RWTH Aachen
Lehrgebiet Theoretische Informatik
Kuinke–Rossmanith–Sanchez

WS 2016
Lösungsvorschlag 13
2.2.2017

Übung zur Vorlesung Effiziente Algorithmen

Tutoraufgabe 33

a) Leider kann es keinen kompetitiven deterministischen Algorithmus geben: Der Gegner kann mit einem sehr kleinen Gegenstand beginnen. Nimmt ihn der Algorithmus, dann folgt ein Gegenstand, der den Behälter ganz ausfüllt. Nimmt ihn der Algorithmus nicht, dann folgt einfach nichts mehr. In jedem Fall ist der Approximationsfaktor beliebig schlecht.

b) Ein Bit an Information könnte einem mitteilen, ob es insgesamt einen Gegenstand gibt, der mindestens halb so groß wie der Behälter ist. Wenn ja, dann kann er Algorithmus auf diesen warten und ihn einpacken. Wenn nein, kann er geizig alles einpacken was paßt und ist in beiden Fällen 2-kompetitiv.

c) Ein randomisierter Algorithmus kann einfach die Antwort auf b) raten. Mit einer Wahrscheinlichkeit von 1/2 wird der Behälter also mindestens halb gefüllt. Das bedeutet, er ist 4-kompetitiv. (Es gibt noch ein schlaueres Verfahren, das 2-kompetitiv ist).

Tutoraufgabe 34

Zuerst zeigen wir, daß FIFO kein Markierungsalgorithmus ist. Dazu betrachten wir die Eingabe $1, 2, \ldots, k, k+1, 2, 1$.

Nach dem Einlesen von $1, \ldots, k$ sind genau diese Werte im Cache und die Phase endet. Einlesen von $k+1$ beginnt die nächste Phase, 1 wird verdrängt. Nun wird 2 markiert, danach aber beim lesen von 1 sofort verdrängt.

Wir zeigen nun weiterhin, daß FIFO trotzdem k-kompetitiv ist.

Leider enthält FIFO am Ende einer Phase nicht genau die Seiten, die in der Phase nachgefragt wurden, wie obiges Beispiel zeigt.

Betrachten wir aber nun eine Eingabe $p_1, \ldots, p_n$, und sei T_i der Zeitpunkt, an dem FIFO genau ik Seitenfehler macht. Zwischen T_i und T_{i+1} treten in FIFO also genau k Seitenfehler auf.

Allerdings müssen zwischen T_i und T_{i+1} mindestens k verschiedene Seiten geladen werden. Angenommen es werden nur die Seiten $s_1, \ldots, s_l$ $l < k$ geladen, so bleiben diese im Cache von FIFO und werden nicht verdrängt, FIFO würde also höchstens l Seitenfehler machen. Damit sind die Abstände zwischen T_i und T_{i+1} aber mindestens so lang wie die einzelnen Phasen.

Da die optimale Lösung mindestens soviele Fehler macht wie es Phasen gibt, ist FIFO also k-kompetitiv.

Hausaufgabe 22 (10 Punkte)

Wir beweisen, daß LIFO nicht kompetitiv ist.

Angenommen LIFO wäre c-kompetitv für irgendein festes c. Wir betrachten dann die Eingabe $1, 2, \ldots, k, (k+1, k)^c$. LIFO wird erst den Cache mit den Seiten $1, \ldots, k$ füllen, und muss dann die Seite $k+1$ nachladen. Da die Seite k zuletzt geladen wurde, wird diese überschrieben. Danach wird aber wiederum k geladen, und $k+1$ entfernt. Dies wird c mal wiederholt und resultiert somit in $2c$ Seitenfehlern.

Eine optimale Strategie kommt offensichtlich mit einem Seitenfehler aus, indem einfach Seite 1 verdrängt wird, sobald $k+1$ geladen wird.

Es gibt also kein c, so daß LIFO c-kompetitiv ist.