Eine Anfrage bzgl. Seite p verlangt, dass $p \in K$. D.h.,

- falls $p \in K$ ✓
- falls $p \notin K$ und noch hinreichend Platz in K, dann wird p geladen.
- falls $p \notin K$ und kein Platz in K, dann muss eine Seite $q \in K$ bestimmt werden. Diese wird dann in den Hintergrundspeicher abgelegt um Platz für p zu schaffen. Danach wird p geladen.

  Kosten hierfür: 1

Frage:

Wie soll q gewählt werden?

Zur Beantwortung dieser Frage werden wir sowohl deterministische als auch randomisierte Methoden kennenlernen.

2.1 Deterministische Algorithmen

Heuristiken:

- LRU (least-recently-used)

  Ersetze diejenige Seite, die am längsten nicht mehr angefragt worden ist.

- FIFO (first-in-first-out)

  Ersetze diejenige Seite, die am längsten im Kern=speicher ist.

- LIFO (last-in-first-out)

  Ersetze diejenige Seite, die am kürzesten im Kernspeicher ist.

- LFU (least-frequently-used)

  Ersetze diejenige Seite, die am wenigsten an=gefragt wurde, seitdem sie im Kernspeicher ist.

- LFD (longest-forward-distance)

  Ersetze diejenige Seite, deren nächste Anfrage am spätesten erfolgen wird

Bemerkung:
LFD ist eine offline-Heuristik. Alle anderen Heuristiken sind online.

Lemma 2.1
Sei ALG ein Pagingalgorithmus (online oder off=line). Dann können die Seitenaustausche in ALG derart verschoben werden, dass die Gesamtkosten sich nicht vergrößern und eine Seite erst dann in den Hauptspeicher geladen wird, wenn sie auch angefragt wird.

Beweis:
Hierzu verschiebt man einen Seitenaustausch unmittelbar vor die nächste Anfrage derjenigen Seite, die gerade geladen werden soll. Man kann sich leicht überlegen, dass sich dadurch die Kosten höchstens verringern können.

∎

Übung:
Arbeiten Sie den Beweis von Lemma 2.1 aus.

Eine LFD-Folge ist eine Folge von Seitenaustauschen, wobei die aus dem Hauptspeicher zu verdrängende Seite stets diejenige Seite ist, die auch mittels der LFD-Heuristik ermittelt worden wäre.

Wir können jeden Pagingalgorithmus derart interpretieren, dass dieser zunächst mit einer (möglicherweise leeren) maximalen LFD-Folge startet. Maximal bedeutet, dass der nachfolgende Seitenaustausch eine andere Seite zur Verdrängung aus dem Hauptspeicher auswählt, als diejenige, die aufgrund der LFD-Heuristik ausgewählt werden würde.

Satz 2.1

LFD ist ein optimaler Offline-Algorithmus.

Beweis:

Sei σ eine beliebige Anfragefolge und sei ALG ein optimaler Pagingalgorithmus bzgl. σ, für den gilt:

1. ALG erfüllt Lemma 2.1.
2. ALG startet mit einer längsten LFD-Folge unter allen optimalen Paging-Algorithmen.

Falls die Startfolge alle Seitenaustausche enthält, dann gilt ALG = LFD bzgl. σ, womit nichts mehr zu beweisen wäre.

Annahme: ALG $\neq$ LFD bzgl. σ.

Mit $p \leftrightarrow q$ bezeichnen wir den Seitenaustausch, der die Seite q aus dem Kernspeicher verdrängt und die Seite p in den Kernspeicher lädt.

Seien

$x \leftrightarrow v$ derjenige Seitenaustausch, der auf die Startfolge in ALG folgt und

$x \leftrightarrow u$ derjenige Seitenaustausch, den statt= dessen LFD durchführen würde.

Idee:

Konstruiere aus ALG einen optimalen Paging= algorithmus ALG* mit längerer LFD-Startfolge, was ein Widerspruch zur Eigenschaft 2 von ALG sein würde.

Durchführung:

ALG* führt zunächst die Startfolge von ALG aus. Dann wird anstatt dem von ALG durchgeführten Seitenaustausch $x \leftrightarrow v$ der von LFD durchgeführte Seitenaustausch $x \leftrightarrow u$ ausgeführt. Im Anschluss daran verfährt ALG* wie ALG bis in ALG ein Seitenaustausch $y \leftrightarrow u$ oder $v \leftrightarrow z$ erfolgt. ALG* ersetzt $y \leftrightarrow u$ durch $y \leftrightarrow v$ bzw. $v \leftrightarrow z$ durch $u \leftrightarrow z$. Danach verfährt ALG* genauso wie ALG.

Bemerkung:

Falls $v \leftrightarrow z$ durch $u \leftrightarrow z$ ersetzt wird, dann erfüllt ALG* nicht mehr Lemma 2.1. Wir schieben dann den Seitenaustausch $u \leftrightarrow z$ vor die nächste Anfrage bzgl. u, so dass ALG* danach Lemma 2.1 erfüllt.

Konstruktion $\Rightarrow$

Anfrage auf u erfolgt nach Anfrage auf v und somit nach dem Seitenaustausch $v \leftrightarrow z$.

$\Rightarrow$

ALG* löst das durch die Anfragefolge σ definierte Pagingproblem korrekt.

Ferner gilt: $ALG^*(\sigma) = ALG(\sigma)$.

Aber: ALG* startet mit einer strikt längeren LFD-Folge und erfüllt

Lemma 2.1.

Widerspruch!

Also ist die Annahme ALG $\neq$ LFD bzgl. σ falsch.

■

<u>(h,k) - Pagingproblem</u>: $h \leq k$

Online-Algorithmus hat Kernspeicher der Größe k.

competitive Analyse mit

Offline-Algorithmus mit Kernspeicher der Größe h.

<u>Markierungsalgorithmen</u>:

Kernspeichergröße k, Anfragefolge σ beliebig, aber fest.

<u>Idee</u>:

Unterteilung von σ in Phasen, die sogenannte <u>k-Phasenunterteilung</u>.

Phase i ist wie folgt definiert:

$i = 0$: die leere Anfragefolge.

$i > 0$: die maximale auf Phase $i-1$ folgende Anfragefolge σ', für die gilt:

σ' enthält $\leq k$ verschiedene Seitenanfragen.

Bemerkung:
Die k-Phasenunterteilung für σ ist eindeutig bestimmt.

Markierungsalgorithmen arbeiten nach folgenden Spielregeln:

1. Jede Seite erhält ein Markierungsbit.
2. Vor Beginn einer Phase werden alle Markierungen entfernt.
3. Wenn während einer Phase eine Seite angefragt wird, dann wird diese markiert.
4. Es wird niemals während einer Phase eine Seite in den Kernspeicher geladen, die auch nicht während derselben Phase angefragt wird.
5. Es wird niemals eine markierte Seite aus dem Kernspeicher verdrängt.

Satz 2.2
Sei ALG ein beliebiger Markierungsalgorithmus bzgl. eines Kernspeichers der Größe k. Dann ist ALG $\frac{k}{k-h+1}$ - competitive bzgl. Offline-Algorithmen, die einen Kernspeicher der Größe $h \leq k$ verwenden.

Beweis:

Fixiere eine beliebige Anfragefolge σ und betrachte die korrespondierende k-Phasenunterteilung.

Beh.:

Für jede Phase $i \geq 1$ verursacht ALG $\leq k$ Seitenwechsel.

Bew. d. Beh.:

Bis auf die letzte Phase gibt es in jeder Phase $i > 0$ k verschiedene Seitenanfragen. Sobald eine solche erfolgt, wird die betreffende Seite markiert und kann während der Phase nicht verdrängt werden. Jede Seite, die während einer Phase in den Kernspeicher geladen wird, wird auch in derselben Phase angefragt. Also folgt die Beh.

□

Sei OPT ein optimaler Offline-Algorithmus bzgl. der Anfragefolge σ mit Kernspeichergröße h. O.B.d.A. erfülle OPT Lemma 2.1.

Idee:

Ordne, bis auf die letzte, jeder Phase $i \geq 1$ mindestens $k-(h-1) = k-h+1$ der von OPT durchgeführten Seitenwechsel zu.

Durchführung:

Betrachte beliebige Phase $i \geq 1$ ungleich der letzten Phase. Sei q die erste in Phase i angefragte Seite.

Betrachte die Anfragefolge $\mathfrak{I}$, die mit der zweiten Seitenanfrage der Phase i beginnt und mit der ersten Seitenanfrage der Phase $(i+1)$ endet.

Konstruktion $\Rightarrow$

- J enthält k unterschiedliche Seitenanfragen, die alle eine Markierung der betreffenden Seiten bedingen.

- Da zu Beginn von Phase $(i+1)$ die Seite q sich im Kernspeicher befindet und die erste Seitenanfrage der Phase $(i+1)$ einen Seitenwechsel bedingt, befindet sich unter den k unterschiedlichen Seitenanfragen von J die Seite q nicht.

- Zu Beginn von J befindet sich die Seite q auch im Kernspeicher von OPT.

$\Rightarrow$

OPT führt während der Anfragefolge J $\geq k-(h-1)$ Seitenwechsel durch.

Diese ordnen wir der Phase i zu.

Die Nichtüberlappung der verschiedenen Phasen zugeordneten Anfragefolgen $\Rightarrow$

Ein Seitenwechsel des Algorithmus OPT wird maximal einer Phase zugeordnet.

Also gilt insgesamt

$$ALG(\sigma) \leq \frac{k}{k-h+1} OPT(\sigma) + \alpha,$$

wobei $\alpha \leq k$ die maximale Anzahl der von ALG während der letzten Phase durchgeführten Seitenwechsel ist.

■

<u>Satz 2.3</u>

LRU ist ein Markierungsalgorithmus.

<u>Beweis:</u>

Fixiere eine beliebige Anfragefolge σ und betrachte ihre k-Phasenunterteilung.

<u>Annahme:</u>

LRU ist kein Markierungsalgorithmus bzgl. σ.

$\Rightarrow$

LRU verdrängt während einer Phase eine markierte Seite x.

Betrachte die erste Anfrage bzgl. x während dieser Phase.

$\Rightarrow$

- x wird markiert.
- x ist unmittelbar nach dieser Anfrage diejenige Seite im Kernspeicher, die als letzte angefragt worden ist.

$\Rightarrow$

Damit x aus dem Kernspeicher verdrängt werden kann, müssen in derselben Phase mindestens k weitere unterschiedliche Seiten angefragt werden.

$\Rightarrow$

Die betrachtete Phase enthält Anfragen bzgl. $\geq k+1$ verschiedenen Seiten.

Dies ist ein Widerspruch zur Definition der k-Phasenunterteilung.

$\Rightarrow$

Die Annahme war falsch.

$\blacksquare$

Satz 2.4

FIFO ist kein Markierungsalgorithmus.

Beweis:

Übung

$\blacksquare$

Ziel:

Beweis, dass FIFO $\frac{k}{k-h+1}$ - competitive ist.

Ein Pagingalgorithmus ALG heißt konservativ, falls er folgende Eigenschaft besitzt:

Während einer beliebigen Anfragefolge mit $\leq k$ verschiedenen Seitenanfragen hat ALG $\leq k$ Seitenfehler (d.h., die angefragte Seite befindet sich nicht im Kernspeicher).

Satz 2.5

Sei ALG ein beliebiger konservativer Online-Pagingalgorithmus bzgl. eines Kernspeichers der Größe k. Ferner sei OPT ein beliebiger optimaler Offline-Algorithmus bzgl. eines Kernspeichers der Größe $h \leq k$, welcher zu Beginn eine Teilmenge des Kernspeicherinhaltes von ALG enthält. Dann gilt für alle Anfragefolgen σ

$$\mathrm{ALG}(\sigma) \leq \frac{k}{k-h+1} \mathrm{OPT}(\sigma).$$

Beweis:

Analog zum Beweis von Satz 2.2. Dort haben wir auch nur verwandt, dass die Anzahl der Seitenwechsel bei k unterschiedlichen Seitenanfragen durch k beschränkt ist.

■

Übung:

a) Arbeiten Sie den Beweis von Satz 2.5 aus.

b) Zeigen Sie, dass LRU und FIFO konservativ sind.

Ziel: Beweis einer unteren Schranke.

Hierzu beweisen wir zunächst eine obere Schranke für den optimalen Offline-Algorithmus LFD.

Lemma 2.2

Sei k die Größe des Kernspeichers. Dann gilt für jede endliche Anfragefolge σ, die stets eine Seite aus einer Menge von $k+1$ Seiten anfragt

$$LFD(\sigma) \leq \frac{|\sigma|}{k}.$$

Beweis:

Jedes Mal, wenn ein Seitenwechsel notwendig ist, verdrängt LFD diejenige Seite p mit dem größten Abstand zur nächsten Anfrage.

$\Rightarrow$

Vor der nächsten Anfrage bzgl. p erfolgen bzgl. den verbleibenen $k-1$ Seiten jeweils mindestens eine Anfrage.

$\Rightarrow$

Pro Anfrage mit Seitenwechsel können wir mindestens $k-1$ Anfragen ohne Seitenwechsel zählen. ∎

Satz 2.6

Sei k die Größe des Kernspeichers und sei ALG ein beliebiger Online-Pagingalgorithmus. Dann gibt es eine Anfragefolge σ beliebiger Länge, so dass $ALG(\sigma) \geq k \cdot LFD(\sigma)$.

Beweis:

Lemma 2.2 $\Rightarrow$

Es genügt bzgl. $k+1$ Seiten $p_1, p_2, \ldots, p_{k+1}$ eine beliebig lange Anfragefolge σ mit $ALG(\sigma) = |\sigma|$ zu konstruieren

Zu jedem Zeitpunkt ist exakt eine Seite nicht im Kernspeicher. Der Gegenspieler fragt immer diese Seite an.

$\Rightarrow$

Jede Anfrage in σ verursacht für ALG einen Seitenwechsel.

$\Rightarrow$

$$ALG(\sigma) = |\sigma|.$$

■

Übung:

Beweisen Sie, dass sowohl für LIFO als auch für LFU keine Konstante c existiert, so dass der Online-Algorithmus c-competitive ist.

Bemerkung:

In

Borodin, El-Yaniv: Online Computation and Competitive Analysis, S. 40-41

wird der Einfluss des Kostenmodells diskutiert.

2.2 Randomisierte Algorithmen

Die Analyse von deterministische Pagingalgorithmen erfolgte immer im Vergleich zu der für den Algorithmus ungünstigste Anfragefolge. Auch bei randomisierte Pagingalgorithmen gehen wir davon aus, dass ein Gegenspieler existiert, der für den Algorithmus eine möglichst ungünstige Anfragefolge konstruiert. Dabei unterscheiden wir zwischen unterschiedlichen Gegenspielertypen.

Gegenspielermodelle:
Der Gegenspieler kennt stets den Online-Algorithmus inklusive der Wahrscheinlichkeitsverteilung, die der Algorithmus verwendet.

Hierbei unterscheiden wir adaptive und vergessliche Gegenspieler.

adaptive Gegenspieler:
Kennt zu jedem Zeitpunkt alles, was der Online-Algorithmus bisher getan hat und verwendet dieses Wissen bei der Auswahl der nächsten Anfrage.

vergessliche (oblivious) Gegenspieler:
Muss die gesamte Anfragefolge im voraus ohne obige Kenntnis wählen.

adaptive Gegenspielerkosten:

a) adaptiv - offline (ADOF)

Kosten Online-Algorithmus $\longleftrightarrow$
optimale Offlinekosten (bzgl. Anfragefolge, die Gegenspieler online kreiert).

b) adaptiv - online (ADON)

Der Gegenspieler muss seine kreierte Anfrage bedienen, bevor der Online-Algorithmus dies tut.

vergessliche Gegenspielerkosten: (OBL)

Kosten Online-Algorithmus $\longleftrightarrow$
optimale Offlinekosten.

Sei ADV ein Gegenspieler vom Typ aus $\{OBL, ADOF, ADON\}$. Wir sagen, dass ein Online-Algorithmus ALG c-competitive gegen ADV ist, falls eine Konstante α existiert, so dass $\forall \sigma$

$$(*) \quad E[ALG(\sigma) - c \cdot ADV(\sigma)] \leq \alpha.$$

Bemerkung

Es gilt: $OBL(\sigma) = OPT(\sigma)$ und $ADOF(\sigma) = OPT(\sigma)$

Unterschied:

- $OBL(\sigma)$ ist eine feste Größe.

- ADOF(σ) ist eine Zufallsvariable, da die Wahl von σ von den von ALG getroffenen Zufallsentscheidungen abhängt. D.h., σ ist eine Zufallsvariable und somit ADOF(σ) = OPT(σ) auch.

- ADON(σ) ist aus demselben Grund auch eine Zufallsvariable. Jedoch haben wir keine genaue Charakterisierung des Wertes ADON(σ).

Also kann (*) folgendermaßen geschrieben werden:

- ADV = OBL:

$$E(ALG(\sigma)] - c \cdot OPT(\sigma) \leq \alpha$$

- ADV = ADOF:

$$E[ALG(\sigma) - c \cdot OPT(\sigma)] \leq \alpha$$

Sei $\bar{R}_{ADV}(ALG)$ das Infimum über alle c, so dass ALG c-competitive gegen Gegenspieler vom Typ ADV ist. Dann heißt $\bar{R}_{ADV}(ALG)$ das <u>competitive Verhältnis</u> von ALG gegen Gegenspieler vom Typ ADV.

Folgende Heuristik für die zufällige Auswahl der zu verdrängenden Seite scheint sinnvoll zu sein:

RAND (Random):

Wenn immer ein Seitenwechsel durchzuführen ist, wähle die zu verdrängende Seite zufällig, wobei jede Seite in k mit derselben Wahrscheinlichkeit verdrängt wird.

Folgender Satz zeigt, dass das erwartete Verhalten von RAND gegen einen adaptiven Online-Gegenspieler genauso gut ist, als das von deterministischen Online-Algorithmen LRU oder FIFO.

Satz 2.7

Der Algorithmus RAND ist $\frac{k}{k-h+1}$-competitive gegen einen adaptiven Online-Gegenspieler für das (h,k)-Pagingproblem.

Beweis:

Siehe Borodin, El-Yaniv S. 47-48.

∎

Für vergessliche Gegenspieler können wir beweisen, dass es nicht besser geht. Hierzu ist folgendes Lemma nützlich:

Lemma 2.3

Sei W eine Zufallsvariable für die Wartezeit auf einen Erfolg in einer Folge von Bernoulli-versuchen mit Erfolgswahrscheinlichkeit p. Für $j \in \mathbb{N}$ definiere wie folgt die Zufallsvariable W_j:

$$W_j := \begin{cases} W & \text{falls } W \le j \\ j & \text{sonst} \end{cases}$$

Dann gilt $E[W_j] = \frac{1}{p}\left(1-(1-p)^j\right)$.

Beweis:

Es gilt:

$$E[W_j] = j \cdot Pr[W > j] + \sum_{i=1}^{j} i \cdot Pr[\text{1. Erfolg in Runde } i]$$

$$= j(1-p)^j + p \cdot \sum_{i=0}^{j-1} (i+1)(1-p)^i$$

Ziel:

Entwicklung einer allgemeinen Formel für $\sum_{i=0}^{j-1} (i+1)(1-p)^i$.

Bemerkung:

- Wenn wir die Formel wüssten, dann könnten wir diese mittels Induktion beweisen.
- Wie man auf die Formel kommt, kann man in
  R.L. Graham, D.E. Knuth, O. Patashnik: Concrete Mathematics: A Foundation for Computer Science, 2nd. Ed., Addison-Wesley 1994, Kapitel 2.2 - 2.3, insbesondere S. 33.
  nachlesen.

Wir erhalten

$$= j(1-p)^j + p\left[\frac{1 - jp(1-p)^j - (1-p)^j}{p^2}\right]$$

$$= \frac{1}{p}\left[1 - (1-p)^j\right].$$

$\blacksquare$

Satz 2.8

Das competitive Verhältnis von RAND gegen einen vergesslichen Gegenspieler für das (h,k)-Pagingproblem ist $\geq \frac{k}{k-h+1}$.

Beweis:

Sei $\varepsilon > 0$ vorgegeben. Seien $b_1, b_2, \ldots$ Seiten, die zu Beginn nicht in K_{RAND} sind.

Zunächst definieren wir eine Anfragefolge σ, für die wir dann die untere Schranke beweisen werden. Sei

$$\sigma = (b_1, a_2, a_3, \ldots, a_h)^\ell, (b_2, a_2, a_3, \ldots, a_h)^\ell, \ldots,$$

wobei $\ell \in \mathbb{N}$ später festgelegt wird.

Die Unterfolge $(b_i, a_2, a_3, \ldots, a_h)^\ell$ $i = 1, 2, \ldots$ heißt i-ter Block der Anfragefolge.

Zu Beginn des i-ten Blockes gilt:

- $b_i \notin K_{RAND}$

$\Longrightarrow$ Zu Beginn des i-ten Blockes gilt:
$|K_{RAND} \cap \{b_i, a_2, \ldots, a_h\}| \leq h-1$.

Betrachte RAND während der Bearbeitung eines Segmentes $b_i, a_2, a_3, \ldots, a_k$ des i-ten Blockes

Annahme:

Unmittelbar vor dem Abarbeiten des Segmentes gilt $|K_{RAND} \cap \{b_i, a_2, a_3, \ldots, a_k\}| \leq k-1$.

Betrachte folgenden Bernoulliversuch:

- Erfolg während des betrachteten Segementes
  $:\Leftrightarrow$
  Während des letzten Seitenfehlers (der wegen obiger Annahme existieren muss) gilt:
  $|K_{RAND} \cap \{b_i, a_2, a_3, \ldots, a_k\}| = k-1$
  und RAND lagert Seite $q \notin \{b_i, a_2, a_3, \ldots, a_k\}$ aus.

Konstruktion $\Rightarrow$

Solange die Bernoulliversuche erfolglos enden, enthält jedes Segment mindestens einen Seitenfehler.

Auswahlregel von RAND $\Rightarrow$

$$Pr[\text{Erfolg}] \leq \frac{k-(k-1)}{k}$$

Mittels Anwendung des Lemmas 3 mit $p = \frac{k-k+1}{k}$ und $j = \ell$ erhalten wir

E[Anzahl der Seitenfehler von RAND während Block i]

$$\geq \frac{k}{k-h+1}\left(1-\left(\frac{h-1}{k}\right)^{\ell}\right)$$

$\xrightarrow[\ell \to \infty]{} 0$

$$\geq \frac{k}{k-h+1} - \varepsilon \quad \text{für } \ell \text{ groß genug.}$$

■

Übung:
Modifizieren Sie den Beweis von Satz 2.8, so dass nur $(k+1)$ Seiten des Hintergrund=speichers benötigt werden.

Somit bringt die naheliegende Heuristik RAND für die zufällige Auswahl der zu verdrängenden Seite keine Verbesserung für das competitive Verhältnis gegenüber den besten bekannten deterministischen Online-Algorithmen für das (h,k)-Pagingproblem.

Idee
Verknüpfe RAND und Markierungsalgorithmen.

Obige Idee führt zu folgender Heuristik:

MARK (Markierungsalgorithmus):

- Zu Beginn sind alle Seiten in K_{MARK} unmarkiert.

- Anfrage p
  - Falls $p \in K_{MARK}$ und unmarkiert, dann markiere p.
  - Falls $p \notin K_{MARK}$, dann

    <u>Seitenaustausch</u> $p \leftrightarrow q$ und markiere p,

    wobei q unter allen nicht markierten Seiten in K_{MARK} zufällig gewählt wird. Falls dies nicht möglich ist, da alle Seiten markiert sind, dann entferne zunächst alle Markierungen.

<u>Ziel:</u>
Analyse von MARK gegen einen vergesslichen Gegenspieler.

<u>Bezeichnung:</u>
Sei $k \in \mathbb{N}$. $H_k = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{k}$ heißt <u>k-te harmonische Zahl.</u>

<u>Übung:</u>
Beweisen Sie $\ln k \leq H_k \leq 1 + \ln k \quad \forall k \geq 1$.

<u>Satz 2.9</u>

$$\bar{R}_{OBL}(\text{MARK}) \leq 2 \cdot H_k.$$

Beweis:

Annahme:

Zu Beginn gilt $k_{MARK} = k_{OBL}$.

Sei σ eine beliebige Anfragefolge.

Idee:

- Herleitung einer
  - oberen Schranke für die erwartete Anzahl der Seitenfehler von MARK.
  - unteren Schranke für die Anzahl der Seitenfehler in OPT(σ).
- Eine obere Schranke für das competetive Verhältnis von Mark ergibt sich dann aus dem Quotienten der beiden Schranken.

Durchführung:

Betrachte die zu σ korrespondierende k-Phasen=unterteilung.

Bezeichnungen (bzgl. Phase i):

- Seiten zu Beginn in k_{MARK} heißen alt.
- Jede in Phase i angefragte nichtalte Seite heißt neu.

Es gilt:

Jede während einer Phase angefragte Seite verläßt den Kernspeicher während dieser

Phase nicht mehr.

Sei m_i die Anzahl der neuen Seiten während Phase i. Die schlechteste Anordnung der Seitenanfragen für den online-Algorithmus während Phase i sieht wie folgt aus:

- Frage zunächst alle m_i neue Seiten an und
- dann führe alle Anfragen bzgl. alte Seiten durch.

Wenn dies so ist, dann verursachen die ersten m_i Anfragen für MARK einen Seitenfehler, also

Kosten m_i.

Ziel:

Analyse der erwarteten Anzahl von Seitenfehlern bzgl. den restlichen $k - m_i$ (ersten) Anfragen bzgl. alten Seiten.

Beobachtung:

Pr [j-te alte Seite befindet sich bei Anfrage in K_{MARK}]

$$= \frac{k - m_i - (j-1)}{k - (j-1)}$$

alte unmarkierte Seiten in K_{MARK}

alte unmarkierte Seiten insgesamt

Also gilt:

Pr[Anfrage auf j-te alte Seite bedingt Seitenfehler]

$$= 1 - \frac{k - m_i - (j-1)}{k-(j-1)}$$

$$= \frac{m_i}{k-j+1}.$$

Also gilt

E[Anzahl der Seitenfehler in Phase i]

$$= m_i + \sum_{j=1}^{k-m_i} \frac{m_i}{k-j+1}$$

$$= m_i + m_i \sum_{j=m_i+1}^{k} \frac{1}{j}$$

$$= m_i + m_i (H_k - H_{m_i})$$

$$= m_i (H_k - H_{m_i} + 1)$$

$$\leq m_i H_k.$$

Also gilt

$$MARK(\sigma) \leq H_k \cdot \left(\sum_i m_i \right).$$

Ziel:

Herleitung einer unteren Schranke für die Anzahl der Seitenfehler in $OPT(\sigma)$.

Definition der k-Phasenunterteilung und m_i

$\Longrightarrow$

Während den Phasen $(i-1)$ und i werden $\geq m_i + k$ verschiedene Seiten angefragt.

$\Rightarrow$

$\forall i > 1$ gilt:

Die Anzahl der Seitenfehler, die OPT während den Phasen $(i-1)$ und i macht, ist

$$\geq m_i.$$

Wegen $K_{OPT} = K_{MARK}$ zu Beginn der Phase 1 gilt:

Die Anzahl der Seitenfehler während der Phase 1 ist

$$\geq m_1.$$

Also gilt:

$$OPT(\sigma) \geq \frac{1}{2} \cdot \sum_i m_i.$$

Also gilt:

$$\frac{MARK(\sigma)}{OPT(\sigma)} \leq \frac{H_k\left(\sum_i m_i\right)}{\frac{1}{2}\left(\sum_i m_i\right)} = 2 \cdot H_k$$

■

<u>Übung:</u>

Exercise 4.5 in Borodin / El Yaniv
Exercise 4.6 in Borodin / El Yaniv.

<u>Ziel:</u>
Herleitung einer unteren Schranke für randomisierte Pagingalgorithmen.

Satz 2.10

Sei ALG ein randomisierter Pagingalgorithmus mit Kernspeichergröße k. Sei die Gesamtanzahl der Seiten $N \geq k+1$. Dann gilt
$$\bar{R}_{OBL}(ALG) \geq H_k.$$

Beweis:

Annahme: $N = k+1$

Ziel:
Konstruktion einer Anfragefolge σ durch einen vergesslichen Gegenspieler, so dass
$$E[ALG(\sigma)] \geq H_k \cdot OPT(\sigma).$$

Gegenspieler:

- Für $1 \leq j \leq N$ speichert sich der Gegenspieler zu jedem Zeitpunkt die aktuelle Wahrscheinlichkeit p_j, dass Seite $j \notin K_{ALG}$

Dies ist möglich, da der Gegenspieler die von ALG verwendete Wahrscheinlichkeitsverteilung kennt.

D.h.,

N-Tupel $(p_1, p_2, \dots, p_N)$ mit der Eigenschaft, dass nach jeder Anfrage
$$\sum_{j=1}^{N} p_j = 1$$
gilt.

Somit ist stets $(p_1, p_2, \ldots, p_N)$ eine Wahrscheinlichkeitsverteilung.

$\Rightarrow$

Erwartete Kosten für eine Anfrage bzgl. i-te Seite für ALG betragen p_i.

Struktur von σ:

σ besteht aus einer beliebig großen Anzahl von k-Phasen. Diese werden mit Hilfe eines Markierungsalgorithmus konstruiert. Dabei merkt sich der Gegenspieler stets die aktuell markierten Seiten.

Ziel:
Konstruktion einer k-Phasenunterteilung nebst dem Beweis, dass die erwartete Anzahl von Seitenfehlern von ALG in jeder k-Phase $\geq H_k$ ist.

Bemerkung:
OPT wird bzgl. jeder k-Phase höchstens einen Seitenfehler haben, woraus dann unmittelbar die untere Schranke folgt.

k-Phase:

Wir unterteilen die k-Phase in k Unterphasen, wobei

- i-te Unterphase startet, wenn genau $k-i+1$ Seiten unmarkiert sind.

- Am Ende der k-ten Phase bleibt die zuletzt angefragte Seite markiert, während von allen anderen Seiten die Markierungen entfernt werden. Dann beginnt eine neue k-Phase.

Bemerkung:
Beachten Sie, dass wir zum Beweis der unteren Schranke einen speziellen Markierungsalgorithmus konstruieren, dessen Markierungsregeln sich von den des früheren Markierungsalgorithmus unterscheiden.

Ziel:
Konstruktion der i-ten Unterphase, so dass die erwarteten Kosten von ALG $\geq \frac{1}{k-i+1}$ sind.

Bemerkung (*):
Es gilt: $\sum_{i=1}^{k} \frac{1}{k-i+1} = \sum_{j=1}^{k} \frac{1}{j} = H_k$.

Struktur der Unterphasen:

- Folge von ≥ 0 Anfragen bzgl. markierten Seiten gefolgt von
- einer Anfrage bzgl. einer unmarkierten Seite, die dann markiert wird.

j-te Unterphase

M Menge der zu Beginn der j-ten Unterphase markierten Seiten.

D.h., $|M| = j$ und
$r := k+1-j$ Anzahl der unmarkierten Seiten.

Sei
$$\gamma := \sum_{i \in M} p_i$$

Wir unterscheiden zwei Fälle.

1. Fall: $\gamma = 0$

Dann existiert eine unmarkierte Seite a mit
$$p_a \geq \frac{1}{r}$$

$\rightsquigarrow$

Als nächstes erfolgt eine Anfrage bzgl. Seite a.

2. Fall: $\gamma > 0$

Dann existiert eine Seite $m \in M$ mit
$$p_m = \varepsilon > 0.$$

Der Gegenspieler ergänzt σ um folgende Anfragefolge:

(1) Anfrage bzgl. Seite m

(2) <u>while</u> erwartete Kosten von ALG für j-te ~~Unterphase~~ Unterphase $< \frac{1}{r}$ und $\gamma > \varepsilon$

<u>do</u>

Anfrage bzgl. Seite $l \in M$ mit

$$p_l = \max_{i \in M} p_i .$$

<u>od.</u>

<u>Beobachtung:</u>

Jede Iteration der while-Schleife erhöht die erwartete Kosten um $\geq \frac{\varepsilon}{|M|}$.

$\Longrightarrow$

Die Anzahl der Durchläufe des Blockes der while-Schleife ist maximal $\frac{|M|}{r \cdot \varepsilon}$

$\Longrightarrow$

Verfahren terminiert.

Betrachte die Situation unmittelbar nach der Terminierung. Wir unterscheiden zwei Fälle:

<u>2.1</u> erwartete Kosten $\geq \frac{1}{r}$

- Beende die Unterphase mit einer Anfrage bzgl. einer beliebigen unmarkierten Seite.

<u>2.2</u> erwartete Kosten $< \frac{1}{r}$

$\Longrightarrow$

Die Schleife terminiert mit $\gamma \leq \varepsilon$.

- Beende die Unterphase mit einer Anfrage bzgl. einer unmarkierten Seite b mit p_b maximal.

$\Rightarrow$

$$p_b \geq \frac{1-\gamma}{r}$$

Also gilt für die erwarteten Kosten von ALG in der j-ten Unterphase

$$\begin{aligned} \geq p_m + p_b &\geq \varepsilon + \frac{1-\gamma}{r} \\ &\geq \varepsilon + \frac{1-\varepsilon}{r} \\ &= \frac{1}{r} + \underbrace{\left(\varepsilon - \frac{\varepsilon}{r}\right)}_{\geq 0} \\ &\geq \frac{1}{r} \end{aligned}$$

Somit haben wir unser Ziel erreicht. Aus Bemerkung (*) und der Tatsache, dass OPT bzgl. jeder k-Phase höchstens einen Seitenfehler hat folgt direkt

$$E[ALG(\sigma)] \geq H_k \cdot OPT(\sigma)$$

und somit

$$\bar{R}_{OBL}(ALG) \geq H_k$$