COS 330: Great Ideas in Theoretical Computer Science

Fall 2025

Precept 8

My stream of consciousness

Practice

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 (note that in lecture we used $\$R$, but here we set $R = 1$)
- **Buying** costs $\$B$ (one-time purchase)

Once you buy skis, you own them and incur no further costs. Once you stop skiing, the trip ends and the problem terminates.

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 for days $1, 2, \ldots, B - 1$
- On day B (if you're 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. So let's turn to randomization to find an algorithm that beats this bound.

For a randomized online algorithm A, we say 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 B is an even integer. Consider the following randomized algorithm H_p (the "hedging" algorithm) parameterized by some probability $p \in [0, 1]$:

- Rent for days $1, 2, \ldots, B/2 - 1$
- On day $B/2$ (if you're still skiing): Buy with probability p, otherwise continue renting
- On day B (if you're still skiing and haven't bought yet): Buy for \$B

Compute $\mathbb{E}[H_p(d)]$ for all values of d.

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

(c) Find a value of $p \in [0, 1]$ such that algorithm H_p achieves a competitive ratio strictly less than 2. Prove that your choice of p achieves this ratio.

Hint: The competitive ratio is $\max_d \frac{\mathbb{E}[H_p(d)]}{\text{OPT}(d)}$. Analyze each case from part (b) separately, 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 $e/(e-1) \approx 1.582$.

However, the analysis becomes more complex as you add more randomization steps. If you're curious to see the full analysis, here is a good reference. 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!

Problem 2: Uniform Sampling from a Stream

Consider a stream of n elements $a_1, a_2, \ldots, a_n$ arriving one at a time. You do not know n in advance. Your goal is to output a *uniformly random* element from the stream. That is, after seeing all n elements, you must output an element such that each a_i has probability exactly $1/n$ of being selected.

Constraints:

- You see elements one at a time: $a_1, a_2, \ldots, a_n$
- You don't know n (the total number of elements) until you've seen them all
- You can only make one pass through the stream
- You must use $O(1)$ memory (i.e., constant space, independent of n)

(a) Consider the following algorithm that satisfies the above constraints:

- Initialize $s \leftarrow a_1$ (store the first element)
- For $i = 2, 3, \ldots$ (while elements arrive):
  - When element a_i arrives, replace s with a_i with probability $1/i$
- Output s

Prove that after seeing the first k elements, each element a_j (for $j \leq k$) has probability exactly $1/k$ of being stored in s.