Advanced Algorithm
Study Group

Online Algorithm

Presented by **Ting-An Chen**

Advisor: De-Nian Yang, Ming-Syan Chen

May 29, 2020

Outline

- Recap. Streaming v.s. Online algorithm
- Offline v.s. Online algorithm
- Online algorithm performance – competitiveness analysis
  - Case 1. Ski Rental Problem
  - Case 2. Deterministic Paging Problem
  - Case 3. Random Marking Algorithms
- Types of adversary

Recap. Streaming algorithm

Characteristics?

- Sequence of data
- Limited memory
- A sketch of data   Summary
- Return output after seeing all data
- All data info. is known
- OPT is known
- ➔ v.s. OPT approximation

What we want to know…
e.g. data properties –
[Recap.] "*Frequency moment*"
Zero-order: #distinct_values
1^{st} -order: total input size

Compared with … Online algorithm

- Sequence of data
- Limited memory
- A sketch of data
- Return output at each time stamp
- Never know the nature of the coming data
- OPT is unknown
- ➔ v.s. Offline OPT (known) *How to compare?*

Online algorithm (A) v.s. Offline OPT (OPT)

- Competitiveness analysis
  - $\frac{\text{Cost(A)}}{\text{Cost(OPT)}} \leq$ bound, then A is *competitive*.

  - $[Def.]\ \alpha - competitive\ online\ algorithm.$
  - $\sigma: an\ input\ sequence$
  - $c: a\ cost\ function$
  - ➔
  - $A\ is\ said\ to\ be\ \alpha - competitive\ \ if\ c_A(\sigma) \leq \alpha \cdot c_{OPT}\ (\sigma).$
  - $\alpha: competitive\ ratio.$

Case 1. Ski-rental problem

- Ski everyday
- Rent or buy the skiing equipment (daily decision)
  - Rent one day, $1.
  - Buy, $C.
- Assumption: might hurt each day then cannot ski.
- Let d be the total number of days skiing.
- Algorithm: "Rent for C days, then buy on (C+1)-th day."
  - $[pf.]$ $2-competitive$ $online\ algorithm,$ $\mathrm{i.e.},$ $c_A(\sigma) \le 2 \cdot c_{OPT}(\sigma)$.

case	$c_A(\sigma)$	$c_{OPT}(\sigma)$
If d ≤ C	**d**	**d**
If d > C	**2C**	**C**

Online algorithm (A) v.s. Offline OPT (OPT)

- Competitiveness analysis
  - $\frac{\text{Cost(A)}}{\text{Cost(OPT)}} \leq$ bound, then A is *competitive*.

  - $[Def.]\ \alpha - competitive\ online\ algorithm.$
  - $\sigma: an\ input\ sequence$
  - $c: a\ cost\ function$
  - ➔
  - $A\ is\ said\ to\ be\ \alpha - competitive\ \ if\ c_A(\sigma) \leq \alpha \cdot c_{OPT}\ (\sigma).$
  - $\alpha: competitive\ ratio.$

Approximation ratio?

Online v.s. Offline (traditional) algorithm

	Online	Offline
Compare to Offline OPT	Competitive ratio	Approximation ratio
Cost(A) related to… 1. Inputs	a. Unknown but expected b. Random w/o known patterns ➔ Online alg. Cost(A): ➔ fluctuate	a. Known b. Not random –OR- Random w/. Distribution ➔ Offline alg. Cost(A): ➔ stable
Cost(A) related to… 2. Algorithm	At different states… Same strategy - Deterministic Different strategies – Random	Single strategy
Inputs to the Alg.	Hard –OR- easy ➔ average case	Hard ➔ worst case

Studying worst case in Online algorithm?

- Adversary!!

To consider the case: the inputs make the algorithm worst
➔ competitive ratio

- Known:
  - 1. the online algorithm, i.e., the strategy at each output state
  - 2. the outcomes of events
  - 3. inputs in the past
- Unknown:
  - 1. inputs in the future

Adversary

- Generate the input sequence that may induce the worst case performance

Oblivious The adversary designs the input sequence σ at the beginning. It does not know any randomness used by algorithm $\mathcal{A}$.

Adaptive At each time step t, the adversary knows all randomness used by algorithm $\mathcal{A}$ thus far. In particular, it knows the exact state of the algorithm. With these in mind, it then picks the $(t+1)$-th element in the input sequence.

Fully adaptive The adversary knows all possible randomness that will be used by the algorithm $\mathcal{A}$ when running on the full input sequence σ. For instance, assume the adversary has access to the same pseudorandom number generator used by $\mathcal{A}$ and can invoke it arbitrarily many times while designing the adversarial input sequence σ.

Online v.s. Offline (traditional) algorithm

Adversarial inputs?

Random inputs

	Online	Offline
Compare to Offline OPT	Competitive ratio	Approximation ratio
Cost(A) related to… 1. Inputs	a. Unknown but expected b. Random w/o known patterns ➔ Online alg. Cost(A): ➔ fluctuate	a. Known b. Not random –OR- Random w/. Distribution ➔ Offline alg. Cost(A): ➔ stable
Cost(A) related to… 2. Algorithm	At different states… Same strategy - Deterministic Different strategies – Random	Single strategy
Inputs to the Alg.	Hard –OR- easy ➔ average case	Hard ➔ worst case

Online algorithm – random v.s. adversarial inputs

	Online (**random** inputs)	Online (**adversarial** inputs)	Offline
Compare to Offline OPT	Competitive ratio $\leq$	Competitive ratio	Approximation ratio
Cost(A) related to… 1. Inputs	a. Unknown but expected b. Random w/o known patterns ➔ Online alg. Cost(A): ➔ fluctuate	a. Known (simulated) b. Not random ➔ Online alg. Cost(A)': ➔ Stable ➔ Online alg. Cost(A) $\leq$ Cost(A)'	a. Known b. Not random –OR- Random w/. Distribution ➔ Offline alg. Cost(A): ➔ stable
Inputs to the Alg.	Hard –OR- easy ➔ average case	Hard ➔ worst case	Hard ➔ worst case

Example. Sorting

- Offline – Selection sort

Initial

1	5	2	3

Find 1st min.

1	5	2	3

Find 2nd min.

1	2	5	3

…

Final

1	2	3	5

- Online – Insertion sort

Initial

1

Comes 1st one, compare and insert

5

1	5

Comes 2nd one, compare and insert

2

1	2	5

…

Final

1	2	3	5

Case 2. Paging problem

- Hard disk – large memory, slow access
- Cache – small memory, fast access
- A sequence of page requests (from cache)
- *Page fault* if requested info. is not in cache
- ➔ access from hard disk
- ➔ large access costs
- Problem:
  - what data is to be stored in cache s.t. fewest page faults
  - more precisely, which data in cache is to be evicted when a new data is requested

最不近(最久之前) request 的，先 evict 拔除

Paging problem – Least Recently Used (LRU)

- Example

request|	cache elements|	page fault|	evicted item|
a	-,-,-	True	-
b	a,-,-	True	-
c	a,b,-	True	-
d	a,b,c	True	a
a	d,b,c	True	b
e	d,a,c	True	c
b	d,a,e	True	d
a	b,a,e	False	
c	b,a,e	True	e
e	b,a,c	True	b

__Deterministic Online Algorithm:__ With specific strategy

Paging problem

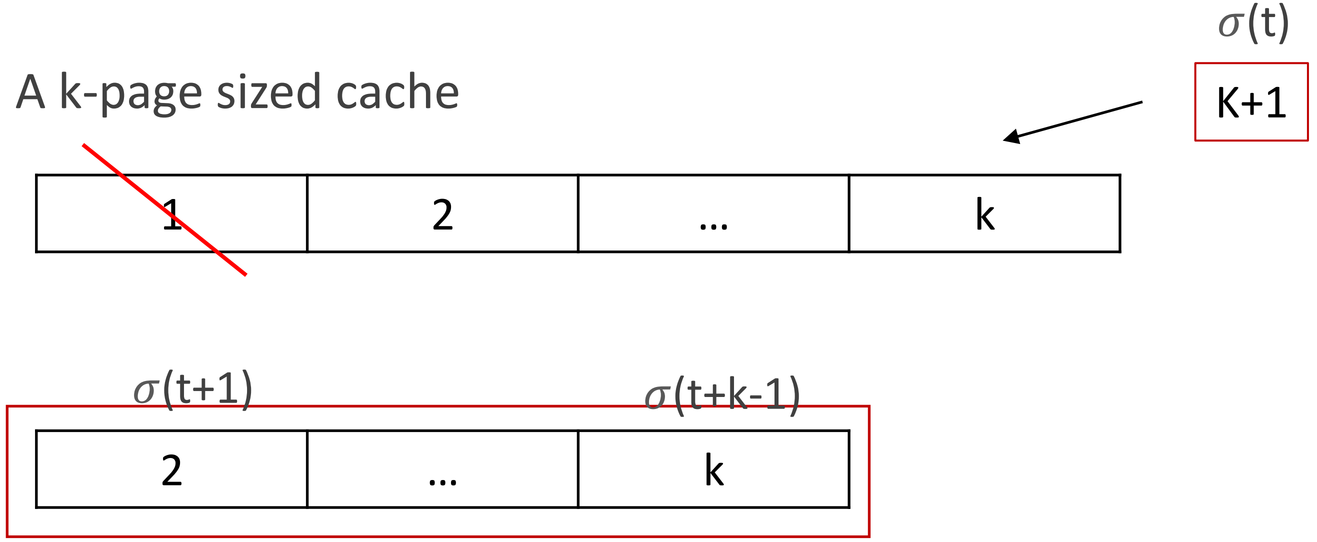
- Claim: If A is a deterministic online algorithm that is $\alpha - competitive$, then $\alpha \geq k$, where k is the cache size. (at most k pages in cache), and total (k+1) distinct pages.
  - [pf.]

$c_A(\sigma)$	$c_{OPT}(\sigma)$
Worst case = $\lvert\sigma\rvert$	

All requests are faults.

Evict p_i at t, then request p_i at (t+1).

Paging problem



- [pf.]

$c_A(\sigma)$	$c_{OPT}(\sigma)$
Worst case = $\lvert\sigma\rvert$	

- Suppose that OPT has a fault on some request σ(t).
- OPT can evict a page is not requested during the next k-1 requests σ(t + 1), …, σ(t + k -1).
- Thus, on any k consecutive requests, OPT has $\leq$ 1 fault

Paging problem

- Claim: If A is a deterministic online algorithm that is $\alpha - competitive$, then $\alpha \geq k$, where k is the cache size. (at most k pages in cache), and total (k+1) distinct pages.
  - [pf.]

$c_A(\sigma)$	$c_{OPT}(\sigma)$
Worst case = $\|\sigma\|$	$\leq \frac{\|\sigma\|}{k}$

Every k pages
has $\leq$ 1 fault

Deterministic v.s. Randomized

- Case 3. Random Marking Algorithm (RMA)

- Initialize all pages as marked
- Upon request of a page p
  - If p is not in cache,
    - If all pages in cache are marked, unmark all
    - Evict a random unmarked page
  - Mark page p

Deterministic v.s. Randomized

- Case 3. Random Marking Algorithm (RMA)

Example Suppose $k = 3$, $\sigma = (2, 5, 2, 1, 3)$.

Suppose the cache is initially:

Cache	1	3	4
Marked?	✓	✓	✓

When $\sigma(1) = 2$ arrives, all pages were unmarked. Suppose the random eviction chose page '3'. The newly added page '2' is then marked.

Cache	1	2	4
Marked?	✗	✓	✗

When $\sigma(2) = 5$ arrives, suppose random eviction chose page '4' (between pages '1' and '4'). The newly added page '5' is then marked.

Cache	1	2	5
Marked?	✗	✓	✓

When $\sigma(3) = 2$ arrives, page '2' in the cache is marked (no change).

Cache	1	2	5
Marked?	✗	✓	✓

When $\sigma(4) = 1$ arrives, page '1' in the cache is marked. At this point, any page request that is not from $\{1, 2, 5\}$ will cause a full unmarking of all pages in the cache.

Cache	1	2	5
Marked?	✓	✓	✓

Deterministic v.s. Randomized

- Competitiveness analysis
- A: c-competitive

$$E[C_A(\sigma)] \leq c \cdot C_{OPT}(\sigma)$$

Summary

- Recap. Streaming algorithm v.s. Online algorithm
- Online v.s. Offline
  - Example. sorting
- Online performance – competitiveness analysis
  - Case 1. Ski Rental Problem
  - Case 2. Deterministic Paging Problem
  - Case 3. Random Marking Algorithms
- Types of adversary

	Online (random inputs)	Online (adversarial inputs)	Offline
Compare to Offline OPT	Competitive ratio $\leq$	Competitive ratio	Approximation ratio
Cost(A) related to… **1. Inputs**	a. Unknown but expected b. Random w/o known patterns ➔ Online alg. Cost(A): ➔ fluctuate	a. Known (simulated) b. Not random ➔ Online alg. Cost(A)’: ➔ Stable ➔ Online alg. Cost(A) ≤ Cost(A)’	a. Known b. Not random –OR- Random w/. Distribution ➔ Offline alg. Cost(A): ➔ stable
Cost(A) related to… **2. Algorithm**	At different states… Same strategy - Deterministic Different strategies – Random	At different states… Same strategy - Deterministic Different strategies – Random	Single strategy
Inputs to the Alg.	Hard –OR- easy ➔ average case	Hard ➔ worst case	Hard ➔ worst case