Advanced Algorithms

Online Algorithms

Ski-Rental Problem and Paging

Johannes Zink · WS20
Introduction

Winter is about to begin . . .
Introduction

Winter is about to begin . . . . . . this means the ski season is back!* 

* in a normal year not being 2020
Introduction

Winter is about to begin . . . . . . this means the ski season is back!* 

But what if there is not always enough snow?  

* in a normal year not being 2020
Ski-Rental Problem

Winter is about to begin . . .

. . . this means the ski season is back!*  

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

* in a normal year not being 2020
Ski-Rental Problem

Winter is about to begin . . .

. . . this means the ski 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.

* in a normal year not being 2020
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.

**Cost.**
- 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 is a **good** day.
- Each morning, we can check if today is a good day, but we can’t check any earlier.

**Cost.**
- 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.

Cost.
■ 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 is a good day.
■ Each morning, we can check if today is a good day, but we can’t check any earlier.

Cost.
■ 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.

Cost.
■ 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?

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

Renting costs 1/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/day
Buying costs $M$
$T$ good days
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 $M$. 
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.

→ 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.

→ 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.

$$\Rightarrow \text{for arbitrarily large } M \text{ arbitrarily bad}$$

**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.

$$\Rightarrow \text{for arbitrarily large } T \text{ arbitrarily bad}$$
Ski-Rental Problem – Strategy III

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

Is there a strategy that cannot become arbitrarily bad?

Renting costs 1/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/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 $\frac{2M-1}{M} = 2 - \frac{1}{M}$ as $M \to \infty$. 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 $\frac{2M-1}{M} = 2 - \frac{1}{M}$.

$\Rightarrow$ 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 $M \xrightarrow{} \infty$; 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 $\frac{2M-1}{M} = 2 - \frac{1}{M}$.
  \[ M \xrightarrow{\sim} \infty \Rightarrow M \gg 0 \Rightarrow \frac{M}{M-1} = 1 + \frac{1}{M-1} \approx 1 \]
  \[ 2 - \frac{1}{M} \approx 2 \]

$\Rightarrow$ 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.
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 days of rental' for a fixed $X$. 

Renting costs 1/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 $\frac{2M-1}{M} = 2 - \frac{1}{M}$ as $M \rightarrow \infty$.
  $\Rightarrow$ Strategy III is deterministic and 2-competitive.

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

**Proof Idea.**

- Any det. strategy can be formulated as 'buy on the $X$-th days 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 $T = X$. 

Renting costs 1/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 $\frac{2M-1}{M} = 2 - \frac{1}{M}$. $M \rightarrow \infty \Rightarrow$ Strategy III is deterministic and 2-competitive.

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

**Proof Idea.**

- Any det. strategy can be formulated as 'buy on the $X$-th days 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 $T = X$.

- $\frac{c_{\text{det}}}{c_{\text{OPT}}}$ costs for deterministic strategy

- $\frac{c_{\text{det}}}{c_{\text{OPT}}}$ costs for optimal strategy

Renting costs 1/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 $\frac{2M-1}{M} = 2 - \frac{1}{M} \quad \text{as} \quad M \to \infty = 2$.

$\Rightarrow$ 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 days 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 $\frac{2M-1}{M} = 2 - \frac{1}{M} \Rightarrow M \xrightarrow{\infty} 2$.

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

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

Proof Idea.

- Any det. strategy can be formulated as 'buy on the $X$-th days 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 $T = X$.

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

$$\frac{c_{\text{det}}}{c_{\text{OPT}}} = \frac{X-1+M}{\min(X,M)} \geq \min \left( \frac{X-1+X+1}{X}, \frac{M-1+M}{M} \right) = \min \left( 2, 2 - \frac{1}{M} \right) = 2 - \frac{1}{M}$$
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 $\frac{2M-1}{M} = 2 - \frac{1}{M}$. $M \Rightarrow \infty$ 2.
  $\Rightarrow$ Strategy III is deterministic and 2-competitive.

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

**Proof Idea.**

- Any det. strategy can be formulated as 'buy on the $X$-th days 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 $T = X$.
- $\frac{c_{\text{det}}}{c_{\text{OPT}}} = \frac{X-1+M}{\min(X,M)} \geq \min \left( \frac{X-1+X+1}{X}, \frac{M-1+M}{M} \right) = \min \left( 2, 2 - \frac{1}{M} \right) = 2 - \frac{1}{M}$. $M \Rightarrow \infty$ 2.
Ski-Rental Problem – Strategy IV

Can we get below this bound using randomization?

Renting costs 1/day
Buying costs $M$
$T$ good days
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 $T = M$: 
  
  \[
  \frac{E[c_{\text{Strategy IV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2} \cdot (2M-1) + \frac{1}{2} \cdot ((1+\alpha)M-1)}{M} = \frac{3+\alpha}{2} - \frac{1}{M} \xrightarrow{M \to \infty} \frac{3+\alpha}{2}
  \]
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 \( T = M \):
  \[
  \frac{E[c_{Strategy IV}]}{c_{OPT}} = \frac{\frac{1}{2} \cdot (2M-1) + \frac{1}{2} \cdot ((1+\alpha)M-1)}{M} = \frac{3+\alpha}{2} - \frac{1}{M} \xrightarrow{M \to \infty} \frac{3+\alpha}{2}
  \]

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

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

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

## Try $\alpha = \frac{1}{2}$

Renting costs 1/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 $T = M$: $E_{c_{\text{StrategyIV}}}^c \frac{c_{\text{StrategyIV}}}{c_{\text{OPT}}} = 2 \cdot \frac{(2M - 1) + \frac{1}{2} \cdot ((1 + \alpha)M - 1)}{M} = \frac{3 + \alpha}{2} - \frac{1}{M} \overset{M \to \infty}{=} \frac{3 + \alpha}{2} = \frac{7}{4} < 2$

  - Case $T = \alpha M$: $E_{c_{\text{StrategyIV}}}^c \frac{c_{\text{StrategyIV}}}{c_{\text{OPT}}} = 2 \cdot \frac{\frac{1}{2} \cdot \alpha M + \frac{1}{2} \cdot ((1 + \alpha)M - 1)}{\alpha M} = 1 + \frac{1}{2\alpha} - \frac{1}{2\alpha M} \overset{M \to \infty}{=} 1 + \frac{1}{2\alpha} = 2$

  - 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 $T = \alpha M$
  
  **try $\alpha = \frac{1}{2}$**

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

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

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

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

- The w. c. ratio is minimum if $\frac{3+\alpha}{2} = 1 + \frac{1}{2\alpha}$
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 $T = M$: \[ \frac{E[c_{\text{Strategy IV}}]}{c_{\text{OPT}}} = \frac{1}{2} \cdot (2M-1) + \frac{1}{2} \cdot ((1+\alpha)M-1) = \frac{3+\alpha}{2} - \frac{1}{M} \overset{M \to \infty}{\Rightarrow} \frac{3+\alpha}{2} \]

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

- The w. c. ratio is minimum if \[ \frac{3+\alpha}{2} = 1 + \frac{1}{2\alpha} \Rightarrow \alpha = \frac{\sqrt{5}-1}{2} \]
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 αM-th good day (α ∈ (0, 1))

- Observation: worst case can only be T = M or T = αM

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

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

⇒ Strategy IV (with α = \( \frac{\sqrt{5}-1}{2} \approx 0.62 \)) 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 even get a competitive ratio of $\frac{e}{e-1} \approx 1.58$. 

Renting costs 1/day
Buying costs $M$ 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

<table>
<thead>
<tr>
<th><strong>Online Algorithm</strong></th>
<th><strong>Offline Algorithm</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>■ No full information available initially <em>(online problem)</em></td>
<td></td>
</tr>
<tr>
<td>■ Decisions are made with incomplete information.</td>
<td></td>
</tr>
<tr>
<td>■ The algorithm may get more informations over time or by exploring the instance.</td>
<td></td>
</tr>
</tbody>
</table>
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 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**.
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
  - Paging (replacing entries in a memory)
  - Insertion Sort
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, Paging (replacing entries in a memory), Insertion Sort
Online vs. Offline Algorithms

**Online Algorithm**
- No full information available initially \((\text{online problem})\)
- 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 \((\text{offline problem})\)
- 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 \textit{competitive ratio}.

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

| 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 having to be fulfilled in order. We have to fulfill a request before we see the next request. |

---

**Given (offline/online):**

| $p_1$, $p_2$, $p_3$, $p_4$, $p_5$, $p_6$, $p_7$, $p_8$, $p_9$ |

$\sigma$ of page requests having 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 having 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 having 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 having 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 \textbf{having 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 having 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 having 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 having 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 **having 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 having 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.
Paging – det. strat.

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

- 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.
Paging – det. strat.  

- 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?

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.
Paging – det. strategies analysis

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

**Proof.** (only for LRU, FIFO similar)
Paging – det. strategies analysis

**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
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.

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.
- 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.
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

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)

- 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

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)

- 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.

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)

- 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’s 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’s 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’s 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’s 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.

\[ \Rightarrow \text{The competitive ratio cannot be better than } \frac{|\sigma^*|}{\lceil |\sigma^*|/k \rceil} \sim \infty = k. \]
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’s 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 $\frac{|\sigma^*|}{k} \sim \infty = k$. 

$\square$
Paging – rand. strat.

**Randomized strategy:** MARKING
Paging – rand. strat.

**Randomized strategy**: MARKING

- Proceeds in phases
Randomized strategy: MARKING

- Proceeds in phases
- At the beginning of each phase, all pages are unmarked.
Paging – rand. strat.

**Randomized strategy:** MARKING

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

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 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.
Paging – rand. strat.

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.
Paging – rand. strat.

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.
Paging – rand. strat.

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.
Paging – rand. strat.

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.
Paging – rand. strat.

**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.
Paging – rand. strat.

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.
Paging – rand. strat.

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.
Paging – rand. strat.

**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.
Paging – rand. strat.

**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.

Phase $P_2$
Paging – rand. strat.

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.
Paging – rand. strat.

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.
Paging – rand. strat.

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.
Paging – rand. strat.

**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.
Paging – rand. strat.

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.
Paging – rand. strat.

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.
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 $k \geq 2$: $H_k < \ln(k) + 1$. 

$$k \begin{array}{c} p_6 \ h \ p_5 \ h \ p_3 \ h \ \\ p_4 \ h \ p_1 \ h \ p_2 \ h \ p_7 \ h \ p_8 \ h \ p_9 \ h \ \\ \end{array}$$

Phase $P_2$
Paging – rand. strategy analysis

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

**Proof.**
Proof.

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 $P_{i-1}$.

We consider phase $P_i$. 

- **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.

We consider phase $P_i$. 

We consider phase $P_i$. 

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)

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.
- $S_{\text{MARK}}$ (or $S_{\text{MIN}}$): 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 $P_i$. 

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

We consider phase $P_i$. 

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

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

- Since the first $d_{\text{begin}} = 0$, 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 $P_i$. 

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

[Figure or table, if any, from the document]
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 this phase so far
- $s(j)$: # phase-initially stale pages having 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 this phase so far
- $s(j)$: # phase-initially stale pages having 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 this phase so far
  - $s(j)$: # phase-initially stale pages having not been requested

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

We consider phase $P_i$. 

- $s(j) = k + 1 - j$
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 this phase so far
$s(j)$: # phase-initially stale pages having not been requested

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

- $\sum_{j=1}^{s} E[F_j] \leq \sum_{j=2}^{k} c \leq c \cdot (H_k - 1)$
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 this phase so far
- $s(j)$: # phase-initially stale pages having not been requested
- $E[F_j] = \frac{s(j) - c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \leq \frac{c}{k+1-j}$
- $\sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j}$

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 this phase so far
- $s(j)$: # phase-initially stale pages having not been requested
- $E[F_j] = \frac{s(j) - c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \leq \frac{c}{k+1-j}$
- $\sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j} \leq \sum_{j=2}^{k} \frac{c}{j}$

We consider phase $P_i$. 

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 this phase so far
- $s(j)$: # phase-initially stale pages having not been requested
- $E[F_j] = \frac{s(j) - c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \leq \frac{c}{k+1-j}$
- $\sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j} \leq \sum_{j=2}^{k} \frac{c}{j}$
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 this phase so far
$s(j)$: # phase-initially stale pages having not been requested

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

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

$s(j)$: # phase-initially stale pages having not been requested

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

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

So the competitive ratio of Marking is $\frac{c+c(H_k-1)}{c/2} = 2H_k$. 

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 this phase so far
- $s(j)$: # phase-initially stale pages having not been requested

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

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

- So the competitive ratio of Marking is $\frac{c + c(H_k - 1)}{c/2} = 2H_k$. \qed
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 this phase so far  
$s(j)$: \# phase-initially stale pages having not been requested

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

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

- So the competitive ratio of Marking is $\frac{c + c(H_k - 1)}{c/2} = 2H_k$. □
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 this phase so far
  $s(j)$: # phase-initially stale pages having not been requested
- $E[F_j] = \frac{s(j) - c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \leq \frac{c}{k+1-j}$
- $\sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j} \leq \sum_{j=2}^{k} \frac{c}{j} = c \cdot (H_k - 1)$
- So the competitive ratio of Marking is $\frac{c + c(H_k - 1)}{c/2} = 2H_k$. 

**Reminder.**
No deterministic strategy is better than $k$-competitive.
$\Rightarrow$ Randomization helps!

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

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?
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.
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.

- 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.”