Institut für Informatik
Prof. Dr. Heiko Röglin
Anna Großwendt

Online-Algorithmen
Sommersemester 2015
Besprechung: KW 16

Präsenzblatt

Aufgabe 1

Eine hungrige Kuh ist auf der Suche nach einer Weide. Sie wandert ziellos durch die Gegend, bis sie auf einen Pfad trifft, von dem sie weiß, dass er zu einer Weide führt. Leider hat sie vergessen, in welche Richtung des Pfades sie wandern muss, um zu der Weide zu gelangen. Sie weiß lediglich, dass die Weide mindestens eine Längeneinheit entfernt ist.

(a) Geben Sie einen deterministischen Online-Algorithmus mit strikt kompetitivem Faktor 9 an, mit dem die Kuh die Weide finden kann.

(b) Zeigen Sie, dass für $r < 3$ kein deterministischer strikt r-kompetitiver Online-Algorithmus für dieses Problem existiert.

(c) Zeigen Sie, dass für $r < 9$ kein deterministischer strikt r-kompetitiver Online-Algorithmus für dieses Problem existiert.

(d) Geben Sie einen randomisierten Algorithmus an, der strikt 7-kompetitiv ist.

Aufgabe 2

Der Paging-Algorithmus CLOCK verwaltet die Cache-Seiten in einer zyklischen *Queue*. Jede einzelne ist mit einem *Used-Bit* ausgestattet. Es gibt einen Zähler, der auf die letzte Seite zeigt, die in den Cache geladen wurde. Sobald auf eine Seite zugegriffen wird, wird ihr Used-Bit auf 1 gesetzt. Bei einem Seitenfehler wird der Zähler um eins erhöht und der Algorithmus sucht ab dieser Stelle eine Seite, deren Used-Bit 0 ist. Die Used-Bits aller bis dahin betrachteten Seiten werden auf 0 gesetzt.

(a) Zeigen Sie, dass CLOCK kein Markierungsalgorithmus ist.

(b) Zeigen Sie, dass CLOCK k-kompetitiv ist.

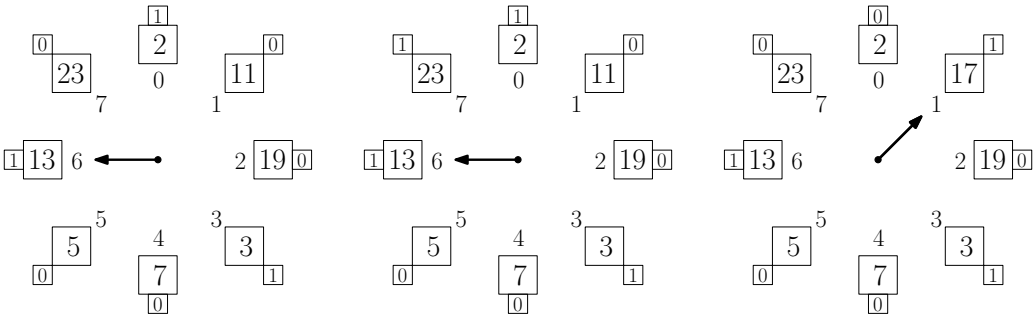

Abbildung 1: Veranschaulichung des CLOCK-Algorithmus; *links*: Situation zu einem willkürlichen Zeitpunkt - Seite 13 wurde als letzte in den Cache geladen; *Mitte*: Zugriff auf Seite 23, die bereits im Cache ist - Used-Bit wird auf 1 gesetzt; *rechts*: Zugriff auf Seite 17, also Seitenfehler - der Zähler wird von 6 über 7 und 0 auf 1 bewegt, wo das Used-Bit das erste Mal 0 ist.

Aufgabe 3

Wir betrachten *Scheduling* mit m identischen Maschinen: Gegeben seien Maschinen $1, \ldots, m$ und n Jobs mit Jobgrößen $p_1, \ldots, p_n$. Ein *Schedule* $\sigma\colon \{1, \ldots, n\} \to \{1, \ldots, m\}$ weist jeden Job einer Maschine zu, von der er abgearbeitet wird. Die Zeit, die eine Maschine zum Abarbeiten aller ihr zugewiesenen Jobs benötigt, ergibt sich aus der Summe der Jobgrößen dieser Jobs. Der *Makespan* eines Schedules ist das Maximum der Ausführungszeiten aller Maschinen. Ziel ist es, einen Schedule mit möglichst kleinem Makespan zu finden.

Bei der Online-Variante des Problems bekommen wir nach und nach die Jobgrößen p_j mitgeteilt. Wir betrachten den Online-Algorithmus Least Loaded, der einen Job immer der Maschine zuweist, die zur Zeit die kleinste Ausführungszeit besitzt. Zeigen Sie, dass Least Loaded strikt 2-kompetitiv ist.

Bitte tragen Sie sich bis zum 12. April 2015 im Tutorienvergabesystem für die Übungen ein.