Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. Peter Sanders
Moritz Laupichler
Hans-Peter Lehmann

6. Übungsblatt zu Algorithmen II im WS 2022/2023

http://algo2.iti.kit.edu/AlgorithmenII_WS22.php
{sanders, moritz.laupichler, hans-peter.lehmann}@kit.edu

Aufgabe 1 (*Analyse: ADAC Mitgliedschaft*)

Der "**A**utomobil **D**urch **A**lgorithmiker **C**lub" (ADAC) leistet auf Autobahnen Pannenhilfe. Ein Autofahrer hat in seiner Zeit als Verkehrsteilnehmer n Pannen, $n \in \mathbb{N}_{\geq 0}$, für die er die Hilfe des ADAC in Anspruch nehmen muss. Für jede geleistete Pannenhilfe verlangt der Club eine Aufwandsentschädigung abhängig von der Schwere der Panne. Mitglieder beim ADAC müssen lediglich ein Viertel dieser Kosten bezahlen. Eine lebenslange Mitgliedschaft kann man sich durch eine Einmalzahlung in Höhe von 1000 DM (**D**ijkstra **M**ark) sichern.
Da nicht schon mit Erwerb des Führerscheins klar ist, wie viele Pannen man in seinem Leben haben wird und wie schwerwiegend diese sein werden, stellt sich die Frage, ab wann es sich lohnt eine Mitgliedschaft beim ADAC zu erwerben.

a) Geben Sie eine Strategie an, die einen kompetitiven Faktor (competitive ratio) von ∞ erreicht. Begründen Sie kurz.

b) Wie gut ist die Strategie, sich nie eine Mitgliedschaft beim ADAC zu sichern? Begründen Sie.

c) Zeigen Sie, dass folgende Strategie einen kompetitiven Faktor $c = 3$ hat. Die Strategie ist, sich beim Pannenhelfer eine Mitgliedschaft zu kaufen, wenn die momentan von ihm bearbeitete Panne die Gesamtausgaben für Pannen (ohne Mitgliedschaft) auf über 500 DM anheben würde.

Hinweis: Verwenden Sie die summierten Gesamtkosten K über alle Pannen (ohne Mitgliederrabatt).

Aufgabe 2 (*Analyse: Online-`gaming`-Algorithmen*)

Angestellte in einem Rechenzentrum haben einen recht eintönigen Job. Ihnen stehen zwar die größten Rechner zur Verfügung, Sie dürfen diese aber nicht selbst verwenden. Stattdessen heißt es nur, die Maschinen möglichst gut auszulasten und Rechnungen zu schreiben. Ein ziemlich langweiliger Job möchte man meinen – zum Glück gibt es noch Computerspiele.
Ein Mitarbeiter hat zu Weihnachten ein neues Computerspiel erhalten, dass er liebsten andauernd spielen würde. Diesem Wunsch steht leider seine Arbeit im Wege. Eine neue Dienstanweisung besagt, dass die teuren Großrechner zu mindestens 50% ausgelastet sein müssen. Da das Scheduling in diesem Rechenzentren noch von Hand durchgeführt wird, muss der spielfreudige Mitarbeiter die laufenden Jobs überwachen und schnell neue Jobs auf leerlaufende Maschinen verteilen. So bleibt leider nur wenig Zeit zum Spielen am Arbeitsplatz.
Jeder Job hat eine Mindestlaufzeit von 5 Minuten. Die Zeit zum Verteilen der Jobs kann für die Bestimmung der Auslastung vernachlässigt werden. Der Mitarbeiter kann in dieser Zeit aber nicht spielen. Ein Job hat während seiner Ausführung die Maschine exklusiv. Es gibt keine automatischen Benachrichtigungen über das Ende eines Jobs. Der Mitarbeiter muss dies selbst periodisch überprüfen.

a) Entwerfen Sie einen *online scheduling* Algorithmus, der die freie Zeit des Mitarbeiters unter Einhaltung der Nebenbedingungen maximiert, wenn er einen Großrechner zu betreuen hat.

b) Zeigen Sie, dass Ihr Algorithmus optimal bzgl. der Freizeit des Mitarbeiters ist.

Aufgabe 3 (*Rechnen+Analyse: Suche in Strings*)

a) Zur Ausführung des KMP-Algorithmus muss zunächst ein sogenanntes *border-array* berechnet werden. Geben Sie das *border-array* für das Suchmuster $P =$ `abacababc` an.

b) Führen Sie den KMP-Algorithmus auf dem Text $T =$ `abacababbabacababc` mit obigem Suchmuster durch.

c) Wie oft muss der KMP-Algorithmus ein Muster P der Länge $|P|$ maximal an einen Text T der Länge $|T|$ anlegen, falls P nicht in T vorkommt? Wie oft minimal? Geben Sie das Ergebnis in Abhängigkeit von $|P|$ und $|T|$ an. Geben Sie außerdem jeweils ein Beispiel für P und T an.

d) Zeigen oder widerlegen Sie:
Das *border-array* kann keine drei aufeinanderfolgenden Einträge enthalten, die jeweils um eins kleiner sind als ihr Vorgänger, falls der erste von diesen drei Einträgen auf ein Suffix der Größe $k \geq 3$ verweist, welches gleichzeitig echtes Präfix ist.

Beispiel: $\text{border}[10] = 3,\ \text{border}[11] = 2,\ \text{border}[12] = 1$

Kein Beispiel: $\text{border}[10] = 2,\ \text{border}[11] = 1,\ \text{border}[12] = 0$

Aufgabe 4 (*Rechnen: Suffixarrays und DC3-Algorithmus*)

Gegeben sei die Zeichenkette $s =$ `aberakadabera`.

a) Geben Sie den Suffixbaum für s an.

b) Geben Sie das Suffixarray für s an.

In der Vorlesung haben Sie einen Linearzeitalgorithmus zur Konstruktion von Suffixarrays kennengelernt. Dieser ist unter dem Namen *DC3-Algorithmus* bekannt. Im Folgenden soll der Algorithmus Schritt für Schritt per Hand ausgeführt werden.
Die Suffixe von s werden zunächst in drei Sequenzen $C^k = \langle s_i \mid (i \bmod 3) = k \rangle$ für $k \in \{0, 1, 2\}$ aufgeteilt. Danach müssen die Sequenzen C^0 und $C^{12} = C^1 \cup C^2$ lexikographisch sortiert werden.

Sortierung von C^{12}:

c) Geben Sie die Tripelsequenzen $R^k = \langle s[i..i+2] \mid (i \bmod 3) = k \rangle$ für $k \in \{1, 2\}$ an. Für $i \geq |s|$ gelte $s[i] =$ \$ (Auffüllen mit zusätzlichen Abschlusszeichen).

d) Bestimmen Sie den *Rang* der Tripel von $R^{12} = R^1 \circ R^2$. Sortieren Sie dazu die Tripel und entfernen mehrfache Vorkommnisse. Die Position eines Tripels in dieser Sortierung gibt seinen Rang an.

e) Die berechneten Ränge definieren eine eindeutige Bezeichnung für jedes Tripel in R^{12}. Drücken Sie R^{12} mit Hilfe dieser Ränge aus. Diese Darstellung ergibt die Zeichenkette s^{12}. Muss der DC3-Algorithmus eine Rekursion ausführen?

f) Geben Sie das Suffixarray SA^{12} für s^{12} von Hand an (unabhängig, ob der DC3-Algorithmus eine Rekursion durchführt). Vergewissern Sie sich, dass SA^{12} eine Sortierung von C^{12} beschreibt.

Sortierung von C^0:

g) Erstellen Sie eine Zuordnung *rank*, die jedem i mit $s_i \in C^{12}$ den Index von s_i in der sortierten Sequenz C^{12} zuweist. Für alle anderen i sei $rank(i) = 0$.

Formale Berechnung von *rank* mit Hilfe des Suffixarray SA^{12} nach:

$$\begin{array}{lcll} rank[3 \cdot (\mathrm{SA}^{12}[i]) & +1] & = & i \quad \mathrm{SA}^{12}[i] < |C^1| \\ rank[3 \cdot (\mathrm{SA}^{12}[i] - |C^1|) + 2] & & = & i \quad \mathrm{SA}^{12}[i] \geq |C^1| \end{array}$$

Alle anderen Werte von $rank[i]$ können gleich 0 gesetzt werden.

h) Erstellen Sie Tupel $(s[i], rank[i{+}1])$ f.a. $s_i \in C^0$ und sortieren diese lexikographisch. Vergewissern Sie sich, dass diese Sortierung einer Sortierung von C^0 entspricht.

Nachdem C^0 und C^{12} sortiert worden sind, kann das Suffixarray von s bestimmt werden:

i) Führen Sie eine Mischen-Operation auf C^0 und C^{12} aus. Die resultierende Sequenz wird mit C bezeichnet. Es gelten folgende Sortierkriterien:

$$s_i \leq s_j \iff \begin{cases} (s[i], \quad rank[i+1]) & \leq & (s[j], \quad rank[j+1]) & s_j \in C^1 \\ (s[i..i+1], rank[i+2]) & \leq & (s[j..j+1], rank[j+2]) & s_j \in C^2 \end{cases}$$

Vergewissern Sie sich, dass C lexikographisch sortiert ist und damit das Suffixarray induziert.

Notation:
- Alle Indizes fangen bei 0 an – analog zu Kapitel 9.3.6, auf dem die Aufgabe basiert.
- $s[i\ (..j)\ ]$: Zeichen an Stelle i (bis j) in s (z.B. $s[1..3] =$ `ber`)
- s_i: Suffix von s ab Stelle i (z.B. $s_2 =$ `erakadabera`)

Aufgabe 5 (*Rechnen+Analyse: LCP-Array*)

Gegeben sei die Zeichenkette $s = \texttt{salsadipp\$}$.

a) Geben Sie den Suffixbaum für s an.

b) Geben Sie das Suffixarray für s an.

c) Geben Sie das LCP-Array für s an.

Im Folgenden sei ein String T sowie dessen Suffixarray SA[·] und dessen LCP-Array LCP[·] gegeben.

d) Wie kann der längste sich wiederholende Substring in T effizient bestimmt werden? (der Substring darf sich dabei selbst überlappen)

e) Wie viele paarweise unterschiedliche Substrings kann ein String der Länge n maximal besitzen? Wie kann die tatsächliche Anzahl für einen konkreten String T bestimmt werden?

f) Ein String lässt sich schlecht komprimieren, wenn er wenig Redundanz besitzt. Ein Maß dafür ist die Anzahl paarweise unterschiedlicher Substrings normiert auf die mögliche Gesamtanzahl unterschiedlicher Substrings. Geben Sie an, wie dieses Maß für T berechnet werden kann.

Aufgabe 6 (*RMQ in Wavelet Trees*)

Gegeben sei ein Universum von Zahlen $\mathcal{U}$ und ein Feld A von Zahlen aus $\mathcal{U}$. Geben sie einen Algorithmus an, mit dem sich unter Benutzung eines Wavelet Trees in $\log |\mathcal{U}|$ Zeit $\arg\min_i\{A[i] \mid i \in [a, b]\}$ für Parameter a, b berechnen lässt.

Aufgabe 7 (*Rechnen: Kompression*)

a) Bestimmen Sie die *Lempel-Ziv-Faktoren* $f_i = (l_i, p_i)$ des Textes $T_1 = \texttt{abbbababbbb\$}$.

b) Gegeben seien folgende *Lempel-Ziv-Faktoren*. Bestimmen Sie den Text T_2, aus dem sie entstanden sind.

$$(0, \texttt{a}), (5, 1), (0, \texttt{b}), (6, 1), (7, 7), (0, \texttt{\$})$$

c) Angenommen, jedes Zeichen kann in 1 byte, sowie jeder *Lempel-Ziv-Faktor* in 2 byte repräsentiert werden. Wie viel Prozent des Speicherplatzes von T_2 konnte somit eingespart werden?

Ausgabe: 31.01.2023
Abgabe: keine Abgabe, keine Korrektur