Das Ski Problem

Wir fahren in den Skiurlaub und sind vor die Entscheidung gestellt,

- ? jeden Tag Skier für 1 Euro pro Tag zu leihen oder
- ? für K Euro Skier zu kaufen.

Leider wissen wir nicht, wie lange die Skisaison dauert:

- ! Wenn wir am ersten Tag Skier kaufen, kann die Saison bereits am nächsten Tag beendet sein.
- ! Wenn wir jeden Tag mieten, hätten wir bei einer Saison von mindestens $K + 1$ Tagen besser am ersten Tag Skier gekauft.

- Eine optimale Entscheidung ist unmöglich, da wir das Wetter während des Rests des Urlaubs nicht kennen.
- Gibt es aber gute Strategien, deren Kosten nicht zuweit von den optimalen Kosten entfernt sind?

Eine optimale deterministische Strategie

Unser Ziel:

Der Quotient (oder Wettbewerbsfaktor)

$$\frac{\text{unsere Kosten}}{\text{optimale Kosten}}$$

sollte im Worst-Case so klein wie möglich sein.

- Unsere Strategie: Miete Skier während der ersten $K-1$ Tage und kaufe am Kten Tag.
  - Wenn die Saison höchstens $K-1$ Tage dauert, dann sind wir optimal, erhalten also den Wettbewerbsfaktor 1.
  - Wenn die Saison K Tage oder länger dauert, dann ist $\frac{K-1+K}{K} < 2$ unser Wettbewerbsfaktor.
- Wenn eine Strategie am Tag $k < K$ kauft, dann ist $\frac{k-1+K}{k} \geq 2$ der Wettbwerbsfaktor.
- Wir sind offensichtlich besser als jede Strategie, die später kauft.

Die allgemeine Fragestellung

- Ein on-line Algorithmus A erhält eine Folge $\sigma = (\sigma_1, \ldots, \sigma_n)$ und muss eine Ausgabefolge $A(\sigma) = (\tau_1, \ldots, \tau_n)$ berechnen.

  Wenn zusätzlich die Eingabe σ_{n+1} erscheint, dann muss A die Ausgabe τ_{n+1} bestimmen.

- Die Ausgabe wird durch eine Funktion f ausgewertet:
  Für jede Folge σ bestimme den Quotienten

$$\frac{f(\sigma, A(\sigma))}{\min_\tau f(\sigma, \tau)}.$$

- Der größte Wert des Quotienten, also

$$\max_\sigma \frac{f(\sigma, A(\sigma))}{\min_\tau f(\sigma, \tau)}$$

definiert den Wettbewerbsfaktor von A.

Die allgemeine Fragestellung II

Eine on-line Agorithmus, der ständig Entscheidungen zu treffen hat ohne die Zukunft zu kennen, muss sich mit dem Optimum vergleichen.

Und nicht nur das: Der Wettbewerbsfaktor misst die Leistung des on-line Algorithmus auf der schlechtesten Eingabefolge.

- Und für randomisierte on-line Algorithmen A?
  - Bestimme diesmal den Erwartungswert $E[f(\sigma, A(\sigma))]$.
  - Der Wettbewerbsfaktor ist jetzt

$$\max_{\sigma} \frac{E[f(\sigma, A(\sigma))]}{\min_{\tau} f(\sigma, \tau)}.$$

Randomisierung für das Ski Problem

Kaufe Skier zu Beginn des $i+1$ten Tages mit Wahrscheinlichkeit π_i und miete mit Wahrscheinlichkeit $1-\pi_i$.

- Wann hat π einen möglichst kleinem Wettbewerbsfaktor?

  Man stelle sich vor, dass ein Gegner, der die Verteilung kennt, auf möglichst gemeine Art und Weise die Skisaison beendet.

- Es wäre unklug, Skier an einem Tag $t > K$ zu kaufen, da ein Kauf am Tag K geringere Kosten verursacht.

  Wir können deshalb o.B.d.A. annehmen, dass $\pi_t = 0$ für $t > K$ gilt.

- Die erwarteten Kosten, wenn die Saison am Tag T endet, sind

$$E(T) = \sum_{t=0}^{T-1} (t+K) \cdot \pi_t + T \cdot \sum_{t=T}^{K} \pi_t.$$

  - Die Kosten betragen $t+K$, wenn am Tag $t+1$ gekauft wird.
  - Mit Wahrscheinlichkeit $\sum_{t=T}^{K} \pi_t$ wird an den Tagen $T+1, \ldots, K+1$ gekauft.

Was passiert für $\pi_t = 1/K$?

Die erwarteten Kosten, wenn die Saison am Tag T endet, sind

$$E(T) = \sum_{t=0}^{T-1} (t+K) \cdot \pi_t + T \cdot \sum_{t=T}^{K} \pi_t.$$

- Wir kaufen zu Beginn des Tages t mit Wahrscheinlichkeit $1/K$.
- Unsere erwarteten Kosten, wenn die Saison am Tag T endet, sind

$$\begin{aligned} E(T) &= \sum_{t=0}^{T-1} \frac{t+K}{K} + T \cdot \sum_{t=T}^{K} \frac{1}{K} \\ &= \frac{T \cdot (T-1)/2 + T \cdot K + T \cdot (K - T + 1)}{K} \\ &= \frac{4T \cdot K - T^2 + T}{2K}. \end{aligned}$$

- Und $E(1) = \frac{4K}{2K} = 2$: Der Wettbewerbsfaktor ist mindestens 2.

Die Analyse

Die erwarteten Kosten, wenn die Saison am Tag T endet, sind

$$E(T) = \sum_{t=0}^{T-1} (t+K) \cdot \pi_t + T \cdot \sum_{t=T}^{K} \pi_t.$$

Wir führen eine heuristische Rechnung durch:

- Wir nehmen an, dass die Variable T reellwertig ist und dass π eine auf $[1, K]$ definierte Verteilung beschreibt.
- Wenn der Wettbwerbsfaktor höchstens α ist, dann

$$F(T) := \int_{t=0}^{T} (t+K) \cdot \pi(t) \cdot dt + T \cdot \int_{t=T}^{K} \pi(t) \cdot dt \overset{!}{\leq} \alpha \cdot T.$$

- Nimm Gleichheit an und differenziere nach T:

$$F'(T) = (T+K) \cdot \pi(T) + \left[\int_{t=T}^{K} \pi(t) \cdot dt - T \cdot \pi(T) \right] \overset{!}{=} \alpha$$

Zwischenstand

$$F'(T) = (T+K)\cdot\pi(T) + \left[\int_{t=T}^{K} \pi(t)\cdot dt - T\cdot\pi(T)\right] \stackrel{!}{=} \alpha$$

- Differenziere nochmals, um das verbleibende Integral loszuwerden:

$$\begin{aligned} F''(T) &= \left[\pi(T) + (T+K)\cdot\pi'(T)\right] + \left[-\pi(T) - \pi(T) - T\cdot\pi'(T)\right] \\ &= K\cdot\pi'(T) - \pi(T) \stackrel{!}{=} 0 \end{aligned}$$

- Die Differentialgleichung $\pi(T) = K\cdot\pi'(T)$ hat aber genau die Lösungen

$$\pi(T) = c\cdot e^{T/K}$$

für eine beliebige Konstante c.

Eine optimale Verteilung π

Es ist $\pi(T) = c \cdot e^{T/K}$. Wie ist c zu wählen?

- π beschreibt eine Verteilung, d.h. es muss $1 = \int_0^K \pi(t) \cdot dt$ gelten.
- $c \cdot K \cdot e^{T/K}$ ist die Stammfunktion von $\pi(T)$, denn
  - $(c \cdot K \cdot e^{T/K})' = c \cdot e^{T/K} = \pi(T)$.
  - Wir müssen $1 = c \cdot K \cdot e - c \cdot K$ fordern, und damit ist $c = \frac{1}{K \cdot (e-1)}$.
  - Unsere Lösung ist

$$\pi(T) = \frac{1}{K \cdot (e-1)} \cdot e^{T/K}.$$

- Wie groß ist der Wettbewerbsfaktor α?
  - Setze π in die Bedingung $F'(T) \stackrel{!}{=} \alpha$ ein
  - und werte für $T = 0$ aus.

Der Wettbewerbsfaktor

Wir haben gefordert

$$F'(T) = (T + K) \cdot \pi(T) + \left[\int_{t=T}^{K} \pi(t) \cdot dt - T \cdot \pi(T) \right] \stackrel{!}{=} \alpha$$

Für $T = 0$ ist

$$\alpha = K \cdot \pi(0) + \int_{t=0}^{K} \pi(t) dt = \frac{1}{e-1} + 1 = \frac{e}{e-1}.$$

Wenn Skier zu Beginn des Tages T mit Wahrscheinlichkeit

$$\pi_T = \frac{1}{K \cdot (e-1)} \cdot e^{T/K}$$

gekauft werden, dann wird der Wettbewerbsfaktor $\frac{e}{e-1} \approx 1,58$ erreicht.

Das Minimum Makespan Problem

Es sind n Aufgaben $A_1, \ldots, A_n$ mit den Laufzeiten $t_1, \ldots, t_n$ gegeben.

Die Aufgaben sind so auf m Maschinen auszuführen, dass der „Makespan“, also die für die Abarbeitung aller Aufgaben anfallende Bearbeitungszeit, minimal ist.

- Wenn $I_j = \{i \mid A_i \text{ wird auf Maschine } j \text{ ausgeführt}\}$ die Menge der auf Maschine j auszuführenden Aufgaben ist, dann ist

$$\max_{1 \leq j \leq m} \left\{ \sum_{i \in I_j} t_i \right\}$$

der Makespan.

- Wir arbeiten mit einem Greedy Algorithmus:
  - Weise die Aufgaben der Reihe nach zu.
  - Führe die aktuelle Aufgabe auf der Maschine mit der bisher geringsten Last aus.

Wie gut ist Greedy Scheduling?

Maschine i trage die größte Last.

- Falls i nur eine Aufgabe ausführt, ist der on-line Algorithmus offensichtlich optimal.
- Sonst führt Maschine i mindestens zwei Aufgaben aus.
  - T sei die Gesamtlaufzeit von i und A_j sei die letzte von i ausgeführte Aufgabe.
  - Zum Zeitpunkt $T - t_j$ sind alle anderen Maschinen beschäftigt:

$$\sum_{k=1}^{n} t_k \geq (m-1) \cdot (T - t_j) + T = mT - (m-1) \cdot t_j \geq mT - m \cdot t_j$$

  - Und als Konsequenz

$$T \leq \frac{1}{m} \cdot \sum_{k=1}^{n} t_k \quad + t_j \leq 2 \cdot \max\left\{ \frac{1}{m} \cdot \sum_{k=1}^{n} t_k,\ t_j \right\}.$$

Greedy Scheduling hat den Wettbewerbsfaktor 2

Hat Greedy Scheduling einen kleineren Wettbewerbsfaktor?

- Eine worst-case Instanz.
  - $m \cdot (m-1)$ „kurze“ Aufgaben der Länge 1 sowie eine „lange“ Aufgabe der Länge m sind gegeben.
  - Wenn die kurzen vor der langen Aufgabe erscheinen, dann verteilt Greedy Scheduling die kurzen gleichmäßig über die m Maschinen:
    $$\text{Makespan} = m - 1 + m = 2m - 1.$$
  - Aber der optimal Makespan ist m.
- Der Wettbewerbsfaktor liegt beliebig nahe bei zwei.

Paging

Angenommen, wir können nur wenige Seiten eines langsamen externen Speichers in einem schnellen internen Speicher halten.

Wenn eine neue Seite angefordert wird, müssen wir eine andere Seite auslagern. Wie bestimmen wir die auszulagernde Seite?

- Die Kapazität des schnellen Speichers, also das Fassungsvermögen des Speichers, sei k.
- Ein on-line Algorithmus für das Paging-Problem erhält für jede angeforderte, aber nicht vorhandene Seite einen Strafpunkt.

Welche Wettbewerbsfaktoren können erreicht werden?

On-line Paging Strategien

- LFU (Least-Frequently-Used): Lagere die am wenigsten angeforderte Seite aus. (Wir zählen die Anzahl der Anforderungen vom Zeitpunkt der letzten Einlagerung der Seite an.)
  - Lese die Seiten $p_1, p_2, \ldots, p_k$ nacheinander ein, gefolgt von der Anforderungsfolge $p_2, \ldots, p_k$.
  - Danach starte die Anforderungen $p_0, p_1, p_0, p_1, \ldots$
    - ⋆ Für jede Seite gibt es einen Strafpunkt. Der Wettbewerbsfaktor ist durch keine Konstante beschränkt.
    - ⋆ Eine optimale Strategie erreicht aber nur einen Strafpunkt.
- LRU (Least-Recently-Used): Lagere die Seite aus, deren letzte Anforderung am weitesten zurückliegt.
- FIFO (First-In-First-Out): Lagere die Seite aus, die am längsten gespeichert wurde.
- RANDOM: Verdränge eine zufällig und uniform gewählte Seite aus dem Speicher.

Eine untere Schranke

- Sei A ein on-line Algorithmus und σ eine Folge, die zu einem ersten Strafpunkt führt.
- Zu diesem Zeitpunkt habe A die Seiten $p_1, \ldots, p_k$ gespeichert und die Seite p_0 ausgelagert.
- Jetzt konstruiere eine Eingabefolge σ' mit Anforderungen an die $k+1$ Seiten $p_0, \ldots, p_k$, so dass A einen Fehler auf **jeder** neu angeforderten Seite macht.
- Eine optimale off-line Strategie erhält andererseits auf $\sigma\sigma'$ (mit Anforderungen an nur $k+1$ Seiten) höchstens $\frac{|\sigma\sigma'|}{k}$ Strafpunkte.

Jede deterministische on-line Paging Strategie hat höchstens den Wettbewerbsfaktor k.

Der Wettbewerbsfaktor von LRU

Sei $\sigma = (\sigma_1, \sigma_2, \ldots)$ eine Eingabefolge, auf der LRU am schlechtesten abschneidet. Zerlege σ in Teilfolgen $\sigma = (\sigma^1, \sigma^2, \ldots)$, so dass

- σ^1 mit dem ersten Strafpunkt von LRU endet und
- σ^j für $j \geq 2$ nach Strafpunkt $(j-1) \cdot k + 1$ endet.

- Es genügt zu zeigen, dass jede off-line Strategie während jeder Teilfolge mindestens einen Strafpunkt erhält.
- Dies ist richtig für σ^1, denn

  LRU erhält ja,

  bei anfänglich leerem Speicher,

  nur dann einen Strafpunkt, wenn $k+1$ verschiedene Seiten nachgefragt wurden.

Analysiere das Verhalten von LRU auf den Teilfolgen σ^j für $j \geq 2$.

Der Wettbewerbsfaktor von LRU II

σ^j endet nach Strafpunkt $(j-1) \cdot k + 1$.

- Fall 1: LRU erhält in σ^j zwei Strafpunkte für dieselbe Seite p.
  - Nach dem ersten Strafpunkt für p wird p in den Speicher geholt.
  - Da ein zweiter Strafpunkt vergeben wird, wurde p zwischenzeitlich ausgelagert.
  - Dies passiert für LRU nur, wenn k neue Seiten zwischen dem ersten und zweiten Strafpunkt für p angefordert werden.
  - Also fordert σ^j mindestens $k+1$ verschiedene Seiten an:

    Jede off-line Strategie erhält auch mindestens einen Strafpunkt.
- Fall 2: LRU erhält Strafpunkte für k verschiedene Seiten.
  Sei q die Seite, auf der LRU den letzten Strafpunkt in σ^{j-1} erhält.
- Fall 2.1: LRU erhält einen Strafpunkt für q in σ^j.
  - Dieser Fall verläuft wie Fall 1.

Der Wettbewerbsfaktor von LRU III

Fall 2.2:

- q war die letzte Seite in σ^{j-1} mit einem Strafpunkt für LRU.
- LRU erhält aber keinen Strafpunkt für q in σ^j.

- Eine optimale off-line Strategie muss zu Beginn von σ^j die Seite q speichern, da gerade eine Anforderung für q erfolgte.
- Nach Annahme werden k verschiedene Seiten in σ^j angefordert, die alle von q verschieden sind.
- Damit muss jede off-line Strategie irgendwann einen Strafpunkt in σ^j erhalten.

LRU hat den optimalen Wettbewerbsfaktor k.
Auch FIFO hat, mit analogem Argument, den Wettbewerbsfaktor k

Randomisiertes Paging

Die Marking Strategie

(1) Die Seite p werde angefordert.

(2) Wenn p nicht gespeichert ist, wähle zufällig eine nicht-markierte Seite und lagere sie aus.

> Sind alle Seiten markiert, dann lösche alle Markierungen und wähle zufällig eine der jetzt unmarkierten Seiten zur Auslagerung.

(3) Markiere p.

- Marking schützt häufig nachgefragte Seiten durch die Markierung und arbeitet in dieser Hinsicht wie LRU.
- Mit der zufälligen Wahl einer auszulagernden Seite erhalten wir viele LRU-Varianten, von denen hoffentlich die meisten nur geringe Kosten verursachen.
- Aber Marking hat auch Ähnlichkeiten mit LFU.

Zerlegung der Eingabefolge σ

Zerlege die Eingabefolge σ in Teilfolgen $\sigma = (\sigma^0, \sigma^1, \ldots)$, so dass

- σ^1 mit dem ersten vergebenen Strafpunkt beginnt.
- σ^i für $i \geq 1$ ein längstes auf σ^{i-1} folgendes Eingabeintervall, in dem genau k verschiedene Seiten angefordert werden.

Die zu Anfang von σ^i gespeicherten Seiten

Genau die während σ^{i-1} angefordert Seiten sind zu Anfang von σ^i (für $i \geq 1$) gespeichert und markiert.

Der Beweis für $i = 1$ ist charakteristisch.

- Zu Begin von σ^1 erhält Marking einen Strafpunkt.

  Marking löscht die Markierung der k verschiedenen, während σ^0 angeforderten Seiten.

- Alle während σ^1 angeforderten Seiten werden sofort markiert und diese Markierung wird während σ^1 nicht gelöscht:

  Es gibt ja noch unmarkierte Seiten.

- Am Ende von σ^1 sind genau die während σ^1 angeforderten Seiten gespeichert und markiert.

Was hängt von der Randomisierung ab?

S_i sei die Menge der vor Beginn von σ^i gespeicherten Seiten.

Die Zerlegung von σ wie auch die Mengen S_i sind unabhängig von den Münzwürfen von Marking.

- σ^1 beginnt mit dem ersten Strafpunkt.
  - Marking hat in σ^0 genau k verschiedene Seiten eingelagert.
  - σ^1 beginnt mit der Anforderung nach der $k+1$ten Seite.
- σ^i ist für $i \geq 1$ ein längstes auf σ^{i-1} folgendes Eingabeintervall, in dem genau k verschiedene Seiten angefordert werden.
  - Die Funktionsweise von Marking ist also in der Definition von σ^i ohne Belang.
- Die Menge S_i stimmt überein mit der Menge der während σ^{i-1} gespeicherten Seiten.

Alte und neue Seiten von σ^i.

Die während σ^i ($i \geq 1$) angeforderten k verschiedenen Seiten sind

- neue Anforderungen, wenn die angeforderte Seite nicht in S_i liegt,
- bzw. alte Anforderungen, wenn die angeforderte Seite in S_i liegt.

Für n_i neue Anforderungen während σ^i zeigen wir:

- Die erwartete Strafpunktzahl für Marking während σ^i ist höchstens

$$n_i \cdot \sum_{j=1}^{k} \frac{1}{j}.$$

- Jeder off-line Algorithmus erhält nach $(\sigma^0, \ldots, \sigma^m)$ mindestens

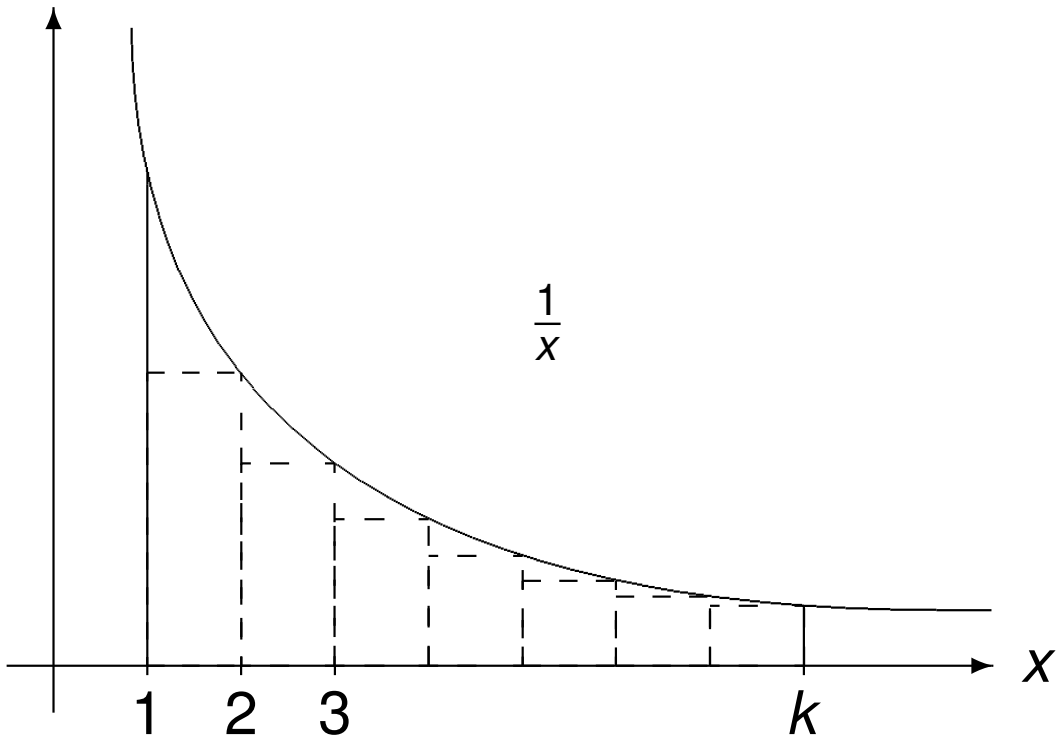
$$\frac{1}{2} \cdot \sum_{i=1}^{m} n_i \text{ Strafpunkte.}$$

Der Wettbewerbsfaktor von Marking ist $\leq 2 \cdot \sum_{j=1}^{k} \frac{1}{j}$.

Der Wettbewerbsfaktor ist logarithmisch!

$\sum_{i=1}^{k} \frac{1}{i} \leq 1 + \int_1^k \frac{1}{x} dx = \ln(k) + 1.$

Wende das Integralkriterium an:



Wir haben den Wettbwerbsfaktor von k auf $2 \cdot (\ln k + 1)$ reduziert.

Die Strafpunktzahl von Marking während σ^i

$\text{strafe}^i_{\text{alt}}$ = Erwartete Strafpunktzahl für die $k - n_i$ alten Anforderungen.

- Wann wird $\text{strafe}^i_{\text{alt}}$ maximal?
  - Zuerst Anforderungen nach neuen Seiten und dann Anforderungen nach alten Seiten oder umgekehrt?
  - Natürlich wenn neue Anforderungen zuerst kommen:
    Die Wahrscheinlichkeit wächst, dass alte Seiten zwischenzeitlich ausgelagert wurden!
- Sei alt_1 die als erste angeforderte alte Seite.
  - p = Wahrscheinlichkeit, dass alt_1 nicht ausgelagert wurde.
  - $p = \binom{k-1}{n_i} / \binom{k}{n_i} = (k - n_i)/k = 1 - n_i/k$.
  - Marking erhält einen Strafpunkt für alt_1 mit Wahrscheinlichkeit n_i/k.

Sei alt_2 die als zweite angeforderte alte Seite. Mit welcher Wahrscheinlichkeit erhält Marking einen Strafpunkt für alt_2?

Die zweite Anforderung nach einer alten Seite I

Annahme: Die Anforderung nach alt_1 war strafpunkt-frei.

- Die Seite alt_1 wurde nicht ausgelagert und ist, nach ihrer Anforderung, sicher.
- Jetzt kommen nur noch $k-1$ alte Seiten für die n_i Auslagerungen in Frage:

  $p = \binom{k-2}{n_i} / \binom{k-1}{n_i} = (k-1-n_i)/(k-1) = 1 - n_i/(k-1)$ ist die Wahrscheinlichkeit, dass alt_2 nicht ausgelagert wurde.
- Die Wahrscheinlichkeit eines Strafpunkts steigt auf $n_i/(k-1)$.

Die zweite Anforderung nach einer alten Seite II

Annahme: Die Anforderung nach alt_1 hat auf einen Strafpunkt geführt.

- alt_1 wurde ausgelagert.
  - Also gibt es $k-1$ Kandidaten für die restlichen $n_i - 1$ Anforderungen nach neuen Seiten.
- Aber alt_1 muss wieder in den Cache geholt werden:
  - Es gibt $k-1$ Kandidaten für die ersten n_i Anforderungen, die nicht zu einer Auslagerung von alt_1 führen.
  - $p = \binom{k-2}{n_i} / \binom{k-1}{n_i} = 1 - n_i/(k-1)$ ist die Wahrscheinlichkeit, dass alt_2 während der ersten n_i Anforderungen nicht ausgelagert wurde.
- Die Wahrscheinlichkeit eines Strafpunkts ist $n_i/(k-1)$ und damit unverändert.

Strafpunkte für alte Seiten

- Wir wiederholen das Argument für die anderen alten Seiten:

$$\text{strafe}^i_{\text{alt}} \leq \frac{n_i}{k} + \frac{n_i}{k-1} + \cdots + \frac{n_i}{k-(k-n_i-1)}.$$

- Die erwartete Anzahl $\text{strafe}^i_{\text{neu}}$ aller während σ^i ausgeführten neuen Anforderungen ist n_i.
- Die erwartete Strafpunktzahl während σ^i ist höchstens

$$\begin{aligned} \text{strafe}^i_{\text{alt}} + \text{strafe}^i_{\text{neu}} &\leq n_i + n_i \cdot \left(\frac{1}{k} + \frac{1}{k-1} + \cdots + \frac{1}{n_i+1} \right) \\ &\leq n_i \cdot \sum_{j=1}^{k} \frac{1}{j}. \end{aligned}$$

- Und das war zu zeigen.

Wieviele Strafpunkt erhält jede Paging Strategie?

Sei A ein off-line Algorithmus für $(\sigma^0, \sigma^1, \ldots, \sigma^m)$.

- Wir arbeiten mit einer Potenzialfunktion Φ:

  $\Phi(i) =$ Anzahl der Seiten, die von A, aber nicht von Marking zu Beginn von σ^i gespeichert sind.

- strafe$^i(A) =$ Die von A für σ^i erhaltene Strafpunktzahl.
  - Höchstens $\Phi(i)$ angeforderte neue Seiten werden von A ohne Strafpunkt abgearbeitet.
  - Also folgt strafe$^i(A) \geq n_i - \Phi(i)$.
- strafe$^i(A) \geq \Phi(i+1)$: Warum?
  - Nach Abarbeiten von σ^i besitzt A genau $\Phi(i+1)$ Seiten, die Marking nicht besitzt und umgekehrt.
  - Marking speichert diese $\Phi(i+1)$ Seiten nur, wenn sie während σ^i angefordert werden.
  - A besitzt diese Seiten nicht: Also hat A die Seiten während σ^i ausgelagert und jeweils einen Strafpunkt erhalten.

Wieviele Strafpunkt erhält jede Paging Strategie? II

Wir wissen:

$\text{strafe}^i(A) \geq n_i - \Phi(i)$ wie auch $\text{strafe}^i(A) \geq \Phi(i+1)$.

- $\max\{a,b\} \geq \frac{1}{2}(a+b)$ und deshalb ist

$$\text{strafe}^i(A) \geq \frac{1}{2}\big(n_i - \Phi(i) + \Phi(i+1)\big).$$

- Und wir sind fertig, weil

$$\begin{aligned}
\sum_{i=1}^{m} \text{strafe}^i(A) &\geq \frac{1}{2} \cdot \sum_{i=1}^{m} \big(n_i - \Phi(i) + \Phi(i+1)\big) \\
&= \left(\frac{1}{2} \cdot \sum_{i=1}^{m} n_i\right) + \frac{1}{2}\big(\Phi(m+1) - \Phi(1)\big) \\
&\geq \frac{1}{2} \cdot \sum_{i=1}^{m} n_i.
\end{aligned}$$

Wie gut können randomisierte Paging Strategien sein?

Ist ein sogar konstanter Wettbewerbsfaktor für randomisierte Strategien erreichbar?

- Der Nachweis, dass deterministische Strategien einen Wettbewerbsfaktor $\geq k$ besitzen, war einfach.
- Wir versuchen eine Reduktion der Analyse randomisierter Strategien auf die Analyse deterministischer Strategien.
- Was ist eine randomisierte Strategie R?
  - Eine Sammlung deterministischer Strategien $(D_r \mid r)$,
  - jeweils eine deterministische Strategie D_r für jede Folge r von Münzwürfen der randomisierten Strategie R.

Die Strategie

Wir versuchen zu zeigen, dass Marking (fast) optimal ist.

1. Bestimme eine möglichst schwierige Verteilung π auf den Eingabefolgen σ.
2. Bestimme eine möglichst gute untere Schranke für die erwartetete Strafpunktzahl

$$E_\pi[D] = \sum_\sigma \text{prob}_\pi[\sigma] \cdot \text{Strafe}(D(\sigma))$$

einer beliebigen deterministischen on-line Strategie D.

3. Vergleiche $E_\pi[D]$ mit der erwarteten Strafpunktzahl

$$E_\pi[\text{Opt}] = \sum_\sigma \text{prob}_\pi[\sigma] \cdot \text{Strafe}(\text{Opt}(\sigma))$$

einer optimalen Strategie.

Wenn $E_\pi[D] \geq \alpha \cdot E_\pi[\mathrm{Opt}]$ für jede deterministische Strategie D, dann hat jede randomisierte Strategie einen Wettbewerbsfaktor $\geq \alpha$.

- Die erwartete Strafpunktzahl $E_\pi[R]$ der randomisierten Strategie $R = (D_r \mid r)$ für die schwierige Verteilung π ist

$$E_\pi[R] := \sum_\sigma \mathrm{prob}_\pi[\sigma] \cdot (\sum_r p[r] \cdot \mathrm{Strafe}(D_r(\sigma))).$$

  $p[r]$ ist die Wahrscheinlichkeit der deterministischen Strategie D_r.

- Nach Summenvertauschung

$$E_\pi[R] = \sum_r p[r] \cdot \sum_\sigma \mathrm{prob}_\pi[\sigma] \cdot \mathrm{Strafe}(D_r(\sigma)) = \sum_r p[r] \cdot E_\pi[D_r].$$

- Wenn $E_\pi[D] \geq \alpha \cdot E_\pi[\mathrm{Opt}]$, dann

$$E_\pi[R] = \sum_r p[r] \cdot E_\pi[D_r] \geq \alpha \cdot E_\pi[\mathrm{Opt}].$$

Welche Verteilung π sollen wir wählen?

- Nur solche Folgen σ erhalten eine positive Wahrscheinlichkeit, die genau $k \cdot (k+1)$ Anfragen für Seiten aus $\{0, 1, \ldots, k\}$ stellen.
- Alle diese Folgen sind gleichwahrscheinlich.

- Annahme: Zu Anfang sind k dieser $k+1$ Seiten gespeichert.
- Zu jedem Zeitpunkt ist die Wahrscheinlichkeit eines Strafpunkts genau $\frac{1}{k+1}$. Für die erwartete Strafpunktzahl $E_\pi[D]$ einer deterministischer Strategie D gilt deshalb

$$E_\pi[D] \geq \frac{k \cdot (k+1)}{k+1} = k.$$

Zeige: Die erwartete Strafpunktzahl einer besten off-line Strategie ist

$$\leq k / \sum_{i=1}^{k+1} \frac{1}{i}.$$

Wie groß ist die optimale Strafpunktzahl?

Eine Anfragefolge σ der Länge $k \cdot (k+1)$
nach Seiten in $\{0, 1, \ldots, k\}$ liegt vor.

- Unterteile $\sigma = (\sigma^0, \sigma^1, \ldots)$ in Teilfolgen σ^i:

  Jede Teilfolge σ^i ist längstmöglich, so dass nur Anforderungen an k verschiedene Seiten gestellt werden.
- Eine optimale Strategie erhält in jeder Teilfolge σ^i höchstens einen Strafpunkt

  und zwar für die zu Beginn von σ^i nicht gespeicherte Seite.
- Die erwartete Strafpunktzahl für σ ist durch die erwartete Anzahl der Teilfolgen von σ beschränkt.

Wieviele Teilfolgen sind zu erwarten?

Coupon Collector

Wieviele Teilfolgen hat eine zufällige Folge σ, wenn jede Teilfolge maximal lang ist mit Anforderungen an $\leq k$ verschiedene Seiten?

- Die erwartete Anzahl der Teilfolgen stimmt überein mit

$$\frac{\text{Folgenlänge}}{\text{erwartete Länge einer Teilfolge}}.$$

- Die erwartete Länge einer Teilfolge ist die durchschnittliche Wartezeit, bis alle $k+1$ verschiedenen Seiten „gezogen“ wurden.

  Die durchschnittliche Wartezeit beträgt $(k+1) \cdot \sum_{i=1}^{k+1} \frac{1}{i}$.

Die erwartete Strafpunktzahl einer optimalen Strategie für σ ist

$$\frac{k \cdot (k+1)}{(k+1) \cdot \sum_{i=1}^{k+1} \frac{1}{i}} = \frac{k}{\sum_{i=1}^{k+1} \frac{1}{i}}.$$

Zusammenfassung

- Deterministische Paging Strategien:
  - Der Wettbewerbsfaktor ist mindestens k.
  - LRU und FIFO haben Wettbewerbsfaktor k.
- Randomisierte Paging Strategien:
  - Der Wettbewerbsfaktor ist mindestens $\sum_{i=1}^{k+1} \frac{1}{i} \approx \ln(k)$.
  - Marking hat den Wettbewerbsfaktor $2 \cdot \sum_{i=1}^{k} \frac{1}{i} \approx 2 \cdot \ln(k)$.