sie gehalten wurden. Sie können davon ausgehen, dass in jedem Array-Element (sowohl in $F$ als auch in $R$) noch genug Platz für eine weitere Zahl frei ist, die Sie als Hilfs-Variable benutzen können. $B$ bezeichnet die Block-Größe des externen Speichers und $M$ die Größe des Hauptspeichers.

**Ausgabe:** 20.12.2021
**Abgabe:** keine Abgabe, keine Korrektur