



Übungsblatt 08

Algorithmen I - Sommersemester 2024

Abgabe im ILIAS bis 21.06.2024, 18:00 Uhr

Die Abgabe erfolgt alleine oder zu zweit als eine PDF-Datei über das Übungsmodul in der Gruppe eures Tutoriums im ILIAS. Beschriftet die Abgabe deutlich mit Matrikelnummer und Name.

- Achtet bei handschriftlichen Abgaben auf Lesbarkeit.
- Achtet darauf, ob Algorithmen in Worten oder Pseudocode beschrieben werden sollen.
- Es werden immer asymptotisch möglichst effiziente Algorithmen erwartet, wenn nicht anders angegeben.
- Wenn Korrektheits- oder Laufzeitanalysen gefordert sind, behandelt diese separat von der Algorithmenbeschreibung.

Gesamtpunkte: 20 (+ 11 Bonus)

Aufgabe 1 - Heappy Potter und die Heiligtümer des Sortierens (11 Punkte)

Auf vergangenen Blättern bist du bereits verschiedene Algorithmen im Bereich des Sortierens begegnet. Da du in der Vorlesung Heaps und deren Verwendung in Heapsort kennengelernt hast, kommt dir der Gedanke, dass dir Heaps helfen könnten, um andere Sortierprobleme schneller zu lösen. Für heute hast du dir vorgenommen Sortieraufgaben der letzten Blätter, wie `k-WAY-MERGESORT`, das Sortieren eines `k`-sortierten Arrays und die Auswahl des `k`-kleinsten Elements, noch einmal anzugehen. Hierbei gehen wir davon aus, dass der Datensatz paarweise verschiedene Zahlen enthält.

- a) Verwende einen Heap, um den Algorithmus `k-WAY-MERGE` umzusetzen und gib Pseudocode für diese Variante des Algorithmus an. Der Algorithmus erhält `k` sortierte Arrays und gibt ein sortiertes Array, das die selben Elemente enthält zurück. Der in dieser Aufgabe entworfene `k-WAY-MERGE` Algorithmus soll dann in dem aus Blatt 4 Aufgabe 2 c) bekanntem `k-WAY-MERGESORT` Algorithmus verwendet werden können. Begründe die Laufzeit, abhängig von `k` und Eingabearraygröße `n`, des resultierenden `k-WAY-MERGESORT`.

Hinweis: verwende nicht die MERGE Methode aus der Vorlesung als Subroutine. (3.5 Punkte)

- b) Für ein Array $A : [\mathbb{N}_0, n]$ und ein Element e ist $\text{sortiert}_A(e)$ die Position, die e in A hätte nachdem wir das Array sortieren. Für ein Array A und ein Element e ist $A(e)$ die Position, die e in A hat. Wir bezeichnen das Array A als k -sortiert, falls für jedes Element e aus A gilt, dass $|\text{sortiert}_A(e) - A(e)| \leq k$.

Beschreibe nun in Worten einen Algorithmus, der ein k -sortiertes Array sortiert. Begründe die Laufzeit und Korrektheit deines Algorithmus. (3.5 Punkte)

- c) Für ein unsortiertes Array $A : [\mathbb{N}_0, n]$ wollen wir das k -kleinste Element in A finden. Beachte, dass nach dem Sortieren von A das k -kleinste Element an Index $k - 1$ steht. Beschreibe einen Algorithmus, der das k -kleinste Element in A mit einem Zeitbedarf in $O(n + k \cdot \log(k))$ ausgibt. Begründe die Korrektheit und die Laufzeit deines Algorithmus. (4 Punkte)

Hinweis: In dieser Laufzeit kannst du zwar `buildHeap` auf A ausführen, aber du kannst auf dem aus A resultierenden Heap `popMin`, `push` und `decPrio` nicht für k Elemente aufrufen. Für k Aufrufe der obigen Funktionen benötigst du einen kleineren Heap der Größe $\mathcal{O}(k)$.

Aufgabe 2 - Heappy Potter und der Gefangene von ask a banality

(5 Punkte)

Du darfst in dieser Aufgabe Min- und Max-Heaps, wie in der Vorlesung eingeführt, nutzen. Max-Heaps sind dabei analog zu Min-Heaps definiert. Zeige oder widerlege die folgenden Aussagen zu binären Heaps der Größe n .

- a) Es ist möglich eine Priority-Queue so zu implementieren, dass `push` auf dieser Datenstruktur Laufzeit $o(\log(n))$ hat. (1 Punkt)
- b) Es gibt eine Permutation der Zahlen $\{1, \dots, n\}$, so dass das Einfügen der Zahlen in einen Min-Heap in dieser Reihenfolge insgesamt Zeit $\Theta(n)$ benötigt. (1 Punkt)
- c) Es gibt eine Permutation der Zahlen $\{1, \dots, n\}$, so dass das Einfügen der Zahlen in einen Min-Heap in dieser Reihenfolge insgesamt Zeit $\Theta(n \cdot \log(n))$ benötigt. (1 Punkt)
- d) Sei A die Umsetzung eines Min-Heaps als Array. Wenn wir nun die Reihenfolge der Elemente in A umkehren, ist das Ergebnisarray die Umsetzung eines Max-Heaps. (1 Punkt)
- e) Wir können einen Max-Heap in Linearzeit in einem Min-Heap umbauen. (1 Punkt)

Aufgabe 3 - Heappy Potter und Major Tom Riddle (4 Punkte)

Dr. Meta hat sich bei seinen Bibern mit EM-Fieber angesteckt. Als verwöhnter Erfolgsfan möchte er natürlich nur die Mannschaft unterstützen, die am Ende auch den Titel gewinnt. Daher hat er sich das folgende Vorgehen für seine Prognose überlegt. Zuerst repräsentiert er die m teilnehmenden Mannschaften jeweils durch eine Gruppe von Bibern. Die Mannschaften werden dann in einer Tabelle basierend auf ihrer Punktzahl angeordnet. Bei Punktgleichheit steht das Team mit kleineren Namen (Sortierung analog zum Duden) in der Tabelle weiter oben. Die Punkte werden durch Duelle, die stets mit einem Sieger enden, zwischen den Bibern, welche die jeweiligen Mannschaften repräsentieren, ermittelt. Nach jedem Duell erhält der Sieger zwei Punkte und dem Verlierer wird ein Punkt abgezogen. Zu Beginn haben alle Teams 0 Punkte. Solange zwei Teams nicht gespielt haben, bleibt diese initiale Ordnung erhalten. Dr. Meta wählt nun so lange zwei Teams aus und lässt diese gegeneinander antreten, bis ein Team n Punkte erreicht hat. Die Mannschaft, die dies erreicht, kann mit Dr. Metas Unterstützung rechnen. Eines der beiden ausgewählten Teams ist stets der aktuelle Tabellenführer, das andere wird durch das Oktopus-Orakel Paul aus den restlichen Teams ausgewählt. Dies erfolgt durch Aufruf der Funktion `paulChoose()` in $\Theta(1)$. Die Funktion gibt hierbei einen Zeiger auf die ausgewählte Mannschaft zurück.

Wir betrachten zur Veranschaulichung das folgende Beispiel mit $n = m = 5$ und Testnationen aus aller Welt. Es treten die Mannschaften Aserbaumschan, Bauhamas, Costa Rinda, Dämmemark und Eibenbaumküste an.

Start		Nach A > D		Nach E > A		Nach E > A		Nach E < B		Nach E > D	
Team	Punkte	Team	Punkte	Team	Punkte	Team	Punkte	Team	Punkte	Team	Punkte
A	0	A	2	E	2	E	4	E	3	E	5
B	0	B	0	A	1	A	0	B	2	B	2
C	0	C	0	B	0	B	0	A	0	A	0
D	0	E	0	C	0	C	0	C	0	C	0
E	0	D	-1	D	-1	D	-1	D	-1	D	-2

Die Graphik zeigt den Verlauf der Testrunden. Es ist jeweils das aktuelle Duell, so wie die daraus resultierende Tabelle angegeben. Die Kleiner- bzw. Größerzeichen geben den Sieger des Duells an. Der aktuelle Tabellenführer steht stets links und das durch Paul gewählte Team rechts. Alle Teams sind durch ihren Anfangsbuchstaben repräsentiert.

- a) Zeige, dass für beliebige $m \geq 2$, eine Mannschaft, die zu keinem Zeitpunkt (bevor n Punkte von einem Team erreicht wurden) der aktuelle Tabellenführer ist, trotzdem als erstes n Punkte erreichen kann. (1 Punkt)

Hinweis: Für einen Existenzbeweis dieser Art, darfst du stets das Ergebnis des Orakels selbst wählen.

- b) Beschreibe nun in Worten, wie du Dr. Metas Vorhersage mit Min- oder Max-Heaps möglichst effizient umsetzen kannst. Begründe die Laufzeit abhängig von m und der Anzahl der Duelle d , die benötigt werden bis der Sieger gefunden wurde. (3 Punkte)

Aufgabe 4 - Zum Knobeln (0 Punkte + 11 Bonus)

Eine Legende besagt, dass der Student, der diese Aufgabe lösen kann, sich den ewigen Respekt Dr. Metas verdient hat:

Starte mit einer Zahl $n \in \mathbb{N}$ und stelle diese in 2er-Superpotenz dar. Das bedeutet, dass wir die Zahl als Summe von 2er-Potenzen darstellen und dann rekursiv wiederum alle Exponenten als 2er-Potenz darstellen. Dies erfolgt so lange, bis wir nur Zahlen kleiner gleich 2 nutzen.

Im nächsten Schritt ersetzen wir jede 2 durch eine 3. Danach ziehen wir 1 von dem Ergebnis ab.

Dieses Vorgehen wiederholen wir. Also als 3er-Superpotenz darstellen, 3 durch 4 ersetzen, 1 abziehen, usw.

Beispiel:

- Zahl: 25
- 2er-Superpotenz: $25 = 2^4 + 2^3 + 1 = 2^{2^2} + 2^{2^1+1} + 1$
- 3er-Superpotenz: $3^{3^3} + 3^{3^1+1} + 1 \geq 7 \cdot 10^{12}$
- 1 abziehen: $3^{3^3} + 3^{3^1+1}$

a) Gib für die Zahl 4 die ersten sechs Folgenglieder an. (1 Punkt)

b) Zeige, dass wir so für jedes n irgendwann bei 0 ankommen. (10 Punkte)

Hinweis: Diese Aufgabe ist schwerer als bisherige Knobelaufgaben. Verzweifle also nicht, falls du nicht auf die Lösung kommst. Es kann trotzdem interessant sein, etwas herum zu überlegen.