Advanced Algorithms

Online Algorithms

Ski-Rental Problem and Paging

Johannes Zink · WS22

\[ \underbrace{p_6, p_5, p_3}_{k} \rightarrow p_9 \]

\[ p_4, p_1, p_2, p_7, p_8, p_9 \]
Introduction

Winter has begun (even in Würzburg!) ...
Introduction

Winter has begun (even in Würzburg!) ... this means the skiing season is back!
Introduction

Winter has begun (even in Würzburg!) ... this means the skiing season is back!

But what if there is not always enough snow?
Ski-Rental Problem

Winter has begun (even in Würzburg!) . . . this means the skiing season is back!

- But what if there is not always enough snow?
- Is it worth **buying** new skis?
- Or should we rather **rent** them?
Ski-Rental Problem

Winter has begun (even in Würzburg!) . . . this means the skiing season is back!

- But what if there is not always enough snow?
- Is it worth **buying** new skis?
- Or should we rather **rent** them?
- We don’t know the weather (much) in advance.
Ski-Rental Problem – Definition

Behavior.
■ Every day when there is “good” weather, you go skiing.
  ■ We call this is a good day.
Ski-Rental Problem – Definition

Behavior.

- Every day when there is “good” weather, you go skiing.
  - We call this is a good day.
- Each morning, we can check if today is a good day, but we can’t check any earlier.
Ski-Rental Problem – Definition

Behavior.
- Every day when there is “good” weather, you go skiing.
  - We call this a good day.
- Each morning, we can check if today is a good day, but we can’t check any earlier.

Costs.
- Renting skis for 1 day costs 1 Euro.
Ski-Rental Problem – Definition

Behavior.
- Every day when there is “good” weather, you go skiing.
  - We call this a good day.
- Each morning, we can check if today is a good day, but we can’t check any earlier.

Costs.
- Renting skis for 1 day costs 1 [Euro].
- Buying skis costs \( M \) [Euros] and you have them forever.
Ski-Rental Problem – Definition

Behavior.
■ Every day when there is “good” weather, you go skiing.
  ■ We call this is a good day.
■ Each morning, we can check if today is a good day, but we can’t check any earlier.

Costs.
■ Renting skis for 1 day costs 1 [Euro].
■ Buying skis costs $M$ [Euros] and you have them forever.
■ In the end, there will have been $T$ good days.
Ski-Rental Problem – Definition

Behavior.
- Every day when there is “good” weather, you go skiing.
  - We call this a good day.
- Each morning, we can check if today is a good day, but we can’t check any earlier.

Costs.
- Renting skis for 1 day costs 1 [Euro].
- Buying skis costs $M$ [Euros] and you have them forever.
- In the end, there will have been $T$ good days.

(When to) buy skis?
Ski-Rental Problem – Definition

**Behavior.**
- Every day when there is “good” weather, you go skiing.
  - We call this is a **good** day.
- Each morning, we can check if today is a good day, but we can’t check any earlier.

**Costs.**
- Renting skis for 1 day costs 1 [Euro].
- Buying skis costs $M$ [Euros] and you have them forever.
- In the end, there will have been $T$ good days.

(When to) buy skis?

**Task.**
- Not knowing $T$, devise a strategy if and when to buy skis.
Ski-Rental Problem – Strategies I and II

Renting costs 1 per day
Buying costs \( M \) 
\( T \) good days
Ski-Rental Problem – Strategies I and II

**Strategy I: Buy** on the first good day
Ski-Rental Problem – Strategies I and II

**Strategy I: Buy** on the first good day

- Imagine this was the only good day the whole winter.
Ski-Rental Problem – Strategies I and II

**Strategy I: Buy** on the **first** good day

- Imagine this was the only good day the whole winter.
- Then we have paid $M$; optimally, we would have rented and paid 1.
Ski-Rental Problem – Strategies I and II

**Strategy I: Buy** on the **first** good day

- Imagine this was the only good day the whole winter.
- Then we have paid $M$; optimally, we would have rented and paid 1.
- So Strategy I is $M$ times worse than the optimal strategy.
Ski-Rental Problem – Strategies I and II

**Strategy I: Buy** on the **first** good day

- Imagine this was the only good day the whole winter.
- Then we have paid $M$; optimally, we would have rented and paid 1.
- So Strategy I is $M$ times worse than the optimal strategy.

→ for arbitrarily large $M$ arbitrarily bad
Ski-Rental Problem – Strategies I and II

**Strategy I: Buy** on the first good day
- Imagine this was the only good day the whole winter.
- Then we have paid $M$; optimally, we would have rented and paid 1.
- So Strategy I is $M$ times worse than the optimal strategy.

→ for arbitrarily large $M$ arbitrarily bad

**Strategy II: never buy, always rent**
Ski-Rental Problem – Strategies I and II

**Strategy I: Buy** on the first good day

- Imagine this was the only good day the whole winter.
- Then we have paid $M$; optimally, we would have rented and paid 1.
- So Strategy I is $M$ times worse than the optimal strategy.

→ for arbitrarily large $M$ arbitrarily bad

**Strategy II: never buy, always rent**

- Suppose there are many good days, i.e., $T > M$. 

Renting costs 1 per day
Buying costs $M$

$T$ good days
Renting costs 1 per day
Buying costs $M$
$T$ good days

**Strategy I: Buy** on the *first* good day

- Imagine this was the only good day the whole winter.
- Then we have paid $M$; optimally, we would have rented and paid 1.
- So **Strategy I** is $M$ times worse than the optimal strategy.

$\Rightarrow$ for arbitrarily large $M$ arbitrarily bad

**Strategy II: never buy, always rent**

- Suppose there are many good days, i.e., $T > M$.
- Then we have paid $T$.
  - Optimally, we would have bought on or before the first good day and paid $M$. 
Ski-Rental Problem – Strategies I and II

**Strategy I: Buy** on the *first* good day
- Imagine this was the only good day the whole winter.
- Then we have paid $M$; optimally, we would have rented and paid 1.
- So Strategy I is $M$ times worse than the optimal strategy.

→ for arbitrarily large $M$ arbitrarily bad

**Strategy II: never buy, always rent**
- Suppose there are many good days, i.e., $T > M$.
- Then we have paid $T$.
  - Optimally, we would have bought on or before the first good day and paid $M$.
- Strategy II is $T/M$ times worse than the optimal strategy.
Ski-Rental Problem – Strategies I and II

**Strategy I: Buy** on the **first** good day

- Imagine this was the only good day the whole winter.
- Then we have paid $M$; optimally, we would have rented and paid 1.
- So Strategy I is $M$ times worse than the optimal strategy.

$\rightarrow$ for arbitrarily large $M$ arbitrarily bad

**Strategy II: never buy, always rent**

- Suppose there are many good days, i.e., $T > M$.
- Then we have paid $T$.
  
  Optimally, we would have bought on or before the first good day and paid $M$.
- Strategy II is $T/M$ times worse than the optimal strategy.

$\rightarrow$ for arbitrarily large $T$ arbitrarily bad
Ski-Rental Problem – Strategies I and II

**Strategy I: Buy** on the *first* good day

- Imagine this was the only good day the whole winter.
- Then we have paid $M$; optimally, we would have rented and paid 1.
- So Strategy I is $M$ times worse than the optimal strategy.

**Strategy II: never buy, always rent**

- Suppose there are many good days, i.e., $T > M$.
- Then we have paid $T$.
  Optimally, we would have bought on or before the first good day and paid $M$.
- Strategy II is $T/M$ times worse than the optimal strategy.

→ for arbitrarily large $M$ arbitrarily bad

→ for arbitrarily large $T$ arbitrarily bad
Ski-Rental Problem – Strategy III

Renting costs 1 per day
Buying costs $M_T$ good days
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad?

Renting costs 1 per day
Buying costs $M$
$T$ good days
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

Renting costs 1 per day
Buying costs $M_T$ good days
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

**Strategy III**: buy on the $M$-th good day
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

**Strategy III:** buy on the $M$-th good day

- Observation: the optimal solution pays $\min(M, T)$
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

**Strategy III:** buy on the \( M \)-th good day

- Observation: the optimal solution pays \( \min(M, T) \)
- If \( T < M \), the competitive ratio is 1.
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

**Strategy III:** buy on the $M$-th good day

- Observation: the optimal solution pays $\min(M, T)$
- If $T < M$, the competitive ratio is 1. Otherwise, it is $\frac{2M-1}{M} = 2 - \frac{1}{M}$
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

**Strategy III**: buy on the \( M \)-th good day

- Observation: the optimal solution pays \( \min(M, T) \)
- If \( T < M \), the competitive ratio is 1. Otherwise, it is \( \frac{2M-1}{M} = 2 - \frac{1}{M} \overset{M \to \infty}{\to} 2 \).
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

**Strategy III:** buy on the $M$-th good day

- Observation: the optimal solution pays $\min(M, T)$
- If $T < M$, the competitive ratio is 1. Otherwise, it is $\frac{2M - 1}{M} = 2 - \frac{1}{M}$ as $M \to \infty$.

$\Rightarrow$ Strategy III is deterministic and 2-competitive.
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

**Strategy III:** buy on the $M$-th good day

- Observation: the optimal solution pays $\min(M, T)$
- If $T < M$, the competitive ratio is 1. Otherwise, it is $\frac{2M-1}{M} = 2 - \frac{1}{M}$.

$\Rightarrow$ Strategy III is deterministic and 2-competitive.

**Theorem 1.** No det. strategy is better than 2-competitive (for $M \mapsto \infty$; in general: $2 - \frac{1}{M}$).
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

**Strategy III**: buy on the $M$-th good day

- Observation: the optimal solution pays $\min(M, T)$
- If $T < M$, the competitive ratio is 1. Otherwise, it is \( \frac{2M-1}{M} = 2 - \frac{1}{M} \) $\Rightarrow$ Strategy III is deterministic and 2-competitive.

**Theorem 1.** No det. strategy is better than 2-competitive (for $M \to \infty$; in general: $2 - \frac{1}{M}$).

**Proof Idea.**
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

**Strategy III:** buy on the $M$-th good day

- Observation: the optimal solution pays $\min(M, T)$
- If $T < M$, the competitive ratio is 1. Otherwise, it is $\frac{2M-1}{M} = 2 - \frac{1}{M}$.

$\Rightarrow$ Strategy III is deterministic and 2-competitive.

**Theorem 1.** No det. strategy is better than 2-competitive (for $M \to \infty$; in general: $2 - \frac{1}{M}$).

**Proof Idea.**

- Any det. strategy can be formulated as “buy on the $X$-th day of rental” for a fixed $X$. 

Renting costs 1 per day
Buying costs $M$ good days
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

**Strategy III:** buy on the $M$-th good day

- Observation: the optimal solution pays $\min(M, T)$
- If $T < M$, the competitive ratio is 1. Otherwise, it is $\frac{2M-1}{M} = 2 - \frac{1}{M}$.

$\Rightarrow$ Strategy III is deterministic and 2-competitive.

**Theorem 1.** No det. strategy is better than 2-competitive (for $M \sim \infty$; in general: $2 - \frac{1}{M}$).

**Proof Idea.**

- Any det. strategy can be formulated as “buy on the $X$-th day of rental” for a fixed $X$.
- For $X = 0$ and $X = \infty$ it’s arbitrarily bad; assume $X \in \mathbb{N}^+$. Observe, w.c. is $T = X$. 
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

**Strategy III:** buy on the \( M \)-th good day

- Observation: the optimal solution pays \( \min(M, T) \)
- If \( T < M \), the competitive ratio is 1. Otherwise, it is \( \frac{2M-1}{M} = 2 - \frac{1}{M} \quad \text{as } M \to \infty \).
  \[ \Rightarrow \text{Strategy III is deterministic and 2-competitive.} \]

**Theorem 1.** No det. strategy is better than 2-competitive (for \( M \to \infty \); in general: \( 2 - \frac{1}{M} \)).

**Proof Idea.**
- Any det. strategy can be formulated as “buy on the \( X \)-th day of rental” for a fixed \( X \).
- For \( X = 0 \) and \( X = \infty \) it’s arbitrarily bad; assume \( X \in \mathbb{N}^+ \). Observe, w.c. is \( T = X \).
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

**Strategy III:** buy on the $M$-th good day

- Observation: the optimal solution pays $\min(M, T)$
- If $T < M$, the competitive ratio is 1. Otherwise, it is $\frac{2M-1}{M} = 2 - \frac{1}{M}$ $M \to \infty$ 2.

$\Rightarrow$ Strategy III is deterministic and 2-competitive.

**Theorem 1.** No det. strategy is better than 2-competitive (for $M \to \infty$; in general: $2 - \frac{1}{M}$).

**Proof Idea.**

- Any det. strategy can be formulated as “buy on the $X$-th day of rental” for a fixed $X$.
- For $X = 0$ and $X = \infty$ it’s arbitrarily bad; assume $X \in \mathbb{N}^+$. Observe, w.c. is $T = X$.

$$\frac{c_{\text{det}}}{c_{\text{OPT}}} = \frac{X-1+M}{\min(X,M)}$$
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

**Strategy III:** buy on the $M$-th good day

- Observation: the optimal solution pays $\min(M, T)$
- If $T < M$, the competitive ratio is 1. Otherwise, it is $\frac{2M-1}{M} = 2 - \frac{1}{M}$. $M \xrightarrow{\sim} \infty \Rightarrow$ Strategy III is deterministic and 2-competitive.

**Theorem 1.** No det. strategy is better than 2-competitive (for $M \xrightarrow{\sim} \infty$; in general: $2 - \frac{1}{M}$).

**Proof Idea.**

- Any det. strategy can be formulated as “buy on the $X$-th day of rental” for a fixed $X$.
- For $X = 0$ and $X = \infty$ it’s arbitrarily bad; assume $X \in \mathbb{N}^+$. Observe, w.c. is $T = X$.

\[
\frac{c_{\text{det}}}{c_{\text{OPT}}} = \frac{X-1+M}{\min(X,M)} \geq \min(\begin{cases} \frac{X-1+X+1}{X} & \text{case } X < M \\ \frac{M-1+M}{M} & \text{case } M \leq X \end{cases})
\]
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

**Strategy III**: buy on the $M$-th good day

- Observation: the optimal solution pays $\min(M, T)$
- If $T < M$, the competitive ratio is 1. Otherwise, it is $\frac{2M-1}{M} = 2 - \frac{1}{M}$ $\xrightarrow{M \to \infty} 2$.

$\Rightarrow$ Strategy III is deterministic and 2-competitive.

**Theorem 1.** No det. strategy is better than 2-competitive (for $M \xrightarrow{} \infty$; in general: $2 - \frac{1}{M}$).

**Proof Idea.**

- Any det. strategy can be formulated as “buy on the $X$-th day of rental” for a fixed $X$.
- For $X = 0$ and $X = \infty$ it’s arbitrarily bad; assume $X \in \mathbb{N}^+$. Observe, w.c. is $T = X$.

$$\frac{c_{\text{det}}}{c_{\text{OPT}}} = \frac{X-1+M}{\min(X,M)} \geq \min \left( \frac{X-1+X+1}{X}, \frac{M-1+M}{M} \right) = \min \left( 2, 2 - \frac{1}{M} \right) = 2 - \frac{1}{M}$$

Renting costs 1 per day
Buying costs $M$ $T$ good days
Ski-Rental Problem – Strategy III

Is there a strategy that cannot become arbitrarily bad? – Yes!

**Strategy III:** buy on the $M$-th good day

- Observation: the optimal solution pays $\min(M, T)$
- If $T < M$, the competitive ratio is 1. Otherwise, it is $2 - \frac{1}{M}$.

$\Rightarrow$ Strategy III is deterministic and 2-competitive.

**Theorem 1.** No det. strategy is better than 2-competitive (for $M \hookrightarrow \infty$; in general: $2 - \frac{1}{M}$).

**Proof Idea.**

- Any det. strategy can be formulated as “buy on the $X$-th day of rental” for a fixed $X$.
- For $X = 0$ and $X = \infty$ it’s arbitrarily bad; assume $X \in \mathbb{N}^+$. Observe, w.c. is $T = X$.

$$
\frac{c_{\text{det}}}{c_{\text{OPT}}} = \frac{X-1+M}{\min(X,M)} \geq \min \left( \frac{X-1+X+1}{X}, \frac{M-1+M}{M} \right) = \min \left( 2, 2 - \frac{1}{M} \right) = 2 - \frac{1}{M} \hookrightarrow \infty = 2
$$
Ski-Rental Problem – Strategy IV

Can we get below this bound using randomization?
Ski-Rental Problem – Strategy IV

Can we get below this bound using randomization? – Let’s try!
Ski-Rental Problem – Strategy IV

Can we get below this bound using randomization? – Let’s try!

**Strategy IV**: throw a coin; **HEAD**: buy on the $M$-th good day
**TAIL**: buy on the $\alpha M$-th good day ($\alpha \in (0, 1)$)
Ski-Rental Problem – Strategy IV

Can we get below this bound using randomization? – Let’s try!

**Strategy IV:** throw a coin; **HEAD:** buy on the $M$-th good day  
**TAIL:** buy on the $\alpha M$-th good day ($\alpha \in (0, 1)$)

- Observation: worst case can only be $T = M$ or $T = \alpha M$
Ski-Rental Problem – Strategy IV

Can we get below this bound using randomization? – Let’s try!

**Strategy IV:** throw a coin; **HEAD:** buy on the \( M \)-th good day

**TAIL:** buy on the \( \alpha M \)-th good day (\( \alpha \in (0, 1) \))

- Observation: worst case can only be \( T = M \) or \( T = \alpha M \)

- Case \( T = M \):
  \[
  \frac{E[c_{\text{Strategy IV}}]}{c_{\text{OPT}}} = \frac{1}{2} \cdot (2M-1) + \frac{1}{2} \cdot ((1+\alpha)M-1) = \frac{3+\alpha}{2} - \frac{1}{M} \quad M \rightarrow \infty \quad \Rightarrow \quad \frac{3+\alpha}{2}
  \]
Ski-Rental Problem – Strategy IV

Can we get below this bound using randomization? – Let’s try!

**Strategy IV:** throw a coin; **HEAD:** buy on the $M$-th good day

**TAIL:** buy on the $\alpha M$-th good day ($\alpha \in (0, 1)$)

- Observation: worst case can only be $T = M$ or $T = \alpha M$

- **Case $T = M$:** 
  \[
  \frac{E[c_{\text{Strategy IV}}]}{c_{\text{OPT}}} = \frac{1}{2} \cdot (2M-1) + \frac{1}{2} \cdot ((1+\alpha)M-1) = \frac{3+\alpha}{2} - \frac{1}{M} \Rightarrow M \xrightarrow{\infty} \frac{3+\alpha}{2}
  \]

- **Case $T = \alpha M$:** 
  \[
  \frac{E[c_{\text{Strategy IV}}]}{c_{\text{OPT}}} = \frac{1}{2} \cdot \alpha M + \frac{1}{2} \cdot ((1+\alpha)M-1) = 1 + \frac{1}{2\alpha} - \frac{1}{2\alpha M} \Rightarrow M \xrightarrow{\infty} 1 + \frac{1}{2\alpha}
  \]
Ski-Rental Problem – Strategy IV

Can we get below this bound using randomization? – Let’s try!

**Strategy IV:** throw a coin; **HEAD:** buy on the $M$-th good day
**TAIL:** buy on the $\alpha M$-th good day ($\alpha \in (0, 1)$)

- Observation: worst case can only be $T = M$ or $T = \alpha M$

- Case $T = M$: $\frac{E[c_{StrategyIV}]}{c_{OPT}} = \frac{1}{2} \cdot (2M-1) + \frac{1}{2} \cdot ((1+\alpha)M-1) = \frac{3+\alpha}{2} - \frac{1}{M} \overset{M \to \infty}{\longrightarrow} \frac{3+\alpha}{2}$

- Case $T = \alpha M$: $\frac{E[c_{StrategyIV}]}{c_{OPT}} = \frac{1}{2} \cdot \alpha M + \frac{1}{2} \cdot ((1+\alpha)M-1) = 1 + \frac{1}{2\alpha} - \frac{1}{2\alpha M} \overset{M \to \infty}{\longrightarrow} 1 + \frac{1}{2\alpha}$

---

Renting costs 1 per day
Buying costs $M$ $T$ good days

try $\alpha = \frac{1}{2}$
Ski-Rental Problem – Strategy IV

Can we get below this bound using randomization? – Let’s try!

**Strategy IV:** throw a coin; **HEAD:** buy on the $M$-th good day
**TAIL:** buy on the $\alpha M$-th good day ($\alpha \in (0, 1)$)

- **Observation:** worst case can only be $T = M$ or $T = \alpha M$

- **Case** $T = M$: 
  $$E\left[\frac{c_{\text{Strategy IV}}}{c_{\text{OPT}}}\right] = \frac{1}{2} \cdot (2M-1) + \frac{1}{2} \cdot ((1+\alpha)M-1) = \frac{3+\alpha}{2} - \frac{1}{M} \xrightarrow{M \to \infty} \frac{3+\alpha}{2} = \frac{7}{4} < 2$$

- **Case** $T = \alpha M$: 
  $$E\left[\frac{c_{\text{Strategy IV}}}{c_{\text{OPT}}}\right] = \frac{1}{2} \cdot \alpha M + \frac{1}{2} \cdot ((1+\alpha)M-1) = 1 + \frac{1}{2\alpha} - \frac{1}{2\alpha M} \xrightarrow{M \to \infty} 1 + \frac{1}{2\alpha}$$

**Try** $\alpha = \frac{1}{2}$
Ski-Rental Problem – Strategy IV

Can we get below this bound using randomization? – Let’s try!

**Strategy IV:** throw a coin; **HEAD:** buy on the $M$-th good day
**TAIL:** buy on the $\alpha M$-th good day ($\alpha \in (0, 1)$)

- **Observation:** worst case can only be $T = M$ or $T = \alpha M$
- **Case** $T = M$: $E[c_{\text{Strategy IV}}] / c_{\text{OPT}} = \frac{1}{2} \cdot (2M-1) + \frac{1}{2} \cdot \frac{(1+\alpha)M-1}{M} = \frac{3+\alpha}{2} - \frac{1}{M} \xrightarrow{M\to\infty} \frac{3+\alpha}{2} = \frac{7}{4} < 2$
- **Case** $T = \alpha M$: $E[c_{\text{Strategy IV}}] / c_{\text{OPT}} = \frac{1}{2} \cdot \alpha M + \frac{1}{2} \cdot \frac{(1+\alpha)M-1}{\alpha M} = 1 + \frac{1}{2\alpha} - \frac{1}{2\alpha M} \xrightarrow{M\to\infty} 1 + \frac{1}{2\alpha} = 2$

Renting costs 1 per day
Buying costs $M$ good days
Ski-Rental Problem – Strategy IV

Can we get below this bound using randomization? – Let’s try!

**Strategy IV:** throw a coin; **HEAD:** buy on the $M$-th good day  
**TAIL:** buy on the $\alpha M$-th good day ($\alpha \in (0, 1)$)

- Observation: worst case can only be $T = M$ or $T = \alpha M$
  
  - **Case $T = M$:** $E_{c_{\text{Strategy IV}}}/c_{\text{OPT}} = \frac{1}{2} (2M-1) + \frac{1}{2} \cdot ((1+\alpha)M-1) = \frac{3+\alpha}{2} - \frac{1}{M}$ $\lim_{M \to \infty} \frac{3+\alpha}{2} = \frac{7}{4} < 2$

  - **Case $T = \alpha M$:** $E_{c_{\text{Strategy IV}}}/c_{\text{OPT}} = \frac{1}{2} \cdot \alpha M + \frac{1}{\alpha M} \cdot ((1+\alpha)M-1) = 1 + \frac{1}{2\alpha} - \frac{1}{2\alpha M} \approx \frac{3+\alpha}{2} \approx 2$

  not better than the deterministic Strategy III
Ski-Rental Problem – Strategy IV

Can we get below this bound using randomization? – Let’s try!

**Strategy IV:** throw a coin; **HEAD:** buy on the $\text{M}$-th good day
**TAIL:** buy on the $\alpha \text{M}$-th good day ($\alpha \in (0, 1)$)

- Observation: worst case can only be $T = \text{M}$ or $T = \alpha \text{M}$

- Case $T = \text{M}$:  
  $$E\left[\frac{c_{\text{Strategy IV}}}{c_{\text{OPT}}}\right] = \frac{\frac{1}{2}(2\text{M} - 1) + \frac{1}{2}((1 + \alpha)\text{M} - 1)}{\text{M}} = \frac{3 + \alpha}{2} - \frac{1}{\text{M}} \xrightarrow{\text{M} \to \infty} \frac{3 + \alpha}{2}$$

- Case $T = \alpha \text{M}$:  
  $$E\left[\frac{c_{\text{Strategy IV}}}{c_{\text{OPT}}}\right] = \frac{\frac{1}{2}\alpha \text{M} + \frac{1}{2}((1 + \alpha)\text{M} - 1)}{\alpha \text{M}} = 1 + \frac{1}{2\alpha} - \frac{1}{2\alpha \text{M}} \xrightarrow{\text{M} \to \infty} 1 + \frac{1}{2\alpha}$$

- The w.c. ratio is minimum if $\frac{3 + \alpha}{2} = 1 + \frac{1}{2\alpha}$
Ski-Rental Problem – Strategy IV

Can we get below this bound using randomization? – Let’s try!

**Strategy IV:** throw a coin; **HEAD:** buy on the $M$-th good day
**TAIL:** buy on the $\alpha M$-th good day ($\alpha \in (0, 1)$)

- Observation: worst case can only be $T = M$ or $T = \alpha M$

- Case $T = M$: 
  \[
  \frac{E[c_{\text{Strategy IV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2} \cdot (2M-1) + \frac{1}{2} \cdot ((1+\alpha)M-1)}{M} = \frac{3+\alpha}{2} - \frac{1}{M} \xrightarrow{M \to \infty} \frac{3+\alpha}{2}
  \]

- Case $T = \alpha M$: 
  \[
  \frac{E[c_{\text{Strategy IV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2} \cdot \alpha M + \frac{1}{2} \cdot ((1+\alpha)M-1)}{\alpha M} = 1 + \frac{1}{2\alpha} - \frac{1}{2\alpha M} \xrightarrow{M \to \infty} 1 + \frac{1}{2\alpha}
  \]

- The w.c. ratio is minimum if 
  \[
  \frac{3+\alpha}{2} = 1 + \frac{1}{2\alpha} \Rightarrow \alpha = \frac{\sqrt{5} - 1}{2}
  \]

Renting costs 1 per day
Buying costs $M$ $T$ good days
Ski-Rental Problem – Strategy IV

Can we get below this bound using randomization? – Let’s try!

**Strategy IV:** throw a coin; **HEAD:** buy on the $M$-th good day
**TAIL:** buy on the $\alpha M$-th good day ($\alpha \in (0, 1)$)

- **Observation:** worst case can only be $T = M$ or $T = \alpha M$

- **Case** $T = M$: 
  \[
  E\left[\frac{c_{\text{Strategy IV}}}{c_{\text{OPT}}}\right] = \frac{1}{2} \cdot (2M - 1) + \frac{1}{2} \cdot ((1 + \alpha)M - 1) = \frac{3 + \alpha}{2} - \frac{1}{M} \quad M \to \infty \Rightarrow \frac{3 + \alpha}{2}
  \]

- **Case** $T = \alpha M$: 
  \[
  E\left[\frac{c_{\text{Strategy IV}}}{c_{\text{OPT}}}\right] = \frac{1}{2} \cdot \alpha M + \frac{1}{2} \cdot ((1 + \alpha)M - 1) = 1 + \frac{1}{2\alpha} - \frac{1}{2\alpha M} \quad M \to \infty \Rightarrow 1 + \frac{1}{2\alpha}
  \]

- **The w.c. ratio is minimum** if \[\frac{3 + \alpha}{2} = 1 + \frac{1}{2\alpha} \Rightarrow \alpha = \frac{\sqrt{5} - 1}{2}\]

⇒ **Strategy IV** (with $\alpha = \frac{\sqrt{5} - 1}{2} \approx 0.62$) is 1.81-competitive, randomized, and better than any deterministic strategy.
Ski-Rental Problem – Strategy IV

Can we get below this bound using randomization? – Let’s try!

**Strategy IV:** throw a coin; **HEAD:** buy on the $M$-th good day
**TAIL:** buy on the $\alpha M$-th good day ($\alpha \in (0, 1)$)

- Observation: worst case can only be $T = M$ or $T = \alpha M$
- Case $T = M$: $\frac{E[c_{\text{Strategy IV}}]}{c_{\text{OPT}}} = \frac{1}{2} \cdot (2M-1) + \frac{1}{2} \cdot ((1+\alpha)M-1) = \frac{3+\alpha}{2} - \frac{1}{M} \xrightarrow{M \to \infty} \frac{3+\alpha}{2}$
- Case $T = \alpha M$: $\frac{E[c_{\text{Strategy IV}}]}{c_{\text{OPT}}} = \frac{1}{2} \cdot \alpha M + \frac{1}{2} \cdot ((1+\alpha)M-1) = 1 + \frac{1}{2\alpha} - \frac{1}{2\alpha M} \xrightarrow{M \to \infty} 1 + \frac{1}{2\alpha}$
- The w.c. ratio is minimum if $\frac{3+\alpha}{2} = 1 + \frac{1}{2\alpha} \Rightarrow \alpha = \frac{\sqrt{5}-1}{2}$

$\Rightarrow$ Strategy IV (with $\alpha = \frac{\sqrt{5}-1}{2} \approx 0.62$) is 1.81-competitive, randomized, and better than any deterministic strategy.

- With a more sophisticated probability distribution for the time we buy skis, we can expect even a competitive ratio of $\frac{e}{e-1} \approx 1.58$. 

Renting costs 1 per day
Buying costs $M$
$T$ good days
Online vs. Offline Algorithms
Online vs. Offline Algorithms

Online Algorithm
Online vs. Offline Algorithms

**Online Algorithm**

- No full information available initially
  
  *(online problem)*
Online vs. Offline Algorithms

**Online Algorithm**

- No full information available initially (*online problem*)
- Decisions are made with incomplete information.
Online vs. Offline Algorithms

**Online Algorithm**

- No full information available initially (online problem)
- Decisions are made with incomplete information.
- The algorithm may get more informations over time or by exploring the instance.
Online vs. Offline Algorithms

**Online Algorithm**
- No full information available initially (*online problem*)
- Decisions are made with incomplete information.
- The algorithm may get more informations over time or by exploring the instance.

**Offline Algorithm**
Online vs. Offline Algorithms

**Online Algorithm**
- No full information available initially (*online problem*)
- Decisions are made with incomplete information.
- The algorithm may get more informations over time or by exploring the instance.

**Offline Algorithm**
- Full information available initially (*offline problem*)
Online vs. Offline Algorithms

**Online Algorithm**
- No full information available initially (*online problem*)
- Decisions are made with incomplete information.
- The algorithm may get more informations over time or by exploring the instance.

**Offline Algorithm**
- Full information available initially (*offline problem*)
- Decisions are made with complete information.
Online vs. Offline Algorithms

**Online Algorithm**
- No full information available initially (online problem)
- Decisions are made with incomplete information.
- The algorithm may get more informations over time or by exploring the instance.

**Offline Algorithm**
- Full information available initally (offline problem)
- Decisions are made with complete information.

- The objective value of the returned solution divided by the objective value of an optimal [offline] solution is the *competitive ratio*. 
# Online vs. Offline Algorithms

**Online Algorithm**
- No full information available initially (*online problem*)
- Decisions are made with incomplete information.
- The algorithm may get more informations over time or by exploring the instance.

**Offline Algorithm**
- Full information available initially (*offline problem*)
- Decisions are made with complete information.

The objective value of the returned solution divided by the objective value of an optimal [offline] solution is the *competitive ratio*. 
# Online vs. Offline Algorithms

### Online Algorithm
- No full information available initially (*online problem*)
- Decisions are made with incomplete information.
- The algorithm may get more informations over time or by exploring the instance.

### Offline Algorithm
- Full information available initially (*offline problem*)
- Decisions are made with complete information.

The objective value of the returned solution divided by the objective value of an optimal [offline] solution is the *competitive ratio*.
Online vs. Offline Algorithms

**Online Algorithm**
- No full information available initially (*online problem*)
- Decisions are made with incomplete information.
- The algorithm may get more informations over time or by exploring the instance.

**Offline Algorithm**
- Full information available initially (*offline problem*)
- Decisions are made with complete information.
- The objective value of the returned solution divided by the objective value of an optimal [offline] solution is the *competitive ratio*.
Online vs. Offline Algorithms

**Online Algorithm**
- No full information available initially (*online problem*)
- Decisions are made with incomplete information.
- The algorithm may get more informations over time or by exploring the instance.

**Offline Algorithm**
- Full information available initially (*offline problem*)
- Decisions are made with complete information.
- The objective value of the returned solution divided by the objective value of an optimal [offline] solution is the *competitive ratio*.

Examples (problems & algos.):
- Ski-Rental Problem, searching in unknown environments, Cow-Path Problem, Job-Shop Scheduling, Insertion Sort, Paging (replacing entries in a memory).
Online vs. Offline Algorithms

**Online Algorithm**
- No full information available initially (*online problem*)
- Decisions are made with incomplete information.
- The algorithm may get more informations over time or by exploring the instance.

**Offline Algorithm**
- Full information available initially (*offline problem*)
- Decisions are made with complete information.
- The objective value of the returned solution divided by the objective value of an optimal [offline] solution is the *competitive ratio*.

Examples (problems & algos.):
Ski-Rental Problem, searching in unkown environments, Cow-Path Problem, Job-Shop Scheduling, Insertion Sort, Paging (replacing entries in a memory)
Online vs. Offline Algorithms

**Online Algorithm**
- No full information available initially (*online problem*)
- Decisions are made with incomplete information.
- The algorithm may get more informations over time or by exploring the instance.

**Offline Algorithm**
- Full information available initially (*offline problem*)
- Decisions are made with complete information.

The objective value of the returned solution divided by the objective value of an optimal [offline] solution is the *competitive ratio*.

Examples (problems & algos.):
- Ski-Rental Problem, searching in unknown environments, Cow-Path Problem, Job-Shop Scheduling, Insertion Sort, Paging (replacing entries in a memory)
Paging – Definition

Given (offline/online):
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of $k$ pages
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of $k$ pages
- Slow access memory with unlimited capacity
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of $k$ pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (page fault), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of $k$ pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (page fault), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of $k$ pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (page fault), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of $k$ pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (page fault), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of $k$ pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (page fault), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of \( k \) pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (page fault), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
- Sequence \( \sigma \) of page requests that need to be fulfilled in order. / We have to fulfill a request before we see the next request.
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of \( k \) pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (page fault), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
- Sequence \( \sigma \) of page requests that need to be fulfilled in order. / We have to fulfill a request before we see the next request.
Paging – Definition

**Given (offline/online):**

- Fast access memory (a cache) with a capacity of \( k \) pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (*page fault*), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
- Sequence \( \sigma \) of page requests that need to be fulfilled in order. / We have to fulfill a request before we see the next request.
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of $k$ pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (page fault), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
- Sequence $\sigma$ of page requests that need to be fulfilled in order. / We have to fulfill a request before we see the next request.
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of \( k \) pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (page fault), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
- Sequence \( \sigma \) of page requests that need to be fulfilled in order. We have to fulfill a request before we see the next request.
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of $k$ pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (page fault), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
- Sequence $\sigma$ of page requests that need to be fulfilled in order. / We have to fulfill a request before we see the next request.
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of \( k \) pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (page fault), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
- Sequence \( \sigma \) of page requests that need to be fulfilled in order. We have to fulfill a request before we see the next request.
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of $k$ pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (page fault), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
- Sequence $\sigma$ of page requests that need to be fulfilled in order. / We have to fulfill a request before we see the next request.
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of $k$ pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (page fault), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
- Sequence $\sigma$ of page requests that need to be fulfilled in order. / We have to fulfill a request before we see the next request.

Objective value:
Paging – Definition

Given (offline/online):

- Fast access memory (a cache) with a capacity of $k$ pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (*page fault*), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
- Sequence $\sigma$ of page requests that need to be fulfilled in order. We have to fulfill a request before we see the next request.

Objective value:

- Minimize the number of page faults while fulfilling $\sigma$. 


- On a page fault, a Paging algorithm chooses which page to evict from the cache.
On a page fault, a Paging algorithm chooses which page to evict from the cache.
On a page fault, a Paging algorithm chooses which page to evict from the cache.

On a page fault, a Paging algorithm chooses which page to evict from the cache.

**Deterministic Strategies:** Evict the page that has . . .
On a page fault, a Paging algorithm chooses which page to evict from the cache.

**Deterministic Strategies:** Evict the page that has . . .

- Least Frequently Used (LFU): . . . the lowest number of accesses since it was loaded.
On a page fault, a Paging algorithm chooses which page to evict from the cache.

**Deterministic Strategies:** Evict the page that has . . .

- Least Frequently Used (LFU): . . . the lowest number of accesses since it was loaded.
On a page fault, a Paging algorithm chooses which page to evict from the cache.

**Deterministic Strategies:** Evict the page that has . . .

- Least Frequently Used (LFU): . . . the lowest number of accesses since it was loaded.
- Least Recently Used (LRU): . . . been accessed least recently.
On a page fault, a Paging algorithm chooses which page to evict from the cache.

**Deterministic Strategies:** Evict the page that has . . .

- Least Frequently Used (LFU): . . . the lowest number of accesses since it was loaded.
- Least Recently Used (LRU): . . . been accessed least recently.
On a page fault, a Paging algorithm chooses which page to evict from the cache.

**Deterministic Strategies:** Evict the page that has . . .

- Least Frequently Used (LFU): . . . the lowest number of accesses since it was loaded.
- Least Recently Used (LRU): . . . been accessed least recently.
- First-in-first-out (FIFO): . . . been in cache the longest.
On a page fault, a Paging algorithm chooses which page to evict from the cache.

**Deterministic Strategies:** Evict the page that has . . .

- Least Frequently Used (LFU): . . . the lowest number of accesses since it was loaded.
- Least Recently Used (LRU): . . . been accessed least recently.
- First-in-first-out (FIFO): . . . been in cache the longest.

- On a page fault, a Paging algorithm chooses which page to evict from the cache.

**Deterministic Strategies:** Evict the page that has...
- Least Frequently Used (LFU): ...the lowest number of accesses since it was loaded.
- Least Recently Used (LRU): ...been accessed least recently.
- First-in-first-out (FIFO): ...been in cache the longest.

Which of them is – theoretically provable – the best strategy?
On a page fault, a Paging algorithm chooses which page to evict from the cache.

**Deterministic Strategies:** Evict the page that has . . .

- Least Frequently Used (LFU): . . . the lowest number of accesses since it was loaded.
- Least Recently Used (LRU): . . . been accessed least recently.
- First-in-first-out (FIFO): . . . been in cache the longest.

**Which of them is – theoretically provable – the best strategy?**

\[ \text{Theorem 2. LRU & FIFO are } k\text{-competitive. No deterministic strategy is better.} \]
Paging – Det. strategies Analysis

Theorem 2. LRU & FIFO are $k$-competitive. No deterministic strategy is better.
Paging – Det. strategies Analysis

**Theorem 2.** LRU & FIFO are $k$-competitive. No deterministic strategy is better.

**Proof.** (only for LRU, FIFO similar)
Theorem 2. LRU & FIFO are $k$-competitive. No deterministic strategy is better.

Proof. (only for LRU, FIFO similar)

MIN: optimal strategy
\( \sigma \): sequence of pages
Theorem 2. LRU & FIFO are $k$-competitive. No deterministic strategy is better.

Proof. (only for LRU, FIFO similar)

- Initially, the cache contains the same pages for all strategies.

MIN: optimal strategy

$\sigma$: sequence of pages
Paging – Det. strategies Analysis

**Theorem 2.** LRU & FIFO are $k$-competitive. No deterministic strategy is better.

**Proof.** (only for LRU, FIFO similar)

- Initially, the cache contains the same pages for all strategies.

- We partition $\sigma$ into phases $P_0, P_1, \ldots$, s.t. LRU has at most $k$ faults in $P_0$ and exactly $k$ faults in each other phase.

MIN: optimal strategy $\sigma$: sequence of pages
Paging – Det. strategies Analysis

**Theorem 2.** LRU & FIFO are $k$-competitive. No deterministic strategy is better.

**Proof.** (only for LRU, FIFO similar)

- Initially, the cache contains the same pages for all strategies.
- We partition $\sigma$ into phases $P_0, P_1, \ldots$, s.t. LRU has at most $k$ faults in $P_0$ and exactly $k$ faults in each other phase.
- We show next: MIN has at least 1 fault in each phase.

MIN: optimal strategy

$\sigma$: sequence of pages
Paging – Det. strategies Analysis

**Theorem 2.** LRU & FIFO are $k$-competitive. No deterministic strategy is better.

**Proof.** (only for LRU, FIFO similar)

- Initially, the cache contains the same pages for all strategies.
- We partition $\sigma$ into phases $P_0, P_1, \ldots$, s.t. LRU has at most $k$ faults in $P_0$ and exactly $k$ faults in each other phase.
- We show next: MIN has at least 1 fault in each phase.
- Clearly, MIN also faults in $P_0$; consider $P_i$ ($i \geq 1$) and let $p$ be the last page of $P_{i-1}$.

MIN: optimal strategy
$\sigma$: sequence of pages
Paging – Det. strategies Analysis

**Theorem 2.** LRU & FIFO are $k$-competitive. No deterministic strategy is better.

**Proof.** (only for LRU, FIFO similar)

- Initially, the cache contains the same pages for all strategies.
- We partition $\sigma$ into phases $P_0, P_1, \ldots$, s.t. LRU has at most $k$ faults in $P_0$ and exactly $k$ faults in each other phase.
- We show next: MIN has at least 1 fault in each phase.
- Clearly, MIN also faults in $P_0$; consider $P_i$ ($i \geq 1$) and let $p$ be the last page of $P_{i-1}$.
- Show: $P_i$ contains $k$ distinct page requests different from $p$ (implies a fault for MIN).
Paging – Det. strategies Analysis

**Theorem 2.** LRU & FIFO are $k$-competitive. No deterministic strategy is better.

**Proof.** (only for LRU, FIFO similar)

- Initially, the cache contains the same pages for all strategies.
- We partition $\sigma$ into phases $P_0, P_1, \ldots$, s.t. LRU has at most $k$ faults in $P_0$ and exactly $k$ faults in each other phase.
- We show next: MIN has at least 1 fault in each phase.
- Clearly, MIN also faults in $P_0$; consider $P_i$ ($i \geq 1$) and let $p$ be the last page of $P_{i-1}$.
- Show: $P_i$ contains $k$ distinct page requests different from $p$ (implies a fault for MIN).
- If the $k$ page faults of LRU in $P_i$ are on distinct pages (different from $p$), we’re done.
Paging – Det. strategies Analysis

Theorem 2. LRU & FIFO are $k$-competitive. No deterministic strategy is better.

Proof. (only for LRU, FIFO similar)

- Initially, the cache contains the same pages for all strategies.
- We partition $\sigma$ into phases $P_0, P_1, \ldots$, s.t. LRU has at most $k$ faults in $P_0$ and exactly $k$ faults in each other phase.
- We show next: MIN has at least 1 fault in each phase.
- Clearly, MIN also faults in $P_0$; consider $P_i$ ($i \geq 1$) and let $p$ be the last page of $P_{i-1}$.
- Show: $P_i$ contains $k$ distinct page requests different from $p$ (implies a fault for MIN).
- If the $k$ page faults of LRU in $P_i$ are on distinct pages (different from $p$), we’re done.
- Assume LRU has in $P_i$ two page faults on one page $q$. In between, $q$ has to be evicted from the cache. According to LRU, there were $k$ distinct page requests in between.

MIN: optimal strategy
$\sigma$: sequence of pages
Paging – Det. strategies Analysis

**Theorem 2.** LRU & FIFO are $k$-competitive. No deterministic strategy is better.

**Proof.** (only for LRU, FIFO similar)

- Initially, the cache contains the same pages for all strategies.
- We partition $\sigma$ into phases $P_0, P_1, \ldots$, s.t. LRU has at most $k$ faults in $P_0$ and exactly $k$ faults in each other phase.
- We show next: MIN has at least 1 fault in each phase.
- Clearly, MIN also faults in $P_0$; consider $P_i$ ($i \geq 1$) and let $p$ be the last page of $P_{i-1}$.
- Show: $P_i$ contains $k$ distinct page requests different from $p$ (implies a fault for MIN).
- If the $k$ page faults of LRU in $P_i$ are on distinct pages (different from $p$), we’re done.
- Assume LRU has in $P_i$ two page faults on one page $q$. In between, $q$ has to be evicted from the cache. According to LRU, there were $k$ distinct page requests in between.
- Similarly, if LRU faults on $p$ in $P_i$, there were $k$ distinct page requests in between.
Paging – Det. strategies Analysis

**Theorem 2.** LRU & FIFO are $k$-competitive. No deterministic strategy is better.

**Proof.** (only for LRU, FIFO similar)

- Remains to prove: No deterministic strategy is better than $k$-competitive.
Paging – Det. strategies Analysis

**Theorem 2.** LRU & FIFO are $k$-competitive. No deterministic strategy is better.

**Proof.** (only for LRU, FIFO similar)

- Remains to prove: No deterministic strategy is better than $k$-competitive.
- Let there be $k + 1$ pages in the memory system.
Paging – Det. strategies Analysis

**Theorem 2.** LRU & FIFO are $k$-competitive. No deterministic strategy is better.

**Proof.** (only for LRU, FIFO similar)

- Remains to prove: No deterministic strategy is better than $k$-competitive.
- Let there be $k + 1$ pages in the memory system.
- For any deterministic strategy there is a worst-case page sequence $\sigma^*$ always requesting the page that is currently not in the cache.
Paging – Det. strategies Analysis

**Theorem 2.** LRU & FIFO are $k$-competitive. No deterministic strategy is better.

**Proof.** (only for LRU, FIFO similar)

- Remains to prove: No deterministic strategy is better than $k$-competitive.
- Let there be $k + 1$ pages in the memory system.
- For any deterministic strategy there is a worst-case page sequence $\sigma^*$ always requesting the page that is currently not in the cache.
- Let MIN have a page fault on the $i$-th page of $\sigma^*$. 

Theorem 2. LRU & FIFO are $k$-competitive. No deterministic strategy is better.
Paging – Det. strategies Analysis

Theorem 2. LRU & FIFO are $k$-competitive. No deterministic strategy is better.

Proof. (only for LRU, FIFO similar)

- Remains to prove: No deterministic strategy is better than $k$-competitive.
- Let there be $k + 1$ pages in the memory system.
- For any deterministic strategy there is a worst-case page sequence $\sigma^*$ always requesting the page that is currently not in the cache.
- Let MIN have a page fault on the $i$-th page of $\sigma^*$.
- Then the next $k - 1$ requested pages are in the cache already & the next fault occurs on the $(i + k)$-th page of $\sigma^*$ the earliest. Until then, the det. strategy has $k$ faults.
Paging – Det. strategies Analysis

**Theorem 2.** LRU & FIFO are $k$-competitive. No deterministic strategy is better.

**Proof.** (only for LRU, FIFO similar)

- Remains to prove: No deterministic strategy is better than $k$-competitive.
- Let there be $k + 1$ pages in the memory system.
- For any deterministic strategy there is a worst-case page sequence $\sigma^*$ always requesting the page that is currently not in the cache.
- Let MIN have a page fault on the $i$-th page of $\sigma^*$.
- Then the next $k - 1$ requested pages are in the cache already & the next fault occurs on the $(i + k)$-th page of $\sigma^*$ the earliest. Until then, the det. strategy has $k$ faults.

$\Rightarrow$ The competitive ratio cannot be better than $\frac{|\sigma^*|}{\lceil \frac{|\sigma^*|}{k} \rceil} \overset{\sim \infty}{=} k$. 

$\square$

Randomized strategy: MARKING
Randomized strategy: MARKING

- Proceeds in phases

**Randomized strategy**: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.

**Randomized strategy:** MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets **marked**.
- A page for eviction is chosen **uniformly at random** from the unmarked pages.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start a new phase.

**Randomized strategy:** MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets **marked**.
- A page for eviction is chosen **uniformly at random** from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.

Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.

Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.

Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.

Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.

Phase $P_1$

**Randomized strategy:** MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets **marked**.
- A page for eviction is chosen **uniformly at random** from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.

**Randomized strategy:** MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets **marked**.
- A page for eviction is chosen **uniformly at random** from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.

Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen \textit{uniformly at random} from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start a new phase.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen \textit{uniformly at random} from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.

Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen **uniformly at random** from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start a new phase.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start a new phase.

**Randomized strategy:** MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen *uniformly at random* from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.

Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.

Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
- When a page is requested, it gets marked.
- A page for eviction is chosen uniformly at random from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all and start new phase.

Theorem 3. MARKING is $2H_k$-competitive.

Remark.

$H_k = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k}$ is the $k$-th harmonic number and for $k \geq 2$: $H_k < \ln(k) + 1$. 
Theorem 3. MARKING is $2H_k$-competitive.

Proof.
Theorem 3. MARKING is $2H_k$-competitive.

Proof. We consider phase $P_i$. 
Theorem 3. MARKING is $2H_k$-competitive.

Proof.

- A page is *stale* if it is unmarked, but was marked in $P_{i-1}$.

We consider phase $P_i$. 
Theorem 3. MARKING is $2H_k$-competitive.

Proof.

- A page is *stale* if it is unmarked, but was marked in $P_{i-1}$.
- A page is *clean* if it is unmarked, but not stale.
Paging – Rand. Strategy Analysis

**Theorem 3.** MARKING is $2H_k$-competitive.

**Proof.**
- A page is *stale* if it is unmarked, but was marked in $P_{i-1}$.
- A page is *clean* if it is unmarked, but not stale.
- $S_{\text{MARK}}$ ($S_{\text{MIN}}$): set of pages in the cache of MARKING (MIN)

We consider phase $P_i$. 
Paging – Rand. Strategy Analysis

Theorem 3. MARKING is $2H_k$-competitive.

Proof.
- A page is *stale* if it is unmarked, but was marked in $P_{i-1}$.
- A page is *clean* if it is unmarked, but not stale.
- $S_{\text{MARK}}$ ($S_{\text{MIN}}$): set of pages in the cache of MARKING (MIN)
- $d_{\text{begin}}$: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the beginning of $P_i$

We consider phase $P_i$. 
Paging – Rand. Strategy Analysis

Theorem 3. MARKING is $2H_k$-competitive.

Proof.

- A page is **stale** if it is unmarked, but was marked in $P_{i-1}$.
- A page is **clean** if it is unmarked, but not stale.
- $S_{\text{MARK}} (S_{\text{MIN}})$: set of pages in the cache of MARKING (MIN)
- $d_{\text{begin}}$: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the beginning of $P_i$
- $d_{\text{end}}$: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the end of $P_i$

We consider phase $P_i$. 
Paging – Rand. Strategy Analysis

Theorem 3. MARKING is $2H_k$-competitive.

Proof.

- A page is **stale** if it is unmarked, but was marked in $P_{i-1}$.
- A page is **clean** if it is unmarked, but not stale.
- $S_{\text{MARK}}$ ($S_{\text{MIN}}$): set of pages in the cache of MARKING (MIN)
- $d_{\text{begin}}$: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the beginning of $P_i$
- $d_{\text{end}}$: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the end of $P_i$
- $c$: number of clean pages requested in $P_i$

We consider phase $P_i$. 

Theorem 3. MARKING is $2H_k$-competitive.
Paging – Rand. Strategy Analysis

Theorem 3. MARKING is $2H_k$-competitive.

Proof.

- A page is *stale* if it is unmarked, but was marked in $P_{i-1}$.
- A page is *clean* if it is unmarked, but not stale.
- $S_{\text{MARK}}$ ($S_{\text{MIN}}$): set of pages in the cache of MARKING (MIN)
- $d_{\text{begin}}$: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the beginning of $P_i$
- $d_{\text{end}}$: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the end of $P_i$
- $c$: number of clean pages requested in $P_i$
- MIN has $\geq \max(c - d_{\text{begin}}, d_{\text{end}})$ faults.

We consider phase $P_i$. 
Paging – Rand. Strategy Analysis

**Theorem 3.** MARKING is $2H_k$-competitive.

**Proof.**

- A page is *stale* if it is unmarked, but was marked in $P_{i-1}$.
- A page is *clean* if it is unmarked, but not stale.
- $S_{\text{MARK}}$ ($S_{\text{MIN}}$): set of pages in the cache of MARKING (MIN)
- $d_{\text{begin}}$: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the beginning of $P_i$
- $d_{\text{end}}$: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the end of $P_i$
- $c$: number of clean pages requested in $P_i$
- MIN has $\geq \max(c - d_{\text{begin}}, d_{\text{end}}) \geq \frac{1}{2}(c - d_{\text{begin}} + d_{\text{end}})$ faults.

We consider phase $P_i$. 

---

**Proof.**

- A page is *stale* if it is unmarked, but was marked in $P_{i-1}$.
- A page is *clean* if it is unmarked, but not stale.
- $S_{\text{MARK}}$ ($S_{\text{MIN}}$): set of pages in the cache of MARKING (MIN)
- $d_{\text{begin}}$: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the beginning of $P_i$
- $d_{\text{end}}$: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the end of $P_i$
- $c$: number of clean pages requested in $P_i$
- MIN has $\geq \max(c - d_{\text{begin}}, d_{\text{end}}) \geq \frac{1}{2}(c - d_{\text{begin}} + d_{\text{end}})$ faults.
Paging – Rand. Strategy Analysis

Theorem 3. MARKING is $2H_k$-competitive.

Proof.

■ A page is *stale* if it is unmarked, but was marked in $P_{i-1}$.

■ A page is *clean* if it is unmarked, but not stale.

■ $S_{MARK}$ ($S_{MIN}$): set of pages in the cache of MARKING (MIN)

■ $d_{begin}$: $|S_{MIN} - S_{MARK}|$ at the beginning of $P_i$

■ $d_{end}$: $|S_{MIN} - S_{MARK}|$ at the end of $P_i$

■ $c$: number of clean pages requested in $P_i$

■ MIN has $\geq \max(c - d_{begin}, d_{end}) \geq \frac{1}{2}(c - d_{begin} + d_{end}) = \frac{c}{2} - \frac{d_{begin}}{2} + \frac{d_{end}}{2}$ faults.

We consider phase $P_i$. 
Paging – Rand. Strategy Analysis

Theorem 3. MARKING is $2H_k$-competitive.

Proof.

- A page is *stale* if it is unmarked, but was marked in $P_{i-1}$.
- A page is *clean* if it is unmarked, but not stale.
- $S_{MARK}$ ($S_{MIN}$): set of pages in the cache of MARKING (MIN)
- $d_{begin}$: $|S_{MIN} - S_{MARK}|$ at the beginning of $P_i$
- $d_{end}$: $|S_{MIN} - S_{MARK}|$ at the end of $P_i$
- $c$: number of clean pages requested in $P_i$
- MIN has $\geq \max(c - d_{begin}, d_{end}) \geq \frac{1}{2}(c - d_{begin} + d_{end}) = \frac{c}{2} - \frac{d_{begin}}{2} + \frac{d_{end}}{2}$ faults.
  Over all phases, all $\frac{d_{begin}}{2}$ and $\frac{d_{end}}{2}$ cancel out, except the first $\frac{d_{begin}}{2}$ and the last $\frac{d_{end}}{2}$.

We consider phase $P_i$. 

Proof.

- A page is *stale* if it is unmarked, but was marked in $P_{i-1}$.
- A page is *clean* if it is unmarked, but not stale.
- $S_{MARK}$ ($S_{MIN}$): set of pages in the cache of MARKING (MIN)
- $d_{begin}$: $|S_{MIN} - S_{MARK}|$ at the beginning of $P_i$
- $d_{end}$: $|S_{MIN} - S_{MARK}|$ at the end of $P_i$
- $c$: number of clean pages requested in $P_i$
- MIN has $\geq \max(c - d_{begin}, d_{end}) \geq \frac{1}{2}(c - d_{begin} + d_{end}) = \frac{c}{2} - \frac{d_{begin}}{2} + \frac{d_{end}}{2}$ faults.
  Over all phases, all $\frac{d_{begin}}{2}$ and $\frac{d_{end}}{2}$ cancel out, except the first $\frac{d_{begin}}{2}$ and the last $\frac{d_{end}}{2}$.

We consider phase $P_i$. 

Paging – Rand. Strategy Analysis

Theorem 3. MARKING is $2H_k$-competitive.

Proof.

- A page is **stale** if it is unmarked, but was marked in $P_{i-1}$.
- A page is **clean** if it is unmarked, but not stale.
- $S_{MARK}$ ($S_{MIN}$): set of pages in the cache of MARKING (MIN)
- $d_{begin}$: $|S_{MIN} - S_{MARK}|$ at the beginning of $P_i$
- $d_{end}$: $|S_{MIN} - S_{MARK}|$ at the end of $P_i$
- $c$: number of clean pages requested in $P_i$
- $d_{begin}$: $|S_{MIN} - S_{MARK}|$ at the beginning of $P_i$
- $d_{end}$: $|S_{MIN} - S_{MARK}|$ at the end of $P_i$
- $c$: number of clean pages requested in $P_i$

MIN has $\geq \max(c - d_{begin}, d_{end}) \geq \frac{1}{2}(c - d_{begin} + d_{end}) = \frac{c}{2} - \frac{d_{begin}}{2} + \frac{d_{end}}{2}$ faults.

Over all phases, all $\frac{d_{begin}}{2}$ and $\frac{d_{end}}{2}$ cancel out, except the first $\frac{d_{begin}}{2}$ and the last $\frac{d_{end}}{2}$.

Since the first $d_{begin} = 0$, MIN has at least $\frac{c}{2}$ faults per phase.
Paging – Rand. Strategy Analysis

**Theorem 3.** MARKING is $2H_k$-competitive.

**Proof.**
- For the clean pages, MARKING has $c$ faults.
Paging – Rand. Strategy Analysis

**Theorem 3.** MARKING is $2H_k$-competitive.

**Proof.**
- For the clean pages, MARKING has $c$ faults.
- For the stale pages, there are $s = k - c \leq k - 1$ requests.

We consider phase $P_i$. 
Paging – Rand. Strategy Analysis

**Theorem 3.** MARKING is $2H_k$-competitive.

**Proof.**

- For the clean pages, MARKING has $c$ faults.
- For the stale pages, there are $s = k - c \leq k - 1$ requests.
- For requests $j = 1, \ldots, s$ to stale pages, consider the expected number of faults $E[F_j]$.  

We consider phase $P_i$. 
Paging – Rand. Strategy Analysis

**Theorem 3.** MARKING is $2H_k$-competitive.

**Proof.**

- For the clean pages, MARKING has $c$ faults.
- For the stale pages, there are $s = k - c \leq k - 1$ requests.
- For requests $j = 1, \ldots, s$ to stale pages, consider the expected number of faults $E[F_j]$.

$c(j)$: # clean pages requested in $P_i$ so far
$s(j)$: # pages that were stale at the beginning of $P_i$ and have not been requested
Paging – Rand. Strategy Analysis

Theorem 3. MARKING is $2H_k$-competitive.

Proof.
- For the clean pages, MARKING has $c$ faults.
- For the stale pages, there are $s = k - c \leq k - 1$ requests.
- For requests $j = 1, \ldots, s$ to stale pages, consider the expected number of faults $E[F_j]$.
- $c(j)$: # clean pages requested in $P_i$ so far
  $s(j)$: # pages that were stale at the beginning of $P_i$ and have not been requested
- $E[F_j] = \frac{s(j) - c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1$

We consider phase $P_i$. 
Paging – Rand. Strategy Analysis

**Theorem 3.** MARKING is \(2H_k\)-competitive.

**Proof.**

- For the clean pages, MARKING has \(c\) faults.
- For the stale pages, there are \(s = k - c \leq k - 1\) requests.
- For requests \(j = 1, \ldots, s\) to stale pages, consider the expected number of faults \(E[F_j]\).
- \(c(j)\): \# clean pages requested in \(P_i\) so far
  
  \(s(j)\): \# pages that were stale at the beginning of \(P_i\) and have not been requested

\[
E[F_j] = \frac{s(j) - c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \leq \frac{c}{k+1-j}
\]
Paging – Rand. Strategy Analysis

Theorem 3. MARKING is $2H_k$-competitive.

Proof.

- For the clean pages, MARKING has $c$ faults.
- For the stale pages, there are $s = k - c \leq k - 1$ requests.
- For requests $j = 1, \ldots, s$ to stale pages, consider the expected number of faults $E[F_j]$.
- $c(j)$: # clean pages requested in $P_i$ so far
- $s(j)$: # pages that were stale at the beginning of $P_i$ and have not been requested
- $E[F_j] = \frac{s(j) - c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \leq \frac{c}{k+1-j}$
- $E\left[\sum_{j=1}^{s} F_j\right] = \sum_{j=1}^{s} E[F_j]$
Paging – Rand. Strategy Analysis

**Theorem 3.** MARKING is $2H_k$-competitive.

**Proof.**

- For the clean pages, MARKING has $c$ faults.
- For the stale pages, there are $s = k - c \leq k - 1$ requests.
- For requests $j = 1, \ldots, s$ to stale pages, consider the expected number of faults $E[F_j]$.

$c(j)$: # clean pages requested in $P_i$ so far
$s(j)$: # pages that were stale at the beginning of $P_i$ and have not been requested

- $E[F_j] = \frac{s(j) - c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \leq \frac{c}{k+1-j}$

- $E\left[\sum_{j=1}^{s} F_j\right] = \sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j}$
Paging – Rand. Strategy Analysis

Theorem 3. MARKING is $2H_k$-competitive.

Proof.

- For the clean pages, MARKING has $c$ faults.
- For the stale pages, there are $s = k - c \leq k - 1$ requests.
- For requests $j = 1, \ldots, s$ to stale pages, consider the expected number of faults $E[F_j]$.

$c(j)$: # clean pages requested in $P_i$ so far
$s(j)$: # pages that were stale at the beginning of $P_i$ and have not been requested

\[ E[F_j] = \frac{s(j) - c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \leq \frac{c}{k+1-j} \]

\[ E \left[ \sum_{j=1}^{s} F_j \right] = \sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \sum_{j=1}^{c} \frac{c}{k+1-j} \leq \sum_{j=2}^{k} \frac{c}{j} \]
Paging – Rand. Strategy Analysis

Theorem 3. MARKING is $2H_k$-competitive.

Proof.

- For the clean pages, MARKING has $c$ faults.
- For the stale pages, there are $s = k - c \leq k - 1$ requests.
- For requests $j = 1, \ldots, s$ to stale pages, consider the expected number of faults $E[F_j]$.

$c(j)$: # clean pages requested in $P_i$ so far
$s(j)$: # pages that were stale at the beginning of $P_i$ and have not been requested

- $E[F_j] = \frac{s(j) - c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \leq \frac{c}{k+1-j}

- $E\left[\sum_{j=1}^{s} F_j\right] = \sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j} \leq \sum_{j=2}^{k} \frac{c}{j} = c \cdot (H_k - 1)$

We consider phase $P_i$. 

For requests $j = 1, \ldots, s$ to stale pages, consider the expected number of faults $E[F_j]$. 

$E\left[\sum_{j=1}^{s} F_j\right] = \sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j} \leq \sum_{j=2}^{k} \frac{c}{j} = c \cdot (H_k - 1)$
Paging – Rand. Strategy Analysis

**Theorem 3.** MARKING is $2H_k$-competitive.

**Proof.**
- For the clean pages, MARKING has $c$ faults.
- For the stale pages, there are $s = k - c \leq k - 1$ requests.
- For requests $j = 1, \ldots, s$ to stale pages, consider the expected number of faults $E[F_j]$.
- $c(j)$: # clean pages requested in $P_i$ so far
- $s(j)$: # pages that were stale at the beginning of $P_i$ and have not been requested

$$E[F_j] = \frac{s(j) - c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \leq \frac{c}{k+1-j}$$

$$E \left[ \sum_{j=1}^{s} F_j \right] = \sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j} \leq \sum_{j=2}^{k} \frac{c}{j} = c \cdot (H_k - 1)$$

So the competitive ratio of MARKING is at most $\frac{c + c(H_k - 1)}{c/2} = 2H_k \in O(\log k)$
Theorem 3. MARKING is $2H_k$-competitive.

Proof.

- For the clean pages, MARKING has $c$ faults.
- For the stale pages, there are $s = k - c \leq k - 1$ requests.
- For requests $j = 1, \ldots, s$ to stale pages, consider the expected number of faults $E[F_j]$.

$c(j)$: # clean pages requested in $P_i$ so far
$s(j)$: # pages that were stale at the beginning of $P_i$ and have not been requested

$$E[F_j] = \frac{s(j) - c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \leq \frac{c}{k+1-j}$$

$$E \left[ \sum_{j=1}^{s} F_j \right] = \sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j} \leq \sum_{j=2}^{k} \frac{c}{j} = c \cdot (H_k - 1)$$

So the competitive ratio of MARKING is at most $\frac{c + c(H_k-1)}{c/2} = 2H_k \in O(\log k)$ □
Theorem 3. MARKING is $2H_k$-competitive.

Proof.

- For the clean pages, MARKING has $c$ faults.
- For the stale pages, there are $s = k - c \leq k - 1$ requests.
- For requests $j = 1, \ldots, s$ to stale pages, consider the expected number of faults $E[F_j]$.

$c(j)$: # clean pages requested in $P_i$ so far  
$s(j)$: # pages that were stale at the beginning of $P_i$ and have not been requested

- $E[F_j] = \frac{s(j) - c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \leq \frac{c}{k+1-j}$

- $E\left[\sum_{j=1}^{s} F_j\right] = \sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j} \leq \sum_{j=2}^{k} \frac{c}{j} = c \cdot (H_k - 1)$

- So the competitive ratio of MARKING is at most $\frac{c + c(H_k - 1)}{c/2} = 2H_k \in O(\log k)$

Reminder.
No deterministic strategy is better than $k$-competitive. 
MARKING is $O(\log k)$-competitive.
**Theorem 3.** MARKING is $2H_k$-competitive.

**Proof.**

- For the clean pages, MARKING has $c$ faults.
- For the stale pages, there are $s = k - c \leq k - 1$ requests.
- For requests $j = 1, \ldots, s$ to stale pages, consider the expected number of faults $E[F_j]$.

$c(j)$: # clean pages requested in $P_i$ so far

$s(j)$: # pages that were stale at the beginning of $P_i$ and have not been requested

$$E[F_j] = \frac{s(j)-c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \leq \frac{c}{k+1-j}$$

$$E\left[\sum_{j=1}^{s} F_j\right] = \sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j} \leq \sum_{j=2}^{k} \frac{c}{j} = c \cdot (H_k - 1)$$

So the competitive ratio of MARKING is at most $\frac{c+c(H_k-1)}{c/2} = 2H_k \in O(\log k)$

**Reminder.**

No deterministic strategy is better than $k$-competitive.

MARKING is $O(\log k)$-competitive $\implies$ exponential improvement!
Discussion

Online algorithms operate in a setting different from that of classical algorithms. However, this setting of incomplete information is very natural and occurs often in real-world applications. Can you think of further examples?
Discussion

- Online algorithms operate in a setting different from that of classical algorithms. However, this setting of incomplete information is very natural and occurs often in real-world applications. Can you think of further examples?

- We might also transform a classical problem with incomplete information into an online problem. E.g.: Matching problem for ride sharing.
Online algorithms operate in a setting different from that of classical algorithms. However, this setting of incomplete information is very natural and occurs often in real-world applications. Can you think of further examples?

We might also transform a classical problem with incomplete information into an online problem. E.g.: Matching problem for ride sharing.

Randomization can help to improve our behavior on worst-case instances. You may also think of: we are less predictable for an adversary.
Literature

Main source:


Original papers:

- [Sleator, Tarjan ’85] “Amortized Efficiency of List Update and Paging Rules.”
- [Fiat, Karp, Luby, McGeoch, Sleator, Young ’91] “Competitive Paging Algorithms.”