Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Lehrstuhl für
INFORMATIK I
Effiziente Algorithmen und
wissensbasierte Systeme

Institut für Informatik

Advanced Algorithms

Winter term 2019/20

Lecture 5. Online Algorithms

(based on lecture notes of Sabine Storandt)

Johannes Zink

Chair for Computer Science I

Introduction

Winter is about to begin . . .

Introduction

Winter is about to begin . . .

. . . this means the ski season is back!

Introduction

Winter is about to begin . . .

. . . this means the ski season is back!

Introduction

Winter is about to begin . . .

. . . this means the ski season is back!

But what if there is not always enough snow?

Introduction

Winter is about to begin . . .

. . . this means the ski season is back!

But what if there is not always enough snow?

Introduction

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?

Introduction

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.

Ski-Rental Problem

- Every day when there is "good" weather, you go skiing (it's a *good* day).

Ski-Rental Problem

- Every day when there is "good" weather, you go skiing (it's a *good* day).
- Every day in the morning, you know if today is a good day.

Ski-Rental Problem

- Every day when there is "good" weather, you go skiing (it's a *good* day).
- Every day in the morning, you know if today is a good day.
- Renting skis for 1 day costs 1 [Euro].

Ski-Rental Problem

- Every day when there is "good" weather, you go skiing (it's a *good* day).
- Every day in the morning, you know if today is a good day.
- Renting skis for 1 day costs 1 [Euro].
- Buying skis costs M [Euros] and you have them forever.

Ski-Rental Problem

- Every day when there is "good" weather, you go skiing (it's a *good* day).
- Every day in the morning, you know if today is a good day.
- 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

- Every day when there is "good" weather, you go skiing (it's a *good* day).
- Every day in the morning, you know if today is a good day.
- 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? – We don't know T!

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

Strategy I: buy on the first good day

Ski-Rental Problem

Renting costs 1/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.

Ski-Rental Problem

Renting costs 1/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.

Ski-Rental Problem

Renting costs 1/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.

Ski-Rental Problem

Renting costs 1/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.

  $\rightarrow$ for arbitrarily large M arbitrarily bad
- So Strategy I is M times worse than the optimal strategy.

Ski-Rental Problem

Renting costs 1/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.

$\rightarrow$ for arbitrarily large M arbitrarily bad

- So Strategy I is M times worse than the optimal strategy.

Strategy II: never buy, always rent

Ski-Rental Problem

Renting costs 1/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.

$\rightarrow$ for arbitrarily large M arbitrarily bad

- So Strategy I is M times worse than the optimal strategy.

Strategy II: never buy, always rent

- Imagine there are many good days ($T > M$).

Ski-Rental Problem

Renting costs 1/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.

$\rightarrow$ for arbitrarily large M arbitrarily bad

- So Strategy I is M times worse than the optimal strategy.

Strategy II: never buy, always rent

- Imagine there are many good days ($T > M$).
- Then we have paid T; optimally, we would have bought on or before the first good day and paid M.

Ski-Rental Problem

Renting costs 1/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.

$\rightarrow$ for arbitrarily large M arbitrarily bad

- So Strategy I is M times worse than the optimal strategy.

Strategy II: never buy, always rent

- Imagine there are many good days ($T > M$).
- Then we have paid T; optimally, we would have bought on or before the first good day and paid M.
- So Strategy II is T/M times worse than the optimal strategy.

Ski-Rental Problem

Renting costs 1/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.

  $\rightarrow$ for arbitrarily large M arbitrarily bad
- So Strategy I is M times worse than the optimal strategy.

Strategy II: never buy, always rent

- Imagine there are many good days ($T > M$).
- Then we have paid T; optimally, we would have bought on or before the first good day and paid M.
- So Strategy II is T/M times worse than the optimal strategy.

  $\rightarrow$ for arbitrarily large T arbitrarily bad

Ski-Rental Problem

Renting costs 1/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.

$\rightarrow$ for arbitrarily large M arbitrarily bad

- So Strategy I is M times worse than the optimal strategy.

Strategy II: never buy, always rent

- Imagine there are many good days ($T > M$).
- Then we have paid T; optimally, we would have bought on or before the first good day and paid M.
- So Strategy II is T/M times worse than the optimal strategy.

$\rightarrow$ for arbitrarily large T arbitrarily bad

Ski-Rental Problem

Renting costs 1/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.

$\rightarrow$ for arbitrarily large M arbitrarily bad

- So Strategy I is M times worse than the optimal strategy.

Strategy II: never buy, always rent

competitive ratio

- Imagine there are many good days ($T > M$).
- Then we have paid T; optimally, we would have bought on or before the first good day and paid M.
- So Strategy II is T/M times worse than the optimal strategy.

$\rightarrow$ for arbitrarily large T arbitrarily bad

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

Is there a strategy that cannot become arbitrarily bad?

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy III: buy after M good days

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy III: buy after M good days

- Observation: the optimal solution pays $\min(M, T)$

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy III: buy after M good days

- Observation: the optimal solution pays $\min(M, T)$
- If $T \leq M$, the competitive ratio is 1.

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy III: buy after M good days

- Observation: the optimal solution pays $\min(M, T)$
- If $T \leq M$, the competitive ratio is 1.
- Otherwise, the competitive ratio is $2M/M = 2$.

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy III: buy after M good days

- Observation: the optimal solution pays $\min(M, T)$
- If $T \leq M$, the competitive ratio is 1.
- Otherwise, the competitive ratio is $2M/M = 2$.

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

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy III: buy after M good days

- Observation: the optimal solution pays $\min(M, T)$
- If $T \leq M$, the competitive ratio is 1.
- Otherwise, the competitive ratio is $2M/M = 2$.

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

Theorem: No deterministic strategy is better than 2-competitive.

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy III: buy after M good days

- Observation: the optimal solution pays $\min(M, T)$
- If $T \leq M$, the competitive ratio is 1.
- Otherwise, the competitive ratio is $2M/M = 2$.

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

Theorem: No deterministic strategy is better than 2-competitive.

Proof Idea:

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy III: buy after M good days

- Observation: the optimal solution pays $\min(M, T)$
- If $T \leq M$, the competitive ratio is 1.
- Otherwise, the competitive ratio is $2M/M = 2$.

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

Theorem: No deterministic strategy is better than 2-competitive.

Proof Idea:
- Every deterministic strategy can be formulated as 'buy after X days of rental' for a fixed X.

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy III: buy after M good days

- Observation: the optimal solution pays $\min(M, T)$
- If $T \leq M$, the competitive ratio is 1.
- Otherwise, the competitive ratio is $2M/M = 2$.

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

Theorem: No deterministic strategy is better than 2-competitive.

Proof Idea:

- Every deterministic strategy can be formulated as 'buy after X days of rental' for a fixed X.
- For $X = 0$ and $X = \infty$ it's arbitrarily bad; assume $X \in \mathbb{N}^+$.

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy III: buy after M good days

- Observation: the optimal solution pays $\min(M, T)$
- If $T \leq M$, the competitive ratio is 1.
- Otherwise, the competitive ratio is $2M/M = 2$.

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

Theorem: No deterministic strategy is better than 2-competitive.

Proof Idea:

- Every deterministic strategy can be formulated as 'buy after X days of rental' for a fixed X.
- For $X = 0$ and $X = \infty$ it's arbitrarily bad; assume $X \in \mathbb{N}^+$.
- $\frac{c_{\text{det}}}{c_{\text{OPT}}}$

costs for deterministic startegy

costs for optimal startegy

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy III: buy after M good days

- Observation: the optimal solution pays $\min(M, T)$
- If $T \leq M$, the competitive ratio is 1.
- Otherwise, the competitive ratio is $2M/M = 2$.

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

Theorem: No deterministic strategy is better than 2-competitive.

Proof Idea:
- Every deterministic strategy can be formulated as 'buy after X days of rental' for a fixed X.
- For $X = 0$ and $X = \infty$ it's arbitrarily bad; assume $X \in \mathbb{N}^+$.
- $\frac{c_{\text{det}}}{c_{\text{OPT}}} = \frac{X+M}{\min(X,M)}$

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy III: buy after M good days

- Observation: the optimal solution pays $\min(M, T)$
- If $T \leq M$, the competitive ratio is 1.
- Otherwise, the competitive ratio is $2M/M = 2$.

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

Theorem: No deterministic strategy is better than 2-competitive.

Proof Idea:
- Every deterministic strategy can be formulated as 'buy after X days of rental' for a fixed X.
- For $X = 0$ and $X = \infty$ it's arbitrarily bad; assume $X \in \mathbb{N}^+$.
- $\frac{c_{\text{det}}}{c_{\text{OPT}}} = \frac{X+M}{\min(X,M)} = \frac{X}{\min(X,M)} + \frac{M}{\min(X,M)}$

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy III: buy after M good days

- Observation: the optimal solution pays $\min(M, T)$
- If $T \leq M$, the competitive ratio is 1.
- Otherwise, the competitive ratio is $2M/M = 2$.

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

Theorem: No deterministic strategy is better than 2-competitive.

Proof Idea:
- Every deterministic strategy can be formulated as 'buy after X days of rental' for a fixed X.
- For $X = 0$ and $X = \infty$ it's arbitrarily bad; assume $X \in \mathbb{N}^+$.
- $\frac{c_{\text{det}}}{c_{\text{OPT}}} = \frac{X+M}{\min(X,M)} = \frac{X}{\min(X,M)} + \frac{M}{\min(X,M)} \geq 1 + 1 = 2$

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

Can we get below this bound using randomization?

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy IV: throw a coin; HEAD: buy after M good days, TAIL: buy after αM good days ($\alpha \in (0,1)$)

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy IV: throw a coin; HEAD: buy after M good days, TAIL: buy after αM good days ($\alpha \in (0,1)$)

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

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy IV: throw a coin; HEAD: buy after M good days, TAIL: buy after αM good days ($\alpha \in (0,1)$)

- Observation: worst case can only be $T = M$ or $T = \alpha M$
- Case $T = M$: $\frac{E[c_{\text{StrategyIV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2}\cdot 2M + \frac{1}{2}\cdot(1+\alpha)M}{M} = \frac{3+\alpha}{2}$

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy IV: throw a coin; HEAD: buy after M good days, TAIL: buy after αM good days ($\alpha \in (0,1)$)

- Observation: worst case can only be $T = M$ or $T = \alpha M$
- Case $T = M$: $\frac{E[c_{\text{StrategyIV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2}\cdot 2M + \frac{1}{2}\cdot(1+\alpha)M}{M} = \frac{3+\alpha}{2}$
- Case $T = \alpha M$: $\frac{E[c_{\text{StrategyIV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2}\cdot \alpha M + \frac{1}{2}\cdot(1+\alpha)M}{\alpha M} = 1 + \frac{1}{2\alpha}$

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy IV: throw a coin; HEAD: buy after M good days, TAIL: buy after αM good days ($\alpha \in (0,1)$)

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

try $\alpha = \frac{1}{2}$

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

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy IV: throw a coin; HEAD: buy after M good days, TAIL: buy after αM good days ($\alpha \in (0,1)$)

- Observation: worst case can only be $T = M$ or $T = \alpha M$
- Case $T = M$: $\frac{E[c_{\text{StrategyIV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2}\cdot 2M + \frac{1}{2}\cdot(1+\alpha)M}{M} = \frac{3+\alpha}{2} = \frac{7}{4} < 2$ try $\alpha = \frac{1}{2}$
- Case $T = \alpha M$: $\frac{E[c_{\text{StrategyIV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2}\cdot \alpha M + \frac{1}{2}\cdot(1+\alpha)M}{\alpha M} = 1 + \frac{1}{2\alpha}$

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy IV: throw a coin; HEAD: buy after M good days, TAIL: buy after αM good days ($\alpha \in (0,1)$)

- Observation: worst case can only be $T = M$ or $T = \alpha M$
- Case $T = M$: $\frac{E[c_{\text{StrategyIV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2}\cdot 2M + \frac{1}{2}\cdot(1+\alpha)M}{M} = \frac{3+\alpha}{2} = \frac{7}{4} < 2$ try $\alpha = \frac{1}{2}$
- Case $T = \alpha M$: $\frac{E[c_{\text{StrategyIV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2}\cdot \alpha M + \frac{1}{2}\cdot(1+\alpha)M}{\alpha M} = 1 + \frac{1}{2\alpha} = 2$

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy IV: throw a coin; HEAD: buy after M good days, TAIL: buy after αM good days ($\alpha \in (0,1)$)

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

try $\alpha = \frac{1}{2}$

- Case $T = M$: $\frac{E[c_{\text{StrategyIV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2}\cdot 2M + \frac{1}{2}\cdot(1+\alpha)M}{M} = \frac{3+\alpha}{2} = \frac{7}{4} < 2$
- Case $T = \alpha M$: $\frac{E[c_{\text{StrategyIV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2}\cdot \alpha M + \frac{1}{2}\cdot(1+\alpha)M}{\alpha M} = 1 + \frac{1}{2\alpha} = 2$

not better than the det. Strategy **III**

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy IV: throw a coin; HEAD: buy after M good days, TAIL: buy after αM good days ($\alpha \in (0,1)$)

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

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy IV: throw a coin; HEAD: buy after M good days, TAIL: buy after αM good days ($\alpha \in (0,1)$)

- Observation: worst case can only be $T = M$ or $T = \alpha M$
- Case $T = M$: $\frac{E[c_{\text{StrategyIV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2}\cdot 2M + \frac{1}{2}\cdot(1+\alpha)M}{M} = \frac{3+\alpha}{2}$
- Case $T = \alpha M$: $\frac{E[c_{\text{StrategyIV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2}\cdot \alpha M + \frac{1}{2}\cdot(1+\alpha)M}{\alpha M} = 1 + \frac{1}{2\alpha}$
- 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

Renting costs 1/day
Buying costs M
T good days

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

Strategy IV: throw a coin; HEAD: buy after M good days, TAIL: buy after αM good days ($\alpha \in (0,1)$)

- Observation: worst case can only be $T = M$ or $T = \alpha M$
- Case $T = M$: $\frac{E[c_{\text{StrategyIV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2}\cdot 2M + \frac{1}{2}\cdot(1+\alpha)M}{M} = \frac{3+\alpha}{2}$
- Case $T = \alpha M$: $\frac{E[c_{\text{StrategyIV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2}\cdot \alpha M + \frac{1}{2}\cdot(1+\alpha)M}{\alpha M} = 1 + \frac{1}{2\alpha}$
- 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.

Ski-Rental Problem

Renting costs 1/day
Buying costs M
T good days

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

Strategy IV: throw a coin; HEAD: buy after M good days, TAIL: buy after αM good days ($\alpha \in (0,1)$)

- Observation: worst case can only be $T = M$ or $T = \alpha M$
- Case $T = M$: $\frac{E[c_{\text{StrategyIV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2}\cdot 2M + \frac{1}{2}\cdot(1+\alpha)M}{M} = \frac{3+\alpha}{2}$
- Case $T = \alpha M$: $\frac{E[c_{\text{StrategyIV}}]}{c_{\text{OPT}}} = \frac{\frac{1}{2}\cdot \alpha M + \frac{1}{2}\cdot(1+\alpha)M}{\alpha M} = 1 + \frac{1}{2\alpha}$
- 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 $\frac{e}{e-1} \approx 1.58$.

Online vs. Offline Algorithms

Online vs. Offline Algorithms

Online Algorithm

Online vs. Offline Algorithms

Online Algorithm

- No full information available initally (*online problem*)

Online vs. Offline Algorithms

Online Algorithm

- No full information available initally (*online problem*)
- Decisions are made with incomplete information.

Online vs. Offline Algorithms

Online Algorithm

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

Online vs. Offline Algorithms

Online Algorithm

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

Offline Algorithm

Online vs. Offline Algorithms

Online Algorithm

- No full information available initally (*online problem*)
- Decisions are made with incomplete information.

Offline Algorithm

- Full information available initally (*offline problem*)

- The algorithm may get more informations over time or by exploring the problem instance.

Online vs. Offline Algorithms

Online Algorithm

- No full information available initally (*online problem*)
- Decisions are made with incomplete information.

Offline Algorithm

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

- The algorithm may get more informations over time or by exploring the problem instance.

Online vs. Offline Algorithms

Online Algorithm

- No full information available initally (*online problem*)
- Decisions are made with incomplete information.

Offline Algorithm

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

- The algorithm may get more informations over time or by exploring the problem instance.

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

Online vs. Offline Algorithms

Online Algorithm

- No full information available initally (*online problem*)
- Decisions are made with incomplete information.

Offline Algorithm

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

- The algorithm may get more informations over time or by exploring the problem instance.

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

Online vs. Offline Algorithms

Online Algorithm

- No full information available initally (*online problem*)
- Decisions are made with incomplete information.

Offline Algorithm

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

- The algorithm may get more informations over time or by exploring the problem instance.

in the w. c. (determ. algo.)

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

Online vs. Offline Algorithms

Online Algorithm

- No full information available initally (*online problem*)
- Decisions are made with incomplete information.

Offline Algorithm

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

- The algorithm may get more informations over time or by exploring the problem instance.

in the w. c. (determ. algo.)

in the worst avg. c. (random. algo.)

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

Online vs. Offline Algorithms

Online Algorithm

- No full information available initally (*online problem*)
- Decisions are made with incomplete information.

Offline Algorithm

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

- The algorithm may get more informations over time or by exploring the problem instance.

in the w. c. (determ. algo.)

in the worst avg. c. (random. algo.)

- The objective value of the returned solution divided by the obj. v. of an optimal [offline] solution is the *competitive ratio*.
- Examples (problems & algos.):

Online vs. Offline Algorithms

Online Algorithm

- No full information available initally (*online problem*)
- Decisions are made with incomplete information.

Offline Algorithm

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

- The algorithm may get more informations over time or by exploring the problem instance.

in the w. c. (determ. algo.)

in the worst avg. c. (random. algo.)

- The objective value of the returned solution divided by the obj. v. 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, Paging (replacing entries in a memory), Insertion Sort

Online vs. Offline Algorithms

Online Algorithm

- No full information available initally (*online problem*)
- Decisions are made with incomplete information.

Offline Algorithm

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

- The algorithm may get more informations over time or by exploring the problem instance.

in the w. c. (determ. algo.)

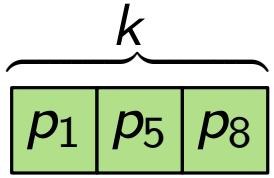
in the worst avg. c. (random. algo.)

- The objective value of the returned solution divided by the obj. v. 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, Paging (replacing entries in a memory), Insertion Sort

Paging

Given (offline/online):

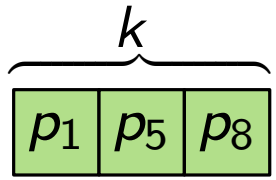
Paging



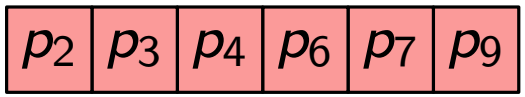
Given (offline/online):

- Fast access memory (a cache) with a capacity of k pages

Paging

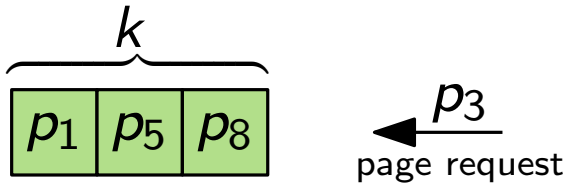


Given (offline/online):

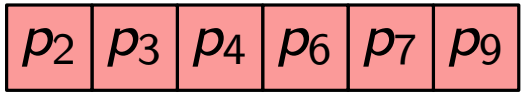


- Fast access memory (a cache) with a capacity of k pages
- Slow access memory with unlimited capacity

Paging

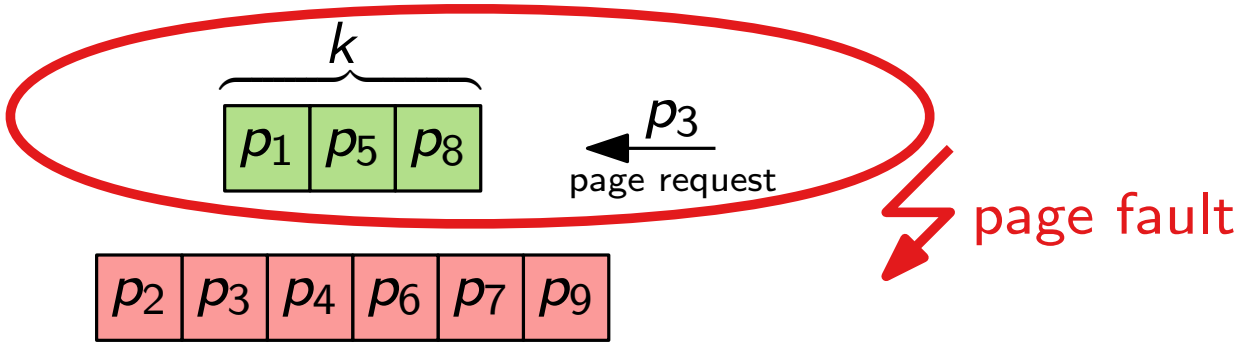


Given (offline/online):



- 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.

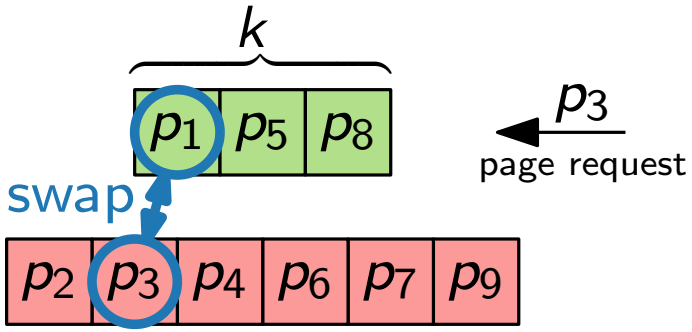
Paging



Given (offline/online): p_2 p_3 p_4 p_6 p_7 p_9

- 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.

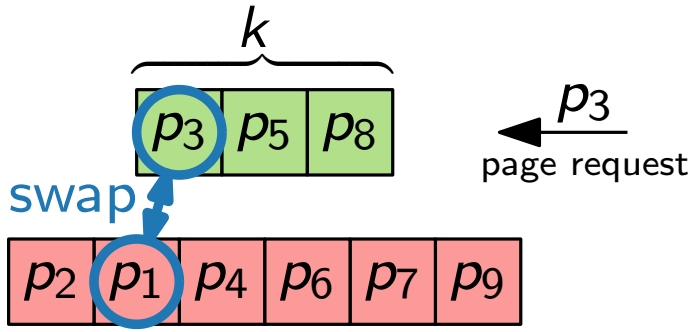
Paging



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.

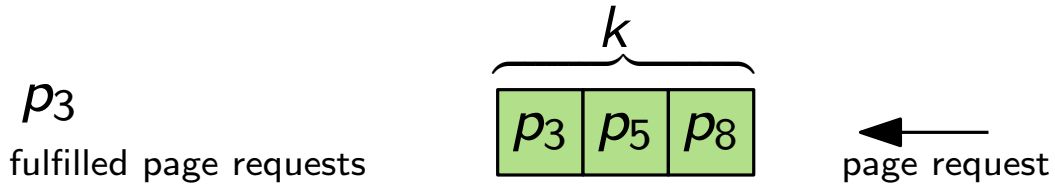
Paging



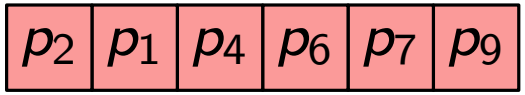
Given (offline/online):

- 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

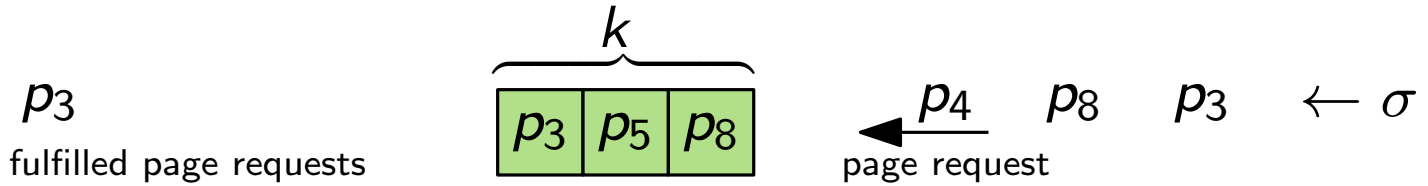


Given (offline/online):

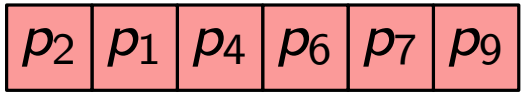


- Fast access memory (a cache) with a capacity of k pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (*page fault*), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.

Paging

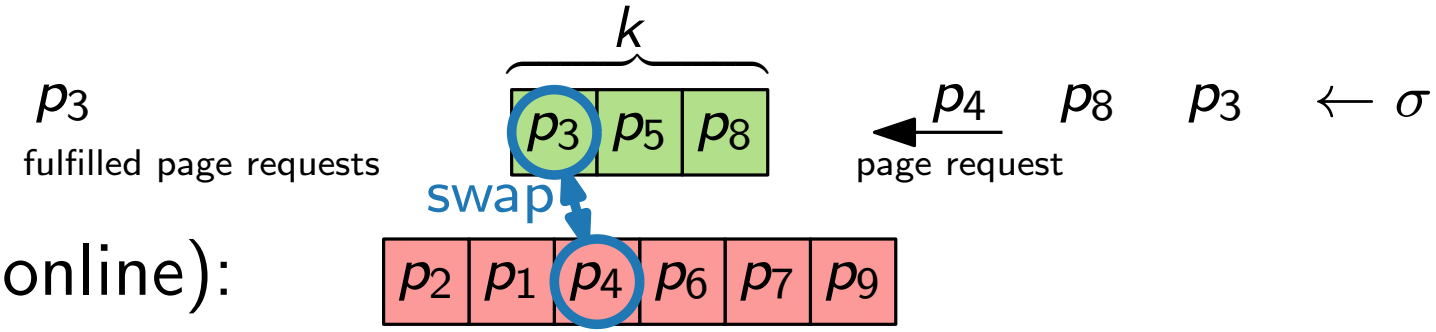


Given (offline/online):



- 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 σ of page requests having to be fulfilled in order/ we have to fulfill a request before we see the next request.

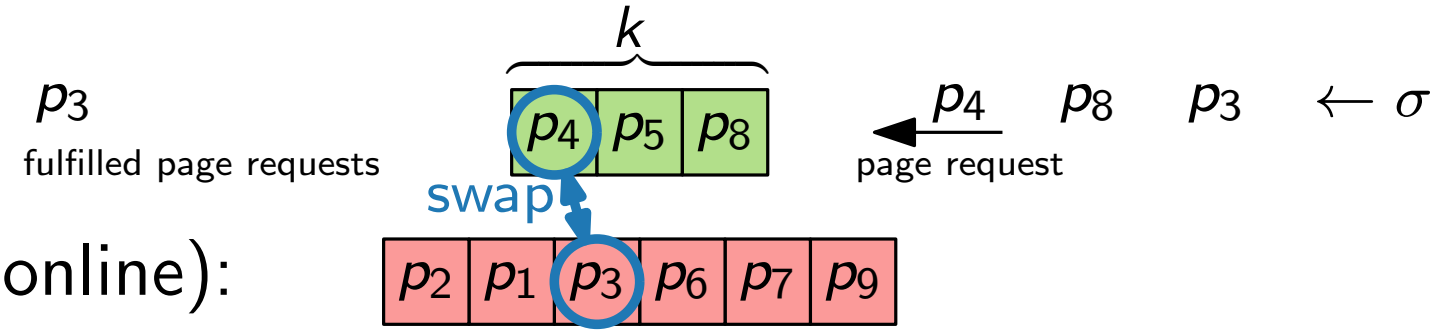
Paging



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 σ of page requests having to be fulfilled in order/ we have to fulfill a request before we see the next request.

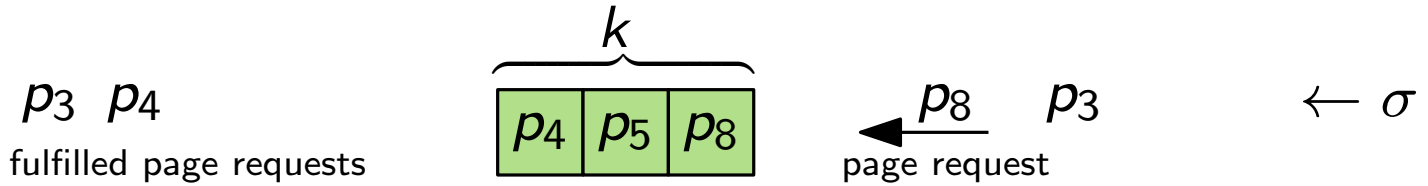
Paging



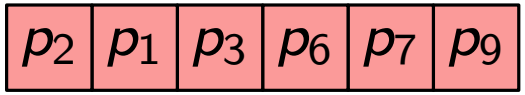
Given (offline/online):

- 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 σ of page requests having to be fulfilled in order/ we have to fulfill a request before we see the next request.

Paging

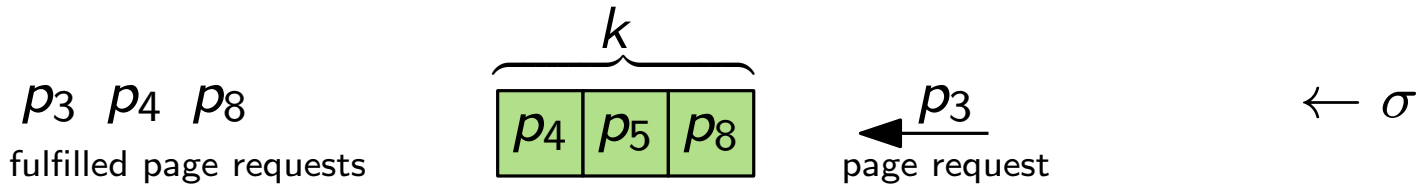


Given (offline/online):

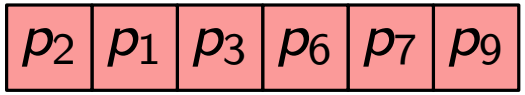


- Fast access memory (a cache) with a capacity of k pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (*page fault*), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
- Sequence σ of page requests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

Paging

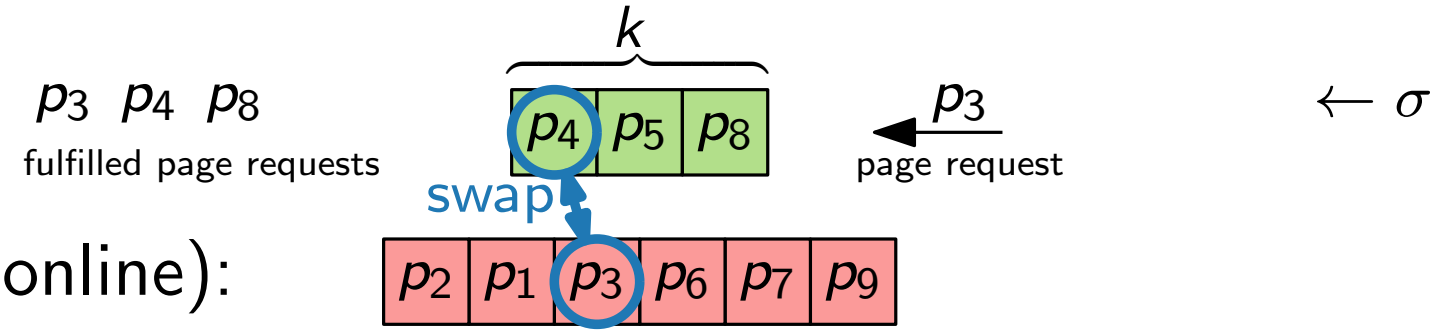


Given (offline/online):



- 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 σ of page requests having to be fulfilled in order/ we have to fulfill a request before we see the next request.

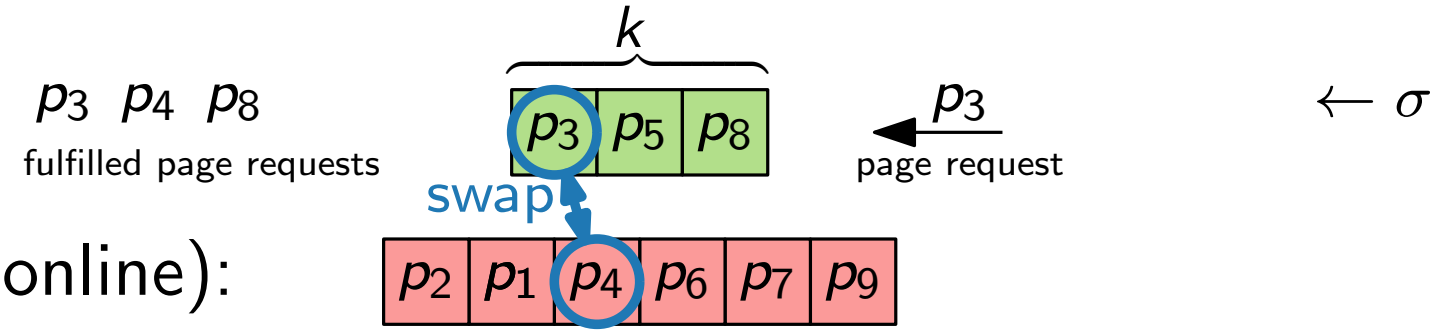
Paging



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 σ of page requests having to be fulfilled in order/ we have to fulfill a request before we see the next request.

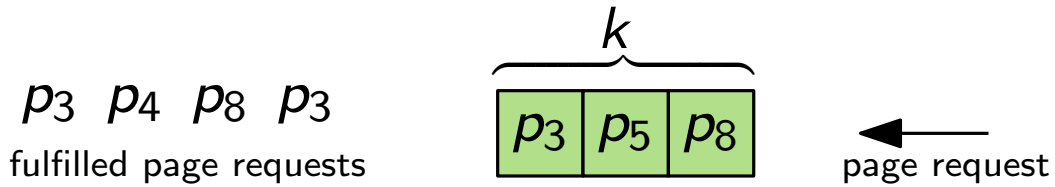
Paging



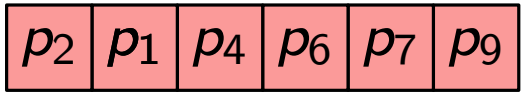
Given (offline/online):

- 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 σ of page requests having to be fulfilled in order/ we have to fulfill a request before we see the next request.

Paging

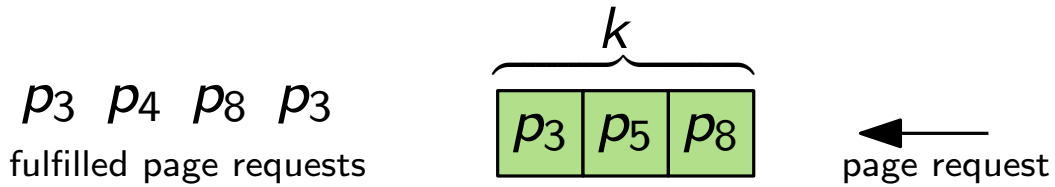


Given (offline/online):

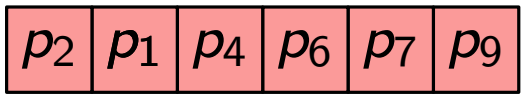


- Fast access memory (a cache) with a capacity of k pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (*page fault*), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
- Sequence σ of page requests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

Paging



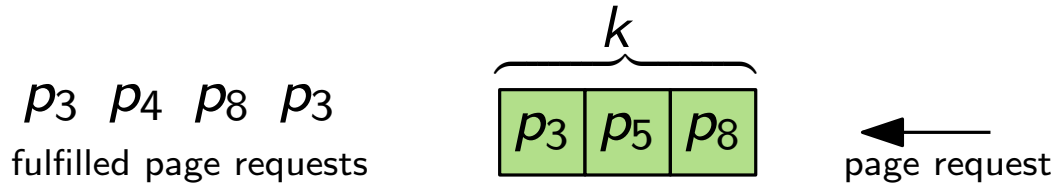
Given (offline/online):



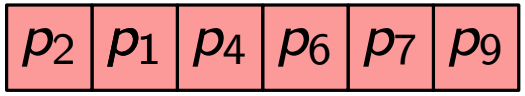
- Fast access memory (a cache) with a capacity of k pages
- Slow access memory with unlimited capacity
- If a page is requested, but it is not in the cache (*page fault*), it has to be swapped with a page in the cache. A page request is fulfilled if the page is in the cache.
- Sequence σ of page requests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

Objective value:

Paging



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 σ 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 σ.

Paging

When a Paging algorithm has to do a swap, it can choose which page it evicts from the cache.

Paging

When a Paging algorithm has to do a swap, it can choose which page it evicts from the cache.

Deterministic Strategies: Evict the page that ...

Paging

When a Paging algorithm has to do a swap, it can choose which page it evicts from the cache.

Deterministic Strategies: Evict the page that ...

- Least Frequently Used (LFU): ... has the lowest number of accesses since it was loaded.

Paging

When a Paging algorithm has to do a swap, it can choose which page it evicts from the cache.

Deterministic Strategies: Evict the page that ...

- Least Frequently Used (LFU): ... has the lowest number of accesses since it was loaded.
- Least Recently Used (LRU): ... was accessed least recently.

Paging

When a Paging algorithm has to do a swap, it can choose which page it evicts from the cache.

Deterministic Strategies: Evict the page that ...

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

Paging

When a Paging algorithm has to do a swap, it can choose which page it evicts from the cache.

Deterministic Strategies: Evict the page that ...

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

Which of them is—theoretically provable—the best strategy?

Paging

When a Paging algorithm has to do a swap, it can choose which page it evicts from the cache.

Deterministic Strategies: Evict the page that ...

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

Which of them is—theoretically provable—the best strategy?

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

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

- MIN: optimal offline strategy
  σ: arbitrary sequence of pages

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

- MIN: optimal offline strategy
  σ: arbitrary sequence of pages

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

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

- MIN: optimal offline strategy
  σ: arbitrary sequence of pages

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

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

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

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

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

- MIN: optimal offline strategy
  σ: arbitrary sequence of pages
- Initially, the cache contains the same pages for all strategies.
- We partition σ into phases $P_0, P_1, \ldots,$ s.t. LRU has at most k faults in P_0 and exactly k faults in every other phase.
- We show next: MIN has at least 1 fault in every phase.
- Clearly, MIN also faults in P_0; consider P_i with $i \geq 1$

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

- We show next: MIN has at least 1 fault in every phase.

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

- We show next: MIN has at least 1 fault in every phase.
- p: last page of P_{i-1}; We show that P_i contains k distinct page requests different from p (implies ≥ 1 fault for MIN).

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

- We show next: MIN has at least 1 fault in every phase.
- p: last page of P_{i-1}; We show that P_i contains k distinct page requests different from p (implies ≥ 1 fault for MIN).
- If the k page faults of LRU in P_i are on distinct pages (all different from p), we're done.

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

- We show next: MIN has at least 1 fault in every phase.
- p: last page of P_{i-1}; We show that P_i contains k distinct page requests different from p (implies ≥ 1 fault for MIN).
- If the k page faults of LRU in P_i are on distinct pages (all different from p), we're done.
- Assume LRU has in P_i two page faults on the same page q. In between, q has to be evicted from the cache. According to LRU, there had to be k distinct page requests in between.

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

- We show next: MIN has at least 1 fault in every phase.
- p: last page of P_{i-1}; We show that P_i contains k distinct page requests different from p (implies ≥ 1 fault for MIN).
- If the k page faults of LRU in P_i are on distinct pages (all different from p), we're done.
- Assume LRU has in P_i two page faults on the same page q. In between, q has to be evicted from the cache. According to LRU, there had to be k distinct page requests in between.
- Similarly, if LRU has a page fault on p in P_i, there had to be k distinct page requests in between.

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

- Remains to prove: No det. strategy is better than k-compet.

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

- Remains to prove: No det. strategy is better than k-compet.
- Let there be $k + 1$ pages in the memory system.

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

- Remains to prove: No det. strategy is better than k-compet.
- Let there be $k + 1$ pages in the memory system.
- For every det. strategy there's a worst-case page sequence σ^* always requesting the page that is currently not in the cache.

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

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

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

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

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

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

$\Rightarrow$ The comp. ratio cannot be better than $\frac{|\sigma^*|}{\left\lceil \frac{|\sigma^*|}{k} \right\rceil} \overset{|\sigma^*| \rightsquigarrow \infty}{=} k$.

Paging

Theorem: LRU & FIFO are k-compet. No det. strategy is better.

Proof (only for LRU, FIFO similar):

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

$\Rightarrow$ The comp. ratio cannot be better than $\frac{|\sigma^*|}{\left\lceil \frac{|\sigma^*|}{k} \right\rceil} \overset{|\sigma^*| \rightsquigarrow \infty}{=} k$. $\square$

Paging

Randomized strategy MARKING:

Paging

Randomized strategy MARKING:

- Proceeds in phases

Paging

Randomized strategy MARKING:

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

Paging

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

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 u.a.r. from the unmarked pages.

Paging

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 u.a.r. from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all.

Paging

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 u.a.r. from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all.

Theorem: Marking is $2H_k$-competitive.

Paging

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 u.a.r. from the unmarked pages.
- If all pages are marked and a page fault occurs, unmark all.

Theorem: 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 $k \geq 2$ it holds that $H_k < \ln(k) + 1$.

Paging

Theorem: Marking is $2H_k$-competitive.

Proof:

Paging

Theorem: Marking is $2H_k$-competitive.

Proof: We consider phase P_i. Definitions:

- S_{MARK} (S_{MIN}): set of pages in the cache of Marking (MIN)
- d_{begin}: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the beginning of P_i
- d_{end}: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the end of P_i
- 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.
- c: number of clean pages requested in P_i

Paging

Theorem: Marking is $2H_k$-competitive.

Proof: We consider phase P_i. Definitions:

- S_{MARK} (S_{MIN}): set of pages in the cache of Marking (MIN)
- d_{begin}: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the beginning of P_i
- d_{end}: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the end of P_i
- 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.
- c: number of clean pages requested in P_i

- MIN has at least $\max(c - d_{\text{begin}}, d_{\text{end}}) \geq \frac{1}{2}(c - d_{\text{begin}} + d_{\text{end}})$ $= \frac{c}{2} - \frac{d_{\text{begin}}}{2} + \frac{d_{\text{end}}}{2}$ faults. Over all phases, all $\frac{d_{\text{begin}}}{2}$ and $\frac{d_{\text{end}}}{2}$ cancel out, except for the first $\frac{d_{\text{begin}}}{2}$ and the last $\frac{d_{\text{end}}}{2}$.

Paging

Theorem: Marking is $2H_k$-competitive.

Proof: We consider phase P_i. Definitions:

- S_{MARK} (S_{MIN}): set of pages in the cache of Marking (MIN)
- d_{begin}: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the beginning of P_i
- d_{end}: $|S_{\text{MIN}} - S_{\text{MARK}}|$ at the end of P_i
- 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.
- c: number of clean pages requested in P_i

- MIN has at least $\max(c - d_{\text{begin}}, d_{\text{end}}) \geq \frac{1}{2}(c - d_{\text{begin}} + d_{\text{end}})$ $= \frac{c}{2} - \frac{d_{\text{begin}}}{2} + \frac{d_{\text{end}}}{2}$ faults. Over all phases, all $\frac{d_{\text{begin}}}{2}$ and $\frac{d_{\text{end}}}{2}$ cancel out, except for 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

Theorem: Marking is $2H_k$-competitive.

Proof:

Paging

Theorem: Marking is $2H_k$-competitive.

Proof:

- For the clean pages of a phase, Marking has c faults.

Paging

Theorem: Marking is $2H_k$-competitive.

Proof:

- For the clean pages of a phase, Marking has c faults.
- Moreover, in this phase, there are $s = k - c \leq k - 1$ requests to stale pages; for requests $j = 1, \ldots, s$ to stale pages we compute the expected number of faults $E[F_j]$.

Paging

Theorem: Marking is $2H_k$-competitive.

Proof:

- For the clean pages of a phase, Marking has c faults.
- Moreover, in this phase, there are $s = k - c \leq k - 1$ requests to stale pages; for requests $j = 1, \ldots, s$ to stale pages we compute 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

Theorem: Marking is $2H_k$-competitive.

Proof:

- For the clean pages of a phase, Marking has c faults.
- Moreover, in this phase, there are $s = k - c \leq k - 1$ requests to stale pages; for requests $j = 1, \ldots, s$ to stale pages we compute 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
- $E[F_j] = \frac{s(j)-c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \leq \frac{c}{k+1-j}$

$s(j) = k + 1 - j$

Paging

Theorem: Marking is $2H_k$-competitive.

Proof:

- For the clean pages of a phase, Marking has c faults.
- Moreover, in this phase, there are $s = k - c \leq k - 1$ requests to stale pages; for requests $j = 1, \ldots, s$ to stale pages we compute 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
- $E[F_j] = \frac{s(j)-c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \overset{s(j)=k+1-j}{\leq} \frac{c}{k+1-j}$
- $\sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j} \leq \sum_{j=2}^{k} \frac{c}{j} = c \cdot (H_k - 1)$

Paging

Theorem: Marking is $2H_k$-competitive.

Proof:

- For the clean pages of a phase, Marking has c faults.
- Moreover, in this phase, there are $s = k - c \leq k - 1$ requests to stale pages; for requests $j = 1, \ldots, s$ to stale pages we compute 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
- $E[F_j] = \frac{s(j)-c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \overset{s(j)=k+1-j}{\leq} \frac{c}{k+1-j}$
- $\sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j} \leq \sum_{j=2}^{k} \frac{c}{j} = c \cdot (H_k - 1)$
- So the competitive ratio of Marking is $\frac{c+c(H_k-1)}{c/2} = 2H_k$.

Paging

Theorem: Marking is $2H_k$-competitive.

Proof:

- For the clean pages of a phase, Marking has c faults.
- Moreover, in this phase, there are $s = k - c \leq k - 1$ requests to stale pages; for requests $j = 1, \ldots, s$ to stale pages we compute 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
- $E[F_j] = \frac{s(j)-c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \overset{s(j)=k+1-j}{\leq} \frac{c}{k+1-j}$
- $\sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j} \leq \sum_{j=2}^{k} \frac{c}{j} = c \cdot (H_k - 1)$
- So the competitive ratio of Marking is $\frac{c+c(H_k-1)}{c/2} = 2H_k$. $\square$

Paging

Reminder: No det. strategy is better than k-competitive.

Theorem: Marking is $2H_k$-competitive.

Proof:

- For the clean pages of a phase, Marking has c faults.
- Moreover, in this phase, there are $s = k - c \leq k - 1$ requests to stale pages; for requests $j = 1, \ldots, s$ to stale pages we compute 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
- $E[F_j] = \frac{s(j)-c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \overset{s(j)=k+1-j}{\leq} \frac{c}{k+1-j}$
- $\sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j} \leq \sum_{j=2}^{k} \frac{c}{j} = c \cdot (H_k - 1)$
- So the competitive ratio of Marking is $\frac{c+c(H_k-1)}{c/2} = 2H_k$. $\square$

Paging

Reminder: No det. strategy is better than k-competitive.

$\Rightarrow$ **Randomization helps!**

Theorem: Marking is $2H_k$-competitive.

Proof:

- For the clean pages of a phase, Marking has c faults.
- Moreover, in this phase, there are $s = k - c \leq k - 1$ requests to stale pages; for requests $j = 1, \ldots, s$ to stale pages we compute 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
- $E[F_j] = \frac{s(j)-c(j)}{s(j)} \cdot 0 + \frac{c(j)}{s(j)} \cdot 1 \leq \frac{c}{k+1-j}$ $\quad s(j) = k + 1 - j$
- $\sum_{j=1}^{s} E[F_j] \leq \sum_{j=1}^{s} \frac{c}{k+1-j} \leq \sum_{j=2}^{k} \frac{c}{j} = c \cdot (H_k - 1)$
- So the competitive ratio of Marking is $\frac{c+c(H_k-1)}{c/2} = 2H_k$. $\square$