COS 330: Great Ideas in Theoretical Computer Science

Fall 2026

Precept 9

Hedge Your Bets

PROBLEM-SOLVING SESSION

> Today is a **problem-solving session** on online algorithms. There is a single problem, in which you will use randomization to beat the best deterministic algorithm for the ski rental problem from lecture. Work on it on your own or in a small group, and ask your TA when you get stuck. Solutions will be shared after precept.

Problem 1: Ski Rental with Randomization

Recall the *ski rental problem* from lecture. You are planning a ski trip, but don't know in advance how many days d you will ski. Each morning, you must decide whether to rent skis or buy skis:

- **Renting** costs $\$1$ per day. (In lecture renting cost $\$R$ per day; here we set $R = 1$.)
- **Buying** costs $\$B$, a one-time purchase, where B is an integer.

Once you buy skis, you own them and pay nothing more. You only learn that the trip is over when it ends.

Let $d \geq 1$ denote the (unknown) total number of days you ski. An optimal offline algorithm knows d in advance, and achieves cost

$$\text{OPT}(d) = \min(d, B).$$

For an online algorithm A, let $A(d)$ denote its cost when the trip lasts d days. Recall from lecture that a deterministic online algorithm A is α-*competitive* if

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

(a) Recall the **Better-Late-Than-Never** deterministic algorithm from lecture:

- Rent on days $1, 2, \ldots, B-1$.
- On day B (if you are still skiing), buy for $\$B$.

Compute the competitive ratio of Better-Late-Than-Never.
(We did this in class, so if you remember the calculation well, feel free to skip writing it out again!)

We also saw in lecture that no deterministic online algorithm can achieve a competitive ratio better than $2 - 1/B$, so Better-Late-Than-Never is the best we can do deterministically. Let's turn to randomization to beat this bound.

For a randomized online algorithm A, its cost $A(d)$ is a random variable. We say that A is α-*competitive in expectation* if

$$\max_{d \geq 1} \frac{\mathbb{E}[A(d)]}{\text{OPT}(d)} \leq \alpha,$$

where the expectation is over the algorithm's random choices.

(b) Assume that B is an even integer. Consider the following randomized algorithm H_p (the "hedging" algorithm), which depends on a probability $p \in [0, 1]$:

- Rent on days $1, 2, \ldots, B/2 - 1$.
- On day $B/2$ (if you are still skiing), buy with probability p, and otherwise rent.
- On day B (if you are still skiing and haven't bought yet), buy for $\$B$.

Compute $\mathbb{E}[H_p(d)]$ for every $d \geq 1$.

Hint: Your answer should be a piecewise function, depending on whether $d < B/2$, $B/2 \leq d < B$, or $d \geq B$.

(c) Find a value of $p \in [0, 1]$ for which H_p is α-competitive in expectation for some α strictly less than $2 - 1/B$ (so that it beats every deterministic algorithm). Prove that your choice of p achieves this.

Hint: The competitive ratio is $\max_d \mathbb{E}[H_p(d)]/\text{OPT}(d)$. Analyze each case from part (b) separately, and then choose p to balance the worst-case ratios.

You can take the above approach and generalize it to get even better competitive ratios, by adding more randomized buying points between days 1 and B. If you do so and optimize the probabilities, you can get arbitrarily close to the optimal competitive ratio of $e/(e-1) \approx 1.582$ that we mentioned in lecture.

However, the analysis becomes more complex as you add more randomized steps. If you're curious to see the full analysis, here is a good reference. And in case you are still not convinced that linear programming is really useful, the analysis of the optimal randomized ski rental algorithm can be formulated as a linear program!