

Online-Algorithmen

offline: Ganze Eingabe anfangs bekannt

online: Eingabe „stückweise“ präsentiert, Reaktion nach jedem Stück

Beispiel: Ski-Problem

Eine Skifahrerin fährt t Tage Ski, t ist unbekannt

Jeden Morgen entscheidet sie:

Kaufe ich mir jetzt Ski oder miete ich sie?

(Sagen wir: Mieten 10 EUR, Kaufen 100 EUR)

Kompetitive Analyse

Ein Online-Algorithmus hat **competitive factor** c , falls

$$C(I) \leq c \cdot C^*(I) + O(1).$$

$C(I)$: Lösung des Online-Algorithmus

$C^*(I)$: Optimale Offline-Lösung.

Worst-case-Analyse: Jede Eingabe ist möglich.

Laufzeit wird ignoriert!

Frage: Competitive factor des Skipproblems?

Approximation und Online-Algorithmen

Manche Approximationsalgorithmen bearbeiten ihre Eingabe sowieso seriell.

→ sie sind schon Online-Algorithmen

LPT-Scheduling:

5	3	3
5	3	
4	4	

Es gibt einen Online-Algorithmus mit competitive factor $4/3$ für das Mehrprozessorscheduling-Problem.

Ist das wirklich wahr?

Cache

LRU (Least Recently Used): verdränge diejenige Seite, deren letzter Zugriff am längsten zurückliegt

LFU (Least Frequently Used): ..., die am seltensten nachgefragt wurde

FIFO (First In First Out): ..., die sich am längsten im Cache befindet

LIFO (Last In First Out): ..., die zuletzt in den Cache geladen wurde

FWF (Flush When Full): entleere den Cache bei jedem Seitenfehler

LFD (Longest Forward Distance): verdränge diejenige Seite, deren nächster Zugriff am weitesten in der Zukunft liegt

Welche Strategie ist **nicht** online?

Markierungsalgorithmen

Der Cache bestehe aus k Seiten.

Die i te Phase bestehe aus einer maximalen Sequenz von Zugriffen, die auf höchstens k verschiedene Seiten zugreift und auf Phase $i - 1$ folgt.

Anfang einer Phase: Alle Seiten unmarkiert.

Seite wird markiert, wenn auf sie zugegriffen wird.

Ein **Markierungsalgorithmus** verdrängt nie eine markierte Seite.

Frage: Welche Strategien sind Markierungsalgorithmen?

Markierungsalgorithmen

LRU und FWF sind Markierungsalgorithmen.

Theorem

Ein Markierungsalgorithmus ist k -competitive.

Beweis

Idee: Alle markierten Seiten bleiben im Cache.

Nur ein Seitenfehler pro Markierung.

Nur k Seitenfehler pro Phase.

Jeder Algorithmus mindestens ein Seitenfehler pro Phase (ausser letzte).

LFU ist nicht kompetitiv

Seitenzugriffe:

$$p_1^t, p_2^t, \dots, p_{k-1}^t, (p_k, p_{k+1})^{t-1}$$

LFU verursacht $2(t - 1)$ Seitenfehler.

FIFO oder FWF (selbst mit nur zwei Seiten) nur $k + 1$ Fehler.

LFU ist nicht c -kompetitiv für jedes $c \in \mathbf{N}$.

LRU und FWF sind k -kompetitiv**Satz**

LRU und FWF sind Markierungsalgorithmen.

Beweis

FWF: Genau die markierten Seiten sind im Cache.

LRU: (Idee) Die zuletzt zugegriffenen Seiten sind markiert.

Korollar

LRU und FWF sind k -kompetitiv

Eine untere Schranke

Satz

Ein deterministischer Online-Algorithmus für das Cache-Problem hat mindestens einen competitive factor k .

Beweis

(Idee) Greife auf $k + 1$ verschiedene Seiten zu und danach m mal immer auf die Seite, die gerade verdrängt wurde.

→ $m + 1$ Seitenfehler!

LFD hat höchstens $k + 1 + m/k$ Seitenfehler.

Mache m sehr groß.

Randomisierte Online-Algorithmen

$C(I)$ ist jetzt eine Zufallsvariable.

Ein Algorithmus ist c -kompetitiv, wenn stets

$$E(C(I)) \leq c \cdot C^*(I) + O(1).$$

Ein „Gegner“ kennt den Algorithmus, aber nicht die Zufallsbits.

Könnte er die Zufallsbits, würde Randomisierung nicht helfen.