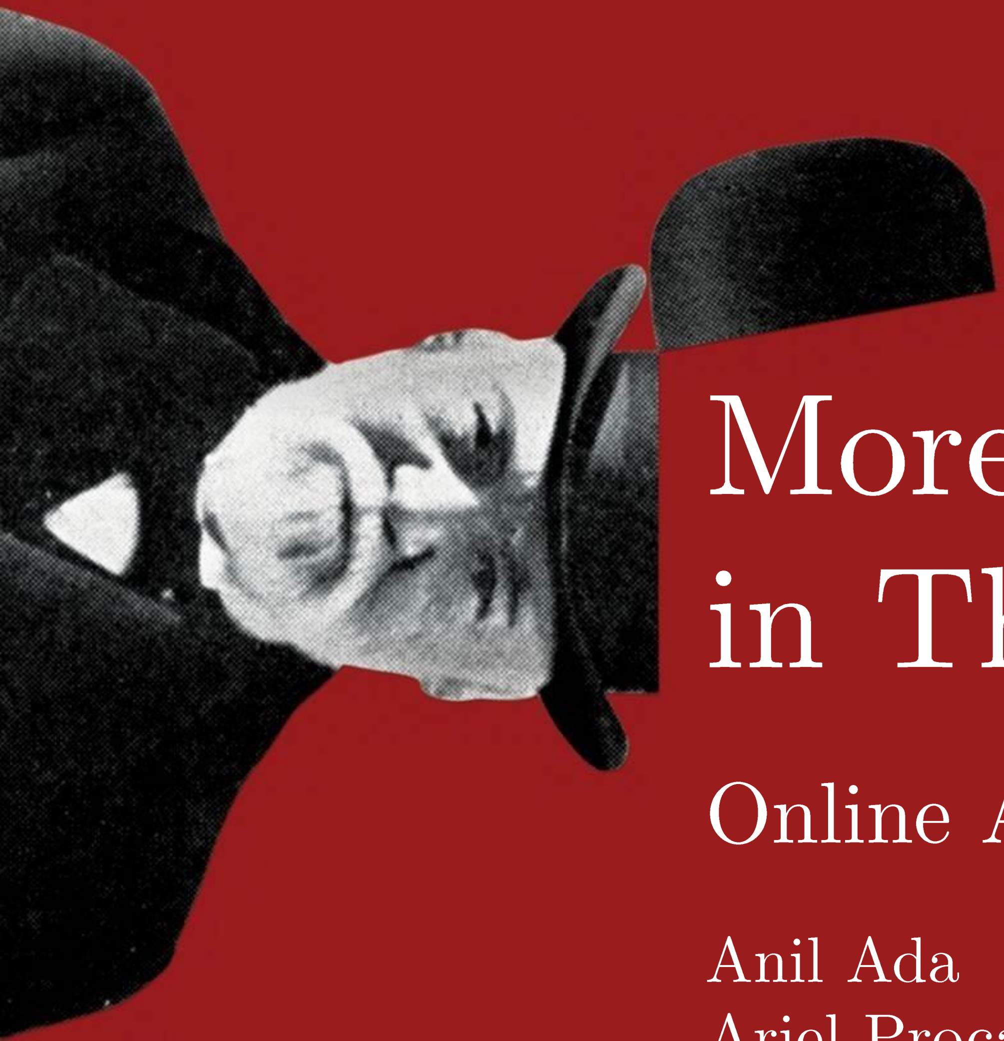


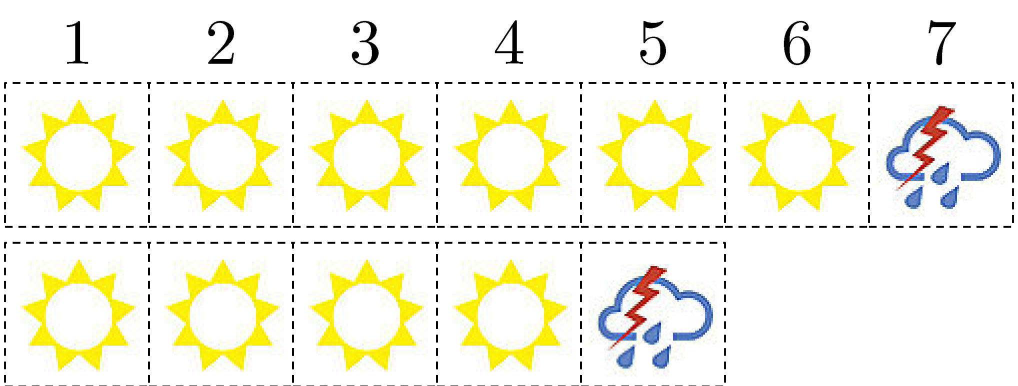
More Great Ideas in Theoretical CS

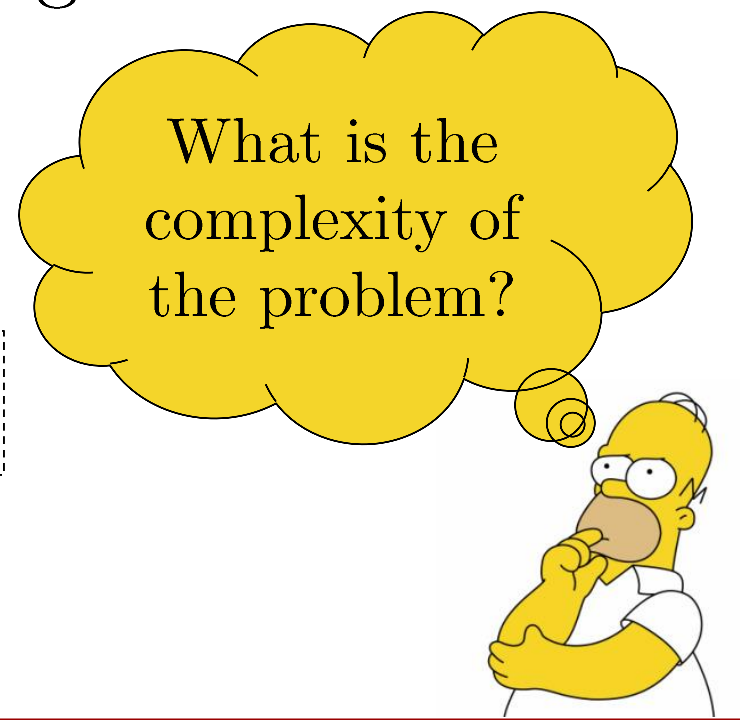
Online Algorithms

Anil Ada
Ariel Procaccia (this time)

Ski rental

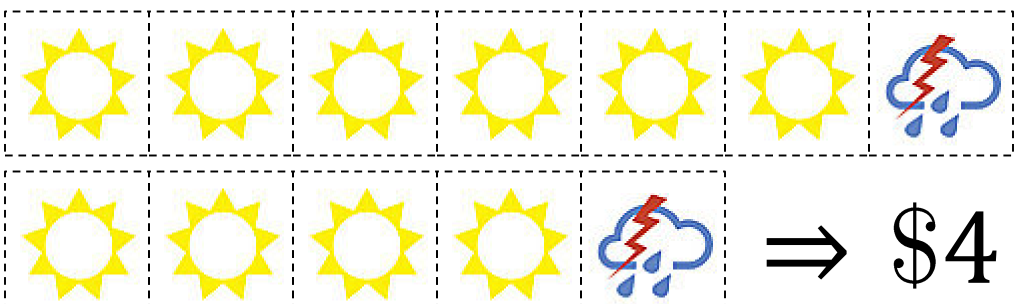
- You are on a ski vacation; you can buy skis for $\$B$ or rent for $\$1$/day
- You're very spoiled: You'll go home when it's not sunny
- Rent or buy when $B = 5$?





Ski rental

- Now assume you don't know in advance how many days of sunshine there are
- Every day of sunshine you need to decide whether to rent or buy
- Algorithm: Rent for B days, then buy



$\Rightarrow$ \$10

$\Rightarrow$ \$4

Ski rental

Assume $B \geq 8$. How bad can the "rent B days, then buy" algorithm be compared to the optimal solution in the worst case?

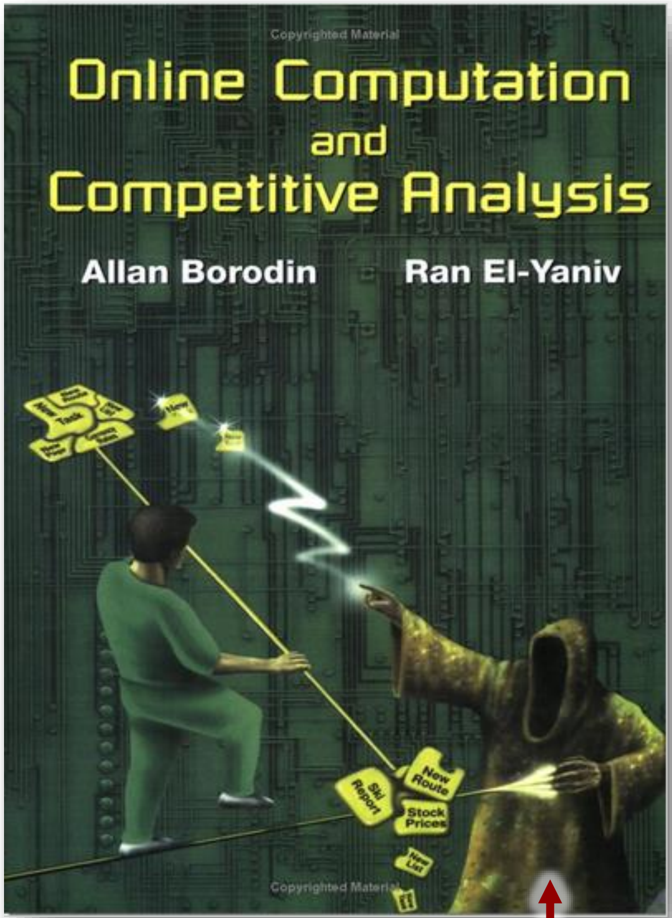
1. $ALG(I) = 2 \cdot OPT(I)$
2. $ALG(I) = 3 \cdot OPT(I)$
3. $ALG(I) = \frac{B}{2} \cdot OPT(I)$
4. $ALG(I) = B \cdot OPT(I)$

Competitive ratio

- For a minimization problem and $c > 1$, ALG is a c-competitive algorithm if for every instance I, $ALG(I) \leq c \cdot OPT(I)$
- For a maximization problem and $c < 1$, ALG is a c-competitive algorithm if for every instance I, $ALG(I) \geq c \cdot OPT(I)$
- The difference from approximation algorithms is that here ALG is online, whereas $OPT(I)$ is the optimal offline solution

SKI RENTAL, REVISITED

- Our ski-rental algorithm is 2-competitive
- Renting for $B - 1$ days is $\left(\frac{2B-1}{B}\right)$-competitive
- We prove that no online algorithm can do better by constructing an evil adversary



Ski rental, revisited

- Theorem: No online algorithm for the ski rental problem is α-competitive for $\alpha < \frac{2B-1}{B}$
- Proof:
  - Alg is defined by renting for K days and buying on day $K+1$
  - Evil adversary makes it rain on day $K+2$
  - $K \geq B$: $OPT(I) = B, ALG(I) = K + B \geq 2B$
  - $K \leq B - 2$: $OPT(I) = K + 1$,
    $ALG(I) = K + B \geq 2K + 2$ ■

Ski rental, revisited

Proving lower bounds for online algorithms is much easier than for approximation algorithms!

Paging

- Hard drive holds N pages, memory holds k pages
- When a page of the hard drive is needed, it is brought into the memory
- If it's already in the memory, we have a hit, otherwise we have a miss
- If the memory is full, we may need to evict a page
- Paging algorithm tries to minimize misses

PAGING

Memory			Request sequence
1	2	3	4
4	2	3	4 1
4	1	3	4 1 3
4	1	3	4 1 3 2
2	1	3	4 1 3 2 4

Paging

Memory			Request sequence
1	2	3	4
1	4	3	4 1
1	4	3	4 1 3
1	4	3	4 1 3 2
2	4	3	4 1 3 2 4

PAGING

- Three online paging algorithms (start with $1, \dots, k$) in memory
- LRU (least recently used)
- FIFO (first in first out): memory works like a queue; evict the page at the head and enqueue the new page
- LIFO (last in first out): memory works like a stack; evict top, push new page

Example: LIFO

Memory			Request sequence
1	2	3	4
1	2	4	4 3
1	2	3	4 3 4
1	2	4	4 3 4 3
1	2	3	4 3 4 3 4

Paging

- Theorem: LRU is k-competitive
- Proof:
  - We divide the request sequence into phases; phase 1 starts at the first page request; each phase is the longest possible with at most k requests for distinct pages
  - Example with $k = 3$:

Phase 1	Phase 2	Phase 3
4 1 2 1	5 3 4 5	1 2 3

Paging

- Theorem: LRU is k-competitive
- Proof (continued):
  - Denote m =#phases, and by p_j^i the jth distinct page in phase i
  - Pages $p_1^i, \ldots, p_k^i, p_1^{i+1}$ are all distinct
  - If OPT hasn't missed on pages $p_2^i, \ldots, p_k^i$, it will miss on p_1^{i+1}, i.e., it misses at least once for every new phase (including phase 1) $\Rightarrow$ $OPT \geq m$

Paging

- Theorem: LRU is k-competitive
- Proof (continued):
  - LRU misses at most once on each distinct page in a phase
  - Therefore, $ALG \leq km$ ■

4	1	2	5
5	1	2	5 3
5	1	3	5 3 4
5	4	3	5 3 4 5

Phase 2 of the example on slide 17

Paging

- Theorem: FIFO is k-competitive
- Proof: Essentially the same ■
- Theorem: No online alg for the paging problem is α-competitive for $\alpha < k$

Paging

- Proof:
  - At each step the evil adversary requests the missing page in $\{1, \ldots, k+1\} \Rightarrow$ miss every time

1	2	3	4
4	2	3	4 1
4	2	1	4 1 3
4	3	1	4 1 3 2
4	3	2	4 1 3 2 1

Paging

- Proof:
  - If OPT evicts a page, it will take at least k requests to miss again ■

1	2	3	4				
1	4	3	4	1			
1	4	3	4	1	3		
1	4	3	4	1	3	2	
1	4	2	4	1	3	2	1