COS330: Great Ideas in Theoretical Computer Science — Fall 2025

New Computational Models ▸ Online Algorithms

Lecture 14

- Motivation and Competitive Ratio
- Ski Rental Problem
- Online Bipartite Matching

PRINCETON COMPUTER SCIENCE

New Computational Models ▸
Online Algorithms

Lecture 14

- Motivation and Competitive Ratio
- Ski Rental Problem
- Online Bipartite Matching

Finding the Maximum

Problem: Find the maximum element in a list

Offline version: You see the entire list at once

- Scan through, keep track of maximum seen so far
- Return the maximum at the end
- Easy!

Online version: Elements arrive one at a time

- You must immediately and irrevocably decide: select this element or not?
- You can only select one element total
- Goal: Select the maximum element

Remark

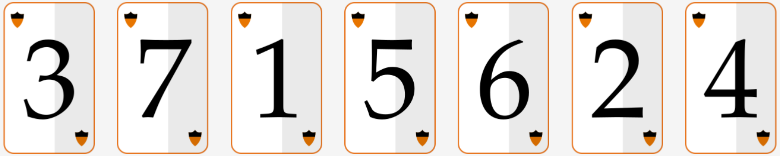
Challenge: How do you decide without knowing what comes next?

Example: Elements Arriving Online

Problem:

- Elements arrive one at a time (you can't see future elements)
- After seeing each element, you must decide: select it or skip it?
- Once you select, you're done (you can only select one element)
- Goal: Select the maximum element in the list

Elements arrive:



Key question: How do we measure the quality of our solution when we can't always be optimal?

Measuring Online Algorithm Performance

Idea: Compare online algorithm to optimal offline algorithm (which knows the future)

$$\text{Competitive Ratio} = \frac{\text{Value of online algorithm}}{\text{Value of optimal offline algorithm}} = \frac{\text{Value of online algorithm}}{\text{Optimal value in hindsight}}$$

Goal: Design algorithms with competitive ratio close to 1

Remark

Notes:

- For minimization problems: competitive ratio ≥ 1 (online pays at least as much as offline)
- For maximization problems: competitive ratio ≤ 1 (online gets at most as much as offline)
- This is worst-case analysis over all possible inputs

Competitive Ratio: Formal Definition

Consider an online algorithm A for a minimization problem:

Let $A(I)$ denote the cost of algorithm A on input I
Let $\mathsf{OPT}(I)$ denote the cost of the optimal offline algorithm on input I

Algorithm A is α-competitive if for all inputs I:

$$A(I) \leq \alpha \cdot \mathsf{OPT}(I)$$

New Computational Models ▸ Online Algorithms

Lecture 14

- Motivation and Competitive Ratio
- Ski Rental Problem
- Online Bipartite Matching

The Ski Rental Problem

Setup: You're going on a ski trip

- You don't know how many days you'll ski
- Each day, you must decide: rent or buy skis?

Costs:

- Renting costs R per day
- Buying costs B (one-time purchase)
- Assume B/R is an integer (for simplicity)

Online constraint: You learn how many days you'll ski only as they pass

- Each morning: do you rent or buy today?
- Once you buy, you own the skis (no more rental costs)
- Once you stop skiing, the season ends

Goal: Minimize total cost compared to optimal offline algorithm (which knows the number of days in advance)

Competitive Ratio for Ski Rental

Let d be the total number of days you ski

Optimal offline cost:

$$\text{OPT}(d) = \begin{cases} Rd & \text{if } Rd < B \quad \text{(rent all days)} \\ B & \text{if } Rd \geq B \quad \text{(buy on day 1)} \end{cases}$$

Equivalently: $\text{OPT}(d) = \min(Rd, B)$

For an online algorithm A:

- Let $A(d)$ be the cost when skiing for d days

Algorithm A is α-competitive if:

$$\max_{d \geq 1} \frac{A(d)}{\text{OPT}(d)} \leq \alpha$$

Goal: Find an online algorithm with the smallest possible competitive ratio

Algorithm 1: Buy Immediately

Algorithm: Buy-First-Day

On day 1: Buy skis for B

Analysis:

- Cost: $A(d) = B$ for all d
- If $Rd < B$: $\text{OPT}(d) = Rd$, so ratio $= \frac{B}{Rd} \leq \frac{B}{R}$
- If $Rd \geq B$: $\text{OPT}(d) = B$, so ratio $= \frac{B}{B} = 1$

Competitive ratio: $\frac{B}{R}$

Remark

This is bad when B/R is large! If you only ski 1 day but buy expensive skis, you waste money compared to renting.

Algorithm 2: Rent Forever

Algorithm: Rent-Forever

Every day: Rent skis for R
Never buy.

Analysis:

- Cost: $A(d) = Rd$
- If $Rd < B$: $\mathrm{OPT}(d) = Rd$, so ratio $= \frac{Rd}{Rd} = 1$
- If $Rd \geq B$: $\mathrm{OPT}(d) = B$, so ratio $= \frac{Rd}{B}$
  - As $d \to \infty$, ratio $\to \infty$

Competitive ratio: Unbounded! (∞)

Remark

This is terrible for long trips! If you ski for many days, you keep paying R per day when you should have bought for B.

Algorithm 3: Buy After B/R Days

Algorithm: Better-Late-Than-Never

For days $1, 2, \ldots, \frac{B}{R} - 1$: Rent skis for R per day
On day $\frac{B}{R}$ (if you're still skiing): Buy skis for B

Intuition:

- Don't commit early (unlike Algorithm 1)
- Don't rent forever (unlike Algorithm 2)
- Switch to buying at the "breakeven" point

Cost:

$$A(d) = \begin{cases} Rd & \text{if } d < \frac{B}{R} \quad \text{(only rented)} \\ R\left(\frac{B}{R} - 1\right) + B = 2B - R & \text{if } d \geq \frac{B}{R} \quad \text{(rented } \frac{B}{R} - 1 \text{ days, then bought)} \end{cases}$$

Analysis of Better-Late-Than-Never

Recall: $A(d) = Rd$ if $d < \frac{B}{R}$ $A(d) = 2B - R$ if $d \geq \frac{B}{R}$ $\text{OPT}(d) = \min(Rd, B)$

Lemma:

Better-Late-Than-Never is $(2 - \frac{R}{B})$-competitive.

Proof

Case 1: $d < \frac{B}{R}$

- $A(d) = Rd = \text{OPT}(d)$, so ratio $= 1$

Case 2: $d \geq \frac{B}{R}$

- $\text{OPT}(d) = B$
- $A(d) = 2B - R$
- Ratio $= \frac{2B-R}{B} = 2 - \frac{R}{B}$

Taking the maximum over all cases: competitive ratio $= 2 - \frac{R}{B}$ $\square$

Optimality of Better-Late-Than-Never

Question: Can we do better than $2 - \frac{R}{B}$?

Answer: No! (For deterministic algorithms)

Exercise: Prove that any deterministic online algorithm for ski rental has competitive ratio at least $2 - \frac{R}{B}$.

Hint. Let k be the first day the algorithm buys (set $k = \infty$ if it never buys). Consider two inputs:

- stop exactly on day k;
- force the trip to last beyond day k.

Compare ALG and OPT in each case and take the worse. Which k minimizes this worst case?

Remark

Note: Randomized algorithms can achieve better competitive ratios! The ski rental problem with randomization achieves ratio $e/(e-1) \approx 1.58$.

New Computational Models ▸
Online Algorithms
Lecture 14

- Motivation and Competitive Ratio
- Ski Rental Problem
- Online Bipartite Matching

Online Bipartite Matching

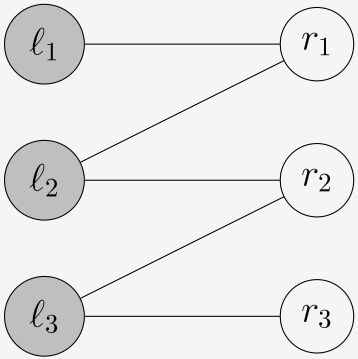
Setup: Bipartite graph $G = (L \cup R, E)$

- Left vertices L are known in advance
- Right vertices R arrive online, one at a time
- When vertex $v \in R$ arrives, we see all its edges to L
- Must immediately match v to an available neighbor in L (or leave unmatched)
- Once matched, a vertex in L cannot be matched again

Goal: Maximize number of matches

Offline optimum: Maximum matching in the graph (known optimal from network flow)
Online challenge: Make matching decisions without knowing future arrivals

Example: Online Matching



Online algorithm's view:

- r_1 arrives: connected to $\{\ell_1, \ell_2\}$. Match to one of them?
- r_2 arrives: connected to $\{\ell_2, \ell_3\}$. Match to one of them?
- r_3 arrives: connected to $\{\ell_3\}$. Match to ℓ_3?

Bad online choice: Match $r_1 \to \ell_2$, match $r_2 \to \ell_3$, can't match r_3 $\to$ 2 matches
Optimal (offline): Match $r_1 \to \ell_1$, match $r_2 \to \ell_2$, match $r_3 \to \ell_3$ $\to$ 3 matches (perfect!)

Upper Bound for Deterministic Algorithms

Lemma:

No deterministic online algorithm can achieve competitive ratio better than $\frac{1}{2}$ for online bipartite matching.

Proof

Setup: $L = \{\ell_1, \ell_2\}$, R arrives online

Instance 1:

- r_1 arrives, connected to both ℓ_1 and ℓ_2
- r_2 arrives, connected only to ℓ_1

To achieve ratio $> \frac{1}{2}$ on Instance 1: Need at least 2 matches (since OPT = 2)
This requires: don't match r_1 to ℓ_1 (so ℓ_1 is available for r_2)

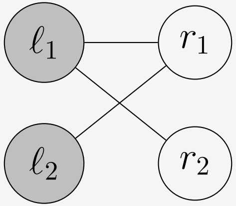
Instance 2:

- r_1 arrives, connected to both ℓ_1 and ℓ_2
- r_2 arrives, connected only to ℓ_2

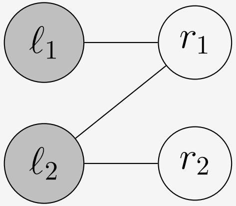
To achieve ratio $> \frac{1}{2}$ on Instance 2: Need at least 2 matches (since OPT = 2)
This requires: don't match r_1 to ℓ_2 (so ℓ_2 is available for r_2)

Upper Bound Proof (Continued)

Instance 1:



Instance 2:



Proof

Key observation: In both instances, when r_1 arrives, the algorithm sees the *exact same* information: r_1 is connected to both ℓ_1 and ℓ_2.

Since the algorithm is deterministic: It must make the same decision in both instances when r_1 arrives.

But we showed:

- Instance 1 requires: r_1 not matched to ℓ_1
- Instance 2 requires: r_1 not matched to ℓ_2

Therefore: Any deterministic algorithm must fail to achieve ratio $> \frac{1}{2}$ on at least one of the instances. $\square$

Alternative Proof: Adversarial Argument

The previous proof can be reframed using an adversary:

Key idea: An adversary *constructs* a worst-case input instance *on the fly* by observing the algorithm's decisions. The adversary reveals vertices one at a time, choosing what to reveal next based on what the algorithm has done so far.

Proof

Setup: $L = \{\ell_1, \ell_2\}$ (known to algorithm)
The adversary will build the instance by deciding which r vertices to send.

Adversarial strategy:

1. **Step 1:** Adversary reveals r_1 connected to both ℓ_1 and ℓ_2
2. **Step 2:** Adversary observes the algorithm's decision and reacts:
    - **If algorithm matches r_1 to some $\ell_i \in \{\ell_1, \ell_2\}$:**
        - Adversary completes the instance by revealing r_2 connected *only* to ℓ_i
        - Now algorithm cannot match r_2 (since ℓ_i is already taken)
        - Result: Algorithm gets 1 match, OPT gets 2 $\rightarrow$ ratio $= \frac{1}{2}$
    - **If algorithm doesn't match r_1:**
        - Adversary completes the instance by revealing no more vertices
        - Result: Algorithm gets 0 matches, OPT gets 1 $\rightarrow$ ratio $= 0 < \frac{1}{2}$

Understanding the Adversarial Proof

Key insight: The adversary is just choosing which of the two instances (from the previous proof) to use based on the algorithm's behavior.

Connection to instance-based proof:

- If algorithm matches $r_1 \to \ell_1$: adversary picks Instance 1 (where r_2 needs ℓ_1)
- If algorithm matches $r_1 \to \ell_2$: adversary picks Instance 2 (where r_2 needs ℓ_2)

Remark

Important: Both proofs establish the same impossibility result. The adversarial framing is often more intuitive and easier to generalize to other problems.

Takeaway: No matter what deterministic strategy the algorithm uses, the adversary can construct an input that forces competitive ratio $\leq \frac{1}{2}$. $\square$

Greedy Algorithm

Can we achieve the $\frac{1}{2}$ upper bound?

Algorithm: Greedy Online Matching

When vertex $v \in R$ arrives:

- If v has at least one unmatched neighbor in L:
  - Match v to an arbitrary unmatched neighbor
- Else: leave v unmatched

Claim: Greedy is $\frac{1}{2}$-competitive

Proof strategy: We'll use LP duality! (note the algorithm doesn't use duality, it's just for analysis)

- Relate matching to LP relaxation
- Use dual (vertex cover) to bound optimal matching
- Show greedy achieves at least half of optimal

LP Formulation and Duality

Recall from Lecture 12:

Matching LP (Primal):

Variables: $x_e \geq 0$ for each edge $e \in E$

Maximize: $\sum_{e \in E} x_e$

Subject to: For each vertex $v \in L \cup R$:

$$\sum_{\substack{e \text{ incident} \\ \text{to } v}} x_e \leq 1$$

Vertex Cover LP (Dual):

Variables: $y_v \geq 0$ for each vertex $v \in L \cup R$

Minimize: $\sum_{v \in L \cup R} y_v$

Subject to: For each edge $e = (u, v) \in E$:

$$y_u + y_v \geq 1$$

Remark

$$\text{Max Matching} \leq \text{Max Matching LP} = \text{Min VC LP} \leq \text{Min VC}$$

Proof strategy: Use greedy algorithm's decisions to construct a feasible vertex cover LP solution. This provides an upper bound on the optimal offline matching value.

Analysis of Greedy Algorithm

Strategy: Define a "pre-dual solution" z based on greedy's decisions, then scale it up to get a feasible dual solution y.

Let M be the matching output by greedy.

Define pre-dual solution z: For every $v \in L \cup R$:

$$z_v = \begin{cases} \frac{1}{2}, & \text{if greedy matches } v \\ 0, & \text{otherwise} \end{cases}$$

Key observation:

$$|M| = \sum_{v \in L \cup R} z_v$$

Why? Each matched edge (v, w) contributes $z_v + z_w = \frac{1}{2} + \frac{1}{2} = 1$ to the sum, and there are $|M|$ such edges.

Pre-Dual Solution Properties

Claim: For every edge $(v, w) \in E$, at least one of z_v, z_w equals $\frac{1}{2}$.

Proof

Case 1: Edge (v, w) is in matching M

- Then both $z_v = \frac{1}{2}$ and $z_w = \frac{1}{2}$

Case 2: Edge (v, w) is not in matching M

- Consider the time $w \in R$ arrived online
- If $z_w = 0$, then greedy didn't match w
- This means w had no unmatched neighbors when it arrived
- In particular, v was already matched when w arrived
- Therefore $z_v = \frac{1}{2}$

Scaling to a Feasible Dual Solution

The claim implies: For every edge $(v, w) \in E$: $z_v + z_w \geq \frac{1}{2}$

This means z is NOT quite feasible for the dual (which requires $z_v + z_w \geq 1$)

Solution: Scale up by a factor of 2! Define $y = 2z$:

$$y_v = 2z_v = \begin{cases} 1, & \text{if greedy matches } v \\ 0, & \text{otherwise} \end{cases}$$

Now y is feasible for the dual:

- For every edge (v, w): $y_v + y_w = 2(z_v + z_w) \geq 2 \cdot \frac{1}{2} = 1$
- All $y_v \geq 0$

Dual objective value:

$$\sum_{v \in L \cup R} y_v = 2 \sum_{v \in L \cup R} z_v = 2|M|$$

Completing the Analysis

What we've shown:

- Greedy matching M has size $|M|$
- Constructed feasible dual solution y with value $2|M|$

By weak duality:

$$\text{OPT}_{\text{matching}} \leq \text{Min dual value} \leq \text{Value of } y = 2|M|$$

Therefore:

$$|M| \geq \frac{1}{2} \cdot \text{OPT}_{\text{matching}}$$

Lemma:

Greedy online matching is $\frac{1}{2}$-competitive for online bipartite matching.

Conclusion: Greedy achieves the best possible competitive ratio for deterministic online algorithms!