COT 6936: Topics in Algorithms

Giri Narasimhan

ECS 254A / EC 2474; Phone x3748; Email: giri@cs.fiu.edu
HOMEPAGE: http://www.cs.fiu.edu/~giri
https://moodle.cis.fiu.edu/v2.1/course/view.php?id=612

Feb 18, 2014

Presentation Outline

What is a Randomized Online Algorithm?

What is a Randomized Online Algorithm?

- It is an online algorithm with randomization

What is a Randomized Online Algorithm?

- It is an online algorithm with randomization
- Example: If you are going on a ski trip, toss a coin and decide whether to rent/buy skis.

What is a Randomized Online Algorithm?

- It is an online algorithm with randomization
- Example: If you are going on a ski trip, toss a coin and decide whether to rent/buy skis.
- Example: When a new item is brought into cache, randomly pick an existing item to evict.

What is a Randomized Online Algorithm?

- It is an online algorithm with randomization
- Example: If you are going on a ski trip, toss a coin and decide whether to rent/buy skis.
- Example: When a new item is brought into cache, randomly pick an existing item to evict.
- Lower bounds for deterministic online algorithms do not apply.

What is a Randomized Online Algorithm?

- It is an online algorithm with randomization
- Example: If you are going on a ski trip, toss a coin and decide whether to rent/buy skis.
- Example: When a new item is brought into cache, randomly pick an existing item to evict.
- Lower bounds for deterministic online algorithms do not apply.
- Lower bound of $k =$ size of cache does not apply for randomized cache replacement strategies.

How to Analyze Online Algorithms?

- Competitive analysis

How to Analyze Online Algorithms?

- Competitive analysis
  - Compare with optimal offline algorithm (OPT)

How to Analyze Online Algorithms?

- Competitive analysis
    - Compare with optimal offline algorithm (OPT)
- Online Algorithm A is α-competitive if there exists constant b such that for every sequence of inputs σ:

$$COST_A(\sigma) \leq \alpha \cdot COST_{OPT}(\sigma) + b$$

How to Analyze Randomized Online Algorithms?

- Online Algorithm A is α-competitive if there exists constant b such that for every sequence of inputs σ:

$$COST_A(\sigma) \leq \alpha \cdot COST_{OPT}(\sigma) + b$$

How to Analyze Randomized Online Algorithms?

- Online Algorithm A is α-competitive if there exists constant b such that for every sequence of inputs σ:

$$COST_A(\sigma) \leq \alpha \cdot COST_{OPT}(\sigma) + b$$

- Randomized Online Algorithm A is α-competitive if there exists constant b such that for every sequence of inputs σ:

$$E[COST_A(\sigma)] \leq \alpha COST_{OPT}(\sigma) + b$$

How to Analyze Randomized Online Algorithms?

- Online Algorithm A is α-competitive if there exists constant b such that for every sequence of inputs σ:

$$COST_A(\sigma) \leq \alpha \cdot COST_{OPT}(\sigma) + b$$

- Randomized Online Algorithm A is α-competitive if there exists constant b such that for every sequence of inputs σ:

$$E[COST_A(\sigma)] \leq \alpha COST_{OPT}(\sigma) + b$$

Use Expected cost instead

Presentation Outline

RANDOM Algorithm

- On a miss: evict an item chosen uniformly at random from all k items in cache.

RANDOM Algorithm

- On a miss: evict an item chosen uniformly at random from all k items in cache.
- RANDOM is k-competitive

RANDOM Algorithm

- On a miss: evict an item chosen uniformly at random from all k items in cache.
- RANDOM is k-competitive
- Can we do better than the deterministic lower bound of k for the competitiveness?

MARKER Algorithm

MARKER Algorithm

- Each of k pages in cache has a MARKer bit

MARKER Algorithm

MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds

MARKER Algorithm

MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds
- Start of round:

MARKER Algorithm

MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds
- Start of round: UNMARK all pages

MARKER Algorithm

MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds
- Start of round: UNMARK all pages
- In each round:

MARKER Algorithm

MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds
- Start of round: UNMARK all pages
- In each round:
  - If request is a hit:

MARKER Algorithm

MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds
- Start of round: UNMARK all pages
- In each round:
    - If request is a hit: MARK the page

MARKER Algorithm

MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds
- Start of round: UNMARK all pages
- In each round:
    - If request is a hit: MARK the page
    - If request is a miss:

MARKER Algorithm

MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds
- Start of round: UNMARK all pages
- In each round:
  - If request is a hit: MARK the page
  - If request is a miss:
    1. Replace (arbitrary) UNMARKed page and MARK it

MARKER Algorithm

MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds
- Start of round: UNMARK all pages
- In each round:
  - If request is a hit: MARK the page
  - If request is a miss:
    1. Replace (arbitrary) UNMARKed page and MARK it
    2. If all pages are MARKed, start next round and UNMARK all pages
- No explicit randomization in this algorithm.

MARKER Algorithm

MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds
- Start of round: UNMARK all pages
- In each round:
  - If request is a hit: MARK the page
  - If request is a miss:
    1. Replace (arbitrary) UNMARKed page and MARK it
    2. If all pages are MARKed, start next round and UNMARK all pages
- No explicit randomization in this algorithm.
  - Arbitrary replacement can be implemented as say a FIFO or LIFO of unmarked items

MARKER Algorithm

MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds
- Start of round: UNMARK all pages
- In each round:
    - If request is a hit: MARK the page
    - If request is a miss:
        1. Replace (arbitrary) UNMARKed page and MARK it
        2. If all pages are MARKed, start next round and UNMARK all pages
- No explicit randomization in this algorithm.
    - Arbitrary replacement can be implemented as say a FIFO or LIFO of unmarked items
- LRU is in fact a MARKing algorithm.

MARKER Algorithm

MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds
- Start of round: UNMARK all pages
- In each round:
  - If request is a hit: MARK the page
  - If request is a miss:
    1. Replace (arbitrary) UNMARKed page and MARK it
    2. If all pages are MARKed, start next round and UNMARK all pages
- No explicit randomization in this algorithm.
  - Arbitrary replacement can be implemented as say a FIFO or LIFO of unmarked items
- LRU is in fact a MARKing algorithm. Why?

Randomized MARKER Algorithm

Randomized MARKER Algorithm

Randomized MARKER Algorithm

Randomized MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds
- Start of round: UNMARK all pages
- In each round:
  - If request is a hit: MARK the page
  - If request is a miss:
    1. Replace ~~(arbitrary)~~ UNMARKed page chosen uniformly at random and MARK it

Randomized MARKER Algorithm

Randomized MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds
- Start of round: UNMARK all pages
- In each round:
    - If request is a hit: MARK the page
    - If request is a miss:
        1. Replace ~~(arbitrary)~~ UNMARKed page chosen uniformly at random and MARK it
        2. If all pages are MARKed, start next round and UNMARK all pages

Randomized MARKER Algorithm

Randomized MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds
- Start of round: UNMARK all pages
- In each round:
  - If request is a hit: MARK the page
  - If request is a miss:
    1. Replace ~~(arbitrary)~~ UNMARKed page chosen uniformly at random and MARK it
    2. If all pages are MARKed, start next round and UNMARK all pages
- Has explicit randomization in this algorithm.

Randomized MARKER Algorithm

Randomized MARKER Algorithm

- Each of k pages in cache has a MARKer bit
- Algorithm proceeds in rounds
- Start of round: UNMARK all pages
- In each round:
  - If request is a hit: MARK the page
  - If request is a miss:
    1. Replace ~~(arbitrary)~~ UNMARKed page chosen uniformly at random and MARK it
    2. If all pages are MARKed, start next round and UNMARK all pages
- Has explicit randomization in this algorithm.
- Analysis: This algorithm is $2H_k$-competitive.

Presentation Outline

Analyzing Randomized MARKER

Divide the request sequence into phases $\sigma(i), \ldots, \sigma(j)$,

Analyzing Randomized MARKER

Divide the request sequence into phases $\sigma(i), \ldots, \sigma(j)$, where j is the smallest integer such that

Analyzing Randomized MARKER

Divide the request sequence into phases $\sigma(i), \ldots, \sigma(j)$, where j is the smallest integer such that

$$\{\sigma(i), \ldots, \sigma(j+1)\}$$

contains $k+1$ distinct pages.
At the end of a phase, all pages are MARKed.

Analyzing Randomized MARKER

Divide the request sequence into phases $\sigma(i), \ldots, \sigma(j)$, where j is the smallest integer such that

$$\{\sigma(i), \ldots, \sigma(j+1)\}$$

contains $k+1$ distinct pages.
At the end of a phase, all pages are MARKed.
A page is stale if it is unMARKed but was MAKRKed in previous phase. A page is clean if it is neither stale nor MARKed. Let $c = \#$ of clean pages requested in phase.
Claim 1: Amortized $\#$ of faults by OPT in phase is $\geq c/2$
Claim 2: Expected number of faults by MARKER is $\leq cH_k$.

Analyzing Randomized MARKER

OPT: Let the # of pages in $S_{OPT} \backslash S_M$ be d_I at start of phase

Analyzing Randomized MARKER

OPT: Let the # of pages in $S_{OPT} \setminus S_M$ be d_I at start of phase and d_F at end of phase.

Analyzing Randomized MARKER

OPT: Let the # of pages in $S_{OPT} \setminus S_M$ be d_I at start of phase and d_F at end of phase. OPT has at least $c - d_I$ faults.

Analyzing Randomized MARKER

OPT: Let the # of pages in $S_{OPT} \backslash S_M$ be d_I at start of phase and d_F at end of phase. OPT has at least $c - d_I$ faults.Also OPT has at least d_F faults.

Analyzing Randomized MARKER

OPT: Let the # of pages in $S_{OPT} \backslash S_M$ be d_I at start of phase and d_F at end of phase. OPT has at least $c - d_I$ faults.Also OPT has at least d_F faults. Why?

Analyzing Randomized MARKER

OPT: Let the # of pages in $S_{OPT} \setminus S_M$ be d_I at start of phase and d_F at end of phase. OPT has at least $c - d_I$ faults.Also OPT has at least d_F faults. Why?
A clean page is a miss for MARKER. Thus c clean pages are not in S_M. At best, they are from $S_{OPT} \setminus S_M$. But $|S_{OPT} \setminus S_M| = d_I$. Hence OPT has $\geq c - d_I$ misses.

Analyzing Randomized MARKER

OPT: Let the # of pages in $S_{OPT} \backslash S_M$ be d_I at start of phase and d_F at end of phase. OPT has at least $c - d_I$ faults.Also OPT has at least d_F faults. Why?
A clean page is a miss for MARKER. Thus c clean pages are not in S_M. At best, they are from $S_{OPT} \backslash S_M$. But $|S_{OPT} \backslash S_M| = d_I$. Hence OPT has $\geq c - d_I$ misses.
The d_F pages from $S_M \backslash S_{OPT}$ were accessed, but not present at end of phase in S_{OPT}. Hence they were evicted by OPT on some miss. Thus OPT had at least d_F misses.

$$\max\{c - d_I, d_F\} \geq (c - d_I + d_F)/2 = c/2 - d_I/2 + d_F/2$$

Amortized over all requests, the second and last terms start to cancel off, giving us $\geq c/2$ faults.

Analyzing Randomized MARKER

OPT: Let the # of pages in $S_{OPT} \backslash S_M$ be d_I at start of phase and d_F at end of phase. OPT has at least $c - d_I$ faults.Also OPT has at least d_F faults. Why?
A clean page is a miss for MARKER. Thus c clean pages are not in S_M. At best, they are from $S_{OPT} \backslash S_M$. But $|S_{OPT} \backslash S_M| = d_I$. Hence OPT has $\geq c - d_I$ misses.
The d_F pages from $S_M \backslash S_{OPT}$ were accessed, but not present at end of phase in S_{OPT}. Hence they were evicted by OPT on some miss. Thus OPT had at least d_F misses.

$$\max\{c - d_I, d_F\} \geq (c - d_I + d_F)/2 = c/2 - d_I/2 + d_F/2$$

Amortized over all requests, the second and last terms start to cancel off, giving us $\geq c/2$ faults.

Analyzing Randomized MARKER

OPT: $\geq c/2$ faults.

Analyzing Randomized MARKER

OPT: $\geq c/2$ faults.
MARKER: $k - c$ stale pages in cache. Let $c(i)$ (and $s(i)$) be the number of clean (and stale, resp.) pages requested before the i-th stale page.
Thus, expected cost of request is

$$\frac{s(i) - c(i)}{s(i)} \cdot 0 + \frac{c(i)}{s(i)} \cdot 1 \leq \frac{c}{s(i)} = \frac{c}{k - i + 1}$$

When summed over all iterations, we have

$$\sum_{i=1}^{s} \frac{c}{k - i + 1} \leq \sum_{i=2}^{k} \frac{c}{i} = c(H_k - 1)$$

Analyzing Randomized MARKER

Claim 1: Amortized # of faults by OPT in phase is $\geq c/2$
Claim 2: Expected number of faults by MARKER is $\leq cH_k$.

Analyzing Randomized MARKER

Claim 1: Amortized # of faults by OPT in phase is $\geq c/2$
Claim 2: Expected number of faults by MARKER is $\leq cH_k$.
Claim 3: Randomized MARKER is $2H_k$-competitive.
Claim 4: If R is a randomized online paging algorithm that is c-competitive against any oblivious adversary, then $c \geq H_k$.

Presentation Outline

Adversary Models

- When doing worst-case analysis of online, we assume that

Adversary Models

- When doing worst-case analysis of online, we assume that there is an adversary

Adversary Models

- When doing worst-case analysis of online, we assume that there is an adversary who is generating a request sequence in order to make algorithm perform as poorly as possible.

Adversary Models

- When doing worst-case analysis of online, we assume that there is an adversary who is generating a request sequence in order to make algorithm perform as poorly as possible. Adversary is powerful – knows your algorithm!
- In analyzing randomized online algs, we have 3 choices:

Adversary Models

- When doing worst-case analysis of online, we assume that there is an adversary who is generating a request sequence in order to make algorithm perform as poorly as possible. Adversary is powerful – knows your algorithm!
- In analyzing randomized online algs, we have 3 choices:
  - **Oblivious adversary**: adversary generates request sequence at start. I.e.,

Adversary Models

- When doing worst-case analysis of online, we assume that there is an adversary who is generating a request sequence in order to make algorithm perform as poorly as possible. Adversary is powerful – knows your algorithm!
- In analyzing randomized online algs, we have 3 choices:
  - **Oblivious adversary**: adversary generates request sequence at start. I.e., cannot see action of algorithm or random choices.

Adversary Models

- When doing worst-case analysis of online, we assume that there is an adversary who is generating a request sequence in order to make algorithm perform as poorly as possible. Adversary is powerful – knows your algorithm!
- In analyzing randomized online algs, we have 3 choices:
  - **Oblivious adversary**: adversary generates request sequence at start. I.e., cannot see action of algorithm or random choices. Adversary serves offline.

Adversary Models

- When doing worst-case analysis of online, we assume that there is an adversary who is generating a request sequence in order to make algorithm perform as poorly as possible. Adversary is powerful – knows your algorithm!
- In analyzing randomized online algs, we have 3 choices:
  - **Oblivious adversary**: adversary generates request sequence at start. I.e., cannot see action of algorithm or random choices. Adversary serves offline.
  - **Adaptive online adversary**: adversary generates request sequence adaptively (online). I.e.,

Adversary Models

- When doing worst-case analysis of online, we assume that there is an adversary who is generating a request sequence in order to make algorithm perform as poorly as possible. Adversary is powerful – knows your algorithm!
- In analyzing randomized online algs, we have 3 choices:
  - **Oblivious adversary**: adversary generates request sequence at start. I.e., cannot see action of algorithm or random choices. Adversary serves offline.
  - **Adaptive online adversary**: adversary generates request sequence adaptively (online). I.e., can see action, but not random choices. Adversary serves online

Adversary Models

- When doing worst-case analysis of online, we assume that there is an adversary who is generating a request sequence in order to make algorithm perform as poorly as possible. Adversary is powerful – knows your algorithm!
- In analyzing randomized online algs, we have 3 choices:
  - **Oblivious adversary**: adversary generates request sequence at start. I.e., cannot see action of algorithm or random choices. Adversary serves offline.
  - **Adaptive online adversary**: adversary generates request sequence adaptively (online). I.e., can see action, but not random choices. Adversary serves online
  - **Adaptive oine adversary**: adversary generates request sequence adaptively, and knows the result of the coin tosses.

Adversary Models

- When doing worst-case analysis of online, we assume that there is an adversary who is generating a request sequence in order to make algorithm perform as poorly as possible. Adversary is powerful – knows your algorithm!
- In analyzing randomized online algs, we have 3 choices:
    - **Oblivious adversary**: adversary generates request sequence at start. I.e., cannot see action of algorithm or random choices. Adversary serves offline.
    - **Adaptive online adversary**: adversary generates request sequence adaptively (online). I.e., can see action, but not random choices. Adversary serves online
    - **Adaptive oine adversary**: adversary generates request sequence adaptively, and knows the result of the coin tosses. Adversary serves offline

Relationship between Adversary Models

- $\exists$ randomized c-competitive algorithm against *adaptive offline adversary* $\Longrightarrow$

Relationship between Adversary Models

- $\exists$ randomized c-competitive algorithm against *adaptive offline adversary* $\implies$ $\exists$ deterministic c-competitive algorithm.

Relationship between Adversary Models

- $\exists$ randomized c-competitive algorithm against *adaptive offline adversary* $\implies$ $\exists$ deterministic c-competitive algorithm.
- $\exists$ randomized c-competitive algorithm against *adaptive online adversary* and

Relationship between Adversary Models

- $\exists$ randomized c-competitive algorithm against *adaptive offline adversary* $\implies$ $\exists$ deterministic c-competitive algorithm.
- $\exists$ randomized c-competitive algorithm against *adaptive online adversary* and $\exists$ d-competitive algorithm against *oblivious adversary* $\implies$

Relationship between Adversary Models

- $\exists$ randomized c-competitive algorithm against *adaptive offline adversary* $\implies$ $\exists$ deterministic c-competitive algorithm.
- $\exists$ randomized c-competitive algorithm against *adaptive online adversary* and $\exists$ d-competitive algorithm against *oblivious adversary* $\implies$ $\exists$ $c \cdot d$-competitive algorithm against any *adaptive offline adversary*.

Relationship between Adversary Models

- $\exists$ randomized c-competitive algorithm against *adaptive offline adversary* $\implies$ $\exists$ deterministic c-competitive algorithm.
- $\exists$ randomized c-competitive algorithm against *adaptive online adversary* and $\exists$ d-competitive algorithm against *oblivious adversary* $\implies$ $\exists$ $c \cdot d$-competitive algorithm against any *adaptive offline adversary*.
- $\exists$ randomized c-competitive algorithm against *adaptive online adversary* $\implies$

Relationship between Adversary Models

- $\exists$ randomized c-competitive algorithm against *adaptive offline adversary* $\implies$ $\exists$ deterministic c-competitive algorithm.
- $\exists$ randomized c-competitive algorithm against *adaptive online adversary* and $\exists$ d-competitive algorithm against *oblivious adversary* $\implies$ $\exists$ $c \cdot d$-competitive algorithm against any *adaptive offline adversary*.
- $\exists$ randomized c-competitive algorithm against *adaptive online adversary* $\implies$ $\exists$ deterministic c^2-competitive algorithm.

Relationship between Adversary Models

- $\exists$ randomized c-competitive algorithm against *adaptive offline adversary* $\implies$ $\exists$ deterministic c-competitive algorithm.
- $\exists$ randomized c-competitive algorithm against *adaptive online adversary* and $\exists$ d-competitive algorithm against *oblivious adversary* $\implies$ $\exists$ $c \cdot d$-competitive algorithm against any *adaptive offline adversary*.
- $\exists$ randomized c-competitive algorithm against *adaptive online adversary* $\implies$ $\exists$ deterministic c^2-competitive algorithm.

[Ben-David, Borodin, Karp, Tardos, Wigderson, Algorithmica, 1994]