IOE 691: Approximation & Online Algorithms

Lecture Notes: Introduction to Online Optimization

Instructor: Viswanath Nagarajan

1 Rent or Buy

Consider the following "rent or buy" problem, a.k.a. *ski rental*. A person needs a house in some town, that can be either rented or bought. They need to decide each month whether to continue renting (at a cost of \$1 per month) or to buy (at a one-time cost of $\$B$). However, the person does not know upfront, the number T of months they will eventually live in the town. So they need to decide in an *online* manner for each month, whether to continue renting or to buy. The goal is to minimize the total cost of renting plus buying. If the person rents for k months and buys on month $k+1$ then the cost incurred is $k+B$. If the actual number of months T is known, it is clear that the optimal cost is $\min(T, B)$. We are interested in an online algorithm that guarantees a cost close to this optimum.

We measure the performance of an online algorithm by means of the *competitive ratio*, which is the worst-case ratio (over all instances) of the cost of the algorithm and the optimum.

$$\text{Competitive ratio of algorithm } \mathcal{A} \;=\; \max_{\text{instance } I} \frac{cost_{\mathcal{A}}(I)}{OPT(I)}.$$

Note that the online algorithm does not have full information on the instance I whereas the optimal solution knows the full instance. So this is an unfair comparison for the online algorithm (which is handicapped due to less information). Nevertheless, in many interesting and practical settings, we can obtain online algorithms with strong guarantees.

Back to the ski rental problem, an instance is given by the buy cost B and the number of months T. An online algorithm $\mathcal{A}$ knows B but does not know T. Let I_j be the instance where there are $T = j$ days. Suppose we knew ahead of time that I_j is the eventual input instance, then we have that $\mathsf{OPT}(I_j) = \min\{j, B\}$ since we can choose to either buy on month 1 or rent every month (depending on which is better).

We can also classify and analyze all possible algorithms. Note that an algorithm for this problem is simply a rule deciding when to buy. For any integer $i \geq 0$ let ALG_i be the algorithm that rents for i days, and then buys on month $i+1$ (if there are so many days). The cost of ALG_i on instance I_j is then

$$\mathsf{ALG}_i(I_j) = (i+B) \cdot \mathbf{1}_{\{i+1 \leq j\}} + j \cdot \mathbf{1}_{\{i+1 > j\}}.$$

What is the best algorithm from the point of view of competitive analysis?

A natural approach is to balance the costs of renting and buying, which leads to the online algorithm ALG_{B-1}. Rent for up to $B-1$ months, and buy on month B (if needed).

Theorem 1.1 *The competitive ratio of algorithm ALG_{B-1} is $2 - \frac{1}{B}$ and this is the best possible ratio for any deterministic algorithm.*

Proof: There are two cases to consider $j < B+1$ and $j \geq B+1$. For the first case, $\mathsf{ALG}_{B-1}(I_j) = j$ and $\mathsf{OPT}(I_j) = j$, so $\frac{\mathsf{ALG}_{B-1}(I_j)}{\mathsf{OPT}(I_j)} = 1$. In the second case, $\mathsf{ALG}_{B-1}(I_j) = 2B-1$ and $\mathsf{OPT}(I_j) = B$, so $\mathsf{ALG}_{B-1}(I_j)/\mathsf{OPT}(I_j) = 2 - 1/B$. Thus the competitive ratio of ALG_{B-1} is

$$\max_{I_j} \frac{\mathsf{ALG}_{B-1}(I_j)}{\mathsf{OPT}(I_j)} = 2 - \frac{1}{B}$$

Now to show that this is the best possible competitive ratio for any deterministic algorithm. Consider any algorithm ALG_i. We claim that instance I_j with $j := i+1$ has $\mathsf{ALG}_i(I_j)/\mathsf{OPT}(I_j) \geq 2-1/B$. Note that $\mathsf{ALG}_i(I_j) = (i+B)$. Consider the following cases.

- If $i \geq B-1$ then $\mathsf{OPT}(I_j) = B$. Hence,

$$\frac{\mathsf{ALG}_i(I_j)}{\mathsf{OPT}(I_j)} = \frac{i+B}{B} = \frac{i+1}{B} + 1 - \frac{1}{B} \geq 2 - \frac{1}{B}.$$

- If $0 \leq i \leq B-2$ then $\mathsf{OPT}(I_j) = i+1$, and

$$\frac{\mathsf{ALG}_i(I_j)}{\mathsf{OPT}(I_j)} = \frac{i+B}{i+1} = 1 + \frac{B-1}{i+1} \geq 2.$$

■

Here we specify that the algorithm is "deterministic" as we will see later that by allowing *randomization*, we can get a better *expected* performance.

2 Online Line Search

Consider the following search scenario. A target is located at an *unknown distance* $T \geq 1$ along a line. We can repeatedly send a "probe" to travel any pre-determined distance b along the line. This means that the probe searches until distance b and then returns. If the target is located within distance b (i.e., $T \leq b$) then the probe finds the target; otherwise (i.e., $T > b$) the probe learns that the target is further away than b. The goal is to find the target using the minimum total distance (traveled by the probe). To keep things simple, we will measure distances only in one direction (although the probe has to return to its origin). By scaling, this is equivalent to measuring the round-trip distance. This problem is also known as *online bidding*. Note that the input here is simply given by the value T of the target distance. Moreover, the optimal value of such an instance is just T.

Any online algorithm for the line search problem can be specified by a pre-determined sequence $\langle b_1, b_2, \cdots \rangle$ of distances traveled by the probe in each step. This algorithm terminates at the first step i where $b_i \geq T$ which is the point when the unknown target is found. Clearly, we can assume that these distances are increasing, i.e., $b_1 < b_2 < \cdots b_i < b_{i+1} < \cdots$. For any value $t \geq 0$, let $I(t)$ denote the minimum index i with $b_i \geq t$. Then, the cost of the algorithm on instance T is $\sum_{i=1}^{I(T)} b_i$. So the competitive ratio is:

$$\max_{T} \frac{\sum_{i=1}^{I(T)} b_i}{T}.$$

One natural strategy is to consider the sequence $b_i = i$ which increases the distance by one each step. In this case, $I(t) = t$ and $cost(T) = \sum_{i=1}^{T} i \approx T^2/2$. This means that the competitive ratio is unbounded!

A better strategy is to increase the distances more aggressively by *doubling* at each step. This corresponds to the sequence

$$2^0, 2^1, 2^2, \cdots 2^{i-1}, \cdots,$$

where the i^{th} number is $b_i = 2^{i-1}$. Here, we have $I(T) = j$ for the unique integer $j \geq 1$ with $2^{j-2} < T \leq 2^{j-1}$. Hence,

$$cost(T) = \sum_{i=1}^{I(T)} b_i = \sum_{i=1}^{j} b_i = \sum_{i=1}^{j} 2^{i-1} = 2^j - 1 \leq 4 \cdot T.$$

Therefore, the competitive ratio is at most 4.

Lower bound. The above 4-competitive algorithm turns out to be the best possible for any deterministic online algorithm. How do we prove such a "lower bound" on the competitive ratio? We first need to characterize online algorithms for the problem: for line search, this is simply an increasing sequence of numbers. Then, we need to show that for *every* such online algorithm ALG there *exists* some input sequence σ with a large ratio $\frac{\mathsf{ALG}(\sigma)}{\mathsf{OPT}(\sigma)}$.

We first provide a simple proof that no online algorithm for line-search has competitive ratio less than 2. Consider any online algorithm ALG given by the increasing sequence $\langle a_1, a_2, \cdots \rangle$. Suppose, for a contradiction, that its competitive ratio is at most 2. For any index i, when $T = a_i + \epsilon$ for an infinitesmal $\epsilon > 0$, we have $\mathsf{ALG}(T) = \sum_{j=1}^{i+1} a_j \leq 2 \cdot \mathsf{OPT}(T) = 2 \cdot a_i$. This implies

$$\sum_{j=1}^{i-1} a_j + 2 \cdot a_i \leq \sum_{j=1}^{i-1} a_j + a_i + a_{i+1} \leq 2 \cdot a_i, \qquad \forall i.$$

Rearranging, we get $\sum_{j=1}^{i-1} a_j \leq 0$ for all i, which is clearly impossible!

The proof for the correct constant 4 is more technical. Again, consider any online algorithm $\langle a_1, a_2, \cdots \rangle$. Suppose (for a contradiction) that the competitive ratio is $r < 4$. Let $S_j = \sum_{i=1}^{j} a_i$ denote the prefix sums. For any index j, with $T = a_{j+1} + \epsilon$ for infinitesmal $\epsilon > 0$, we have $s_{j+2} = cost(T) \leq r \cdot a_{j+1}$.

$$S_{j+2} \leq cost(T) \leq r(S_{j+1} - S_j) \tag{1}$$

We now claim that $\frac{S_{j+2}}{S_{j+1}} \leq \frac{S_{j+1}}{S_j}$, that is $S_j \cdot S_{j+2} - S_{j+1}^2 \leq 0$. By (1), it suffices to show

$$rS_j(S_{j+1} - S_j) - S_{j+1}^2 = rS_j \cdot a_{j+1} - S_{j+1}^2 \leq 0. \tag{2}$$

Let $S = S_j$ and $a = a_{j+1}$. So we need to show $rSa - (S + a)^2 \leq 0$. This is equivalent to

$$\left(S - (\frac{r-2}{2}) \cdot a\right)^2 + a^2 - \left(\frac{r-2}{2}\right)^2 a^2 \geq 0,$$

which is true because $r \leq 4$.

We showed above that the sequence $\left\{\frac{S_{j+1}}{S_j}\right\}$ is non-increasing. Moreover, this sequence is bounded below by one: so it has a limit $y \geq 1$. For a large enough j, dividing (2) by S_j^2 we have

$$0 \geq r\left(\frac{S_{j+1}}{S_j} - 1\right) - \left(\frac{S_{j+1}}{S_j}\right)^2 = r(y-1) - y^2$$

However, $y^2 - ry + r = (y - r/2)^2 + r(1 - r/4) > 0$ as $r < 4$, which is a contradiction!

To summarize,

Theorem 2.1 *There is a deterministic online algorithm for online bidding with competitive ratio 4. Moreover, no deterministic algorithm can do better.*

Next, we will see that this performance can be improved using *randomization*.

2.1 Randomized Line Search

We now revisit the line search problem using randomization. Can we achieve a better ratio if we choose a *random* sequence $\mathtt{S}$? Recall that the deterministic algorithm involves geometrically increasing values. We consider shifting these values by a random multiplicative offset, which leads to a smaller *expected* competitive ratio. Furthermore, to get the "right" constant, we also pick the base of the geometric series carefully.

Let $Z \in [0, 1]$ be a uniformly distributed random variable. Then, the random sequence $\mathtt{S}$ is

$$e^Z, e^{Z+1}, \cdots e^{Z+i}, \cdots,$$

where e is the base of the natural logarithm. We now analyze its (expected) competitive ratio.

We first bound the cost incurred. It will be convenient to consider the target T in logarithmic scale. Let $\ell = \lfloor \ln T \rfloor \geq 0$ and $f = \ln T - \ell \in [0, 1)$ be the integer and fractional parts of $\ln T$. Note that the number of steps/probes k performed by the above algorithm is the unique integer with

$$e^{Z+k-1} < T \leq e^{Z+k} \quad \Leftrightarrow \quad Z + k - 1 < \ln T \leq Z + k.$$

It follows that

$$k = \begin{cases} \ell + 1 & \text{if } 0 \leq Z < f \\ \ell & \text{if } f \leq Z \leq 1 \end{cases}.$$

In either case, the cost incurred is $\sum_{i=0}^{k} e^{Z+i} = e^Z \cdot \frac{e^{k+1}-1}{e-1} \leq \frac{e^{Z+k+1}}{e-1}$. Taking expectations,

$$\begin{aligned} \mathbb{E}[cost_{\mathtt{S}}(T)] &\leq \frac{e}{e-1} \cdot \mathbb{E}[e^{Z+k}] = \frac{e}{e-1}\left(e^{\ell+1}\int_{Z=0}^{f} e^Z dZ \;+\; e^{\ell}\int_{Z=f}^{1} e^Z dZ\right) \\ &= \frac{e^{\ell+1}}{e-1}\Big(e(e^f - 1) + (e - e^f)\Big) = e^{\ell+f+1} = e \cdot T. \end{aligned}$$

As this holds for all $T \geq 1$, we get:

Theorem 2.2 *There is a randomized e-competitive algorithm for line search.*

3 Randomized Rent-or-Buy (Ski Rental)

Recall tha ski-rental problem. The rent cost is 1 and buy cost is B. We need to decide whether to rent/buy for an unknown number of months T. As discussed earlier, any deterministic online algorithm is parameterized by a single "threshold" i, which denotes the months (of renting) before buying. We also saw that the "optimal" choice of i was $i = B - 1$, which lead to a competitive ratio of $2 - \frac{1}{B}$. We now ask the following question:

> Is there any benefit to *randomizing* the choice of threshold i?

It turns out that we can indeed obtain a smaller *expected* competitive ratio, where the expectation is taken over the random choice of i. In fact, for any online problem, we have the following general insight about randomized algorithms: *a randomized algorithm is a distribution over deterministic algorithms.*

Example. As an example, we first obtain a good distribution for the case $B = 4$. We construct the following table of payoffs where the rows correspond to deterministic algorithms ALG_i and the columns correspond to instances I_j.

	I_1	I_2	I_3	I_∞
ALG_0	4/1	4/2	4/3	4/4
ALG_1	1/1	5/2	5/3	5/4
ALG_2	1/1	2/2	6/3	6/4
ALG_3	1/1	2/2	3/3	7/4

While the real game is infinite along with coordinates, we do not need to put columns for $I_4, I_5, \ldots$ because these strategies for the adversary are dominated by the column I_∞. Now given that the adversary would rather give us only these inputs, we do not need to put rows after $B = 4$ since buying after month B is worse than buying on month B for these inputs. (Please check these for yourself!) This means we can think of the above table as a 2-player zero-sum game with 4 strategies each. The row player chooses an algorithm and the column player chooses an instance, then the number in the corresponding entry indicates the loss of the row player. Thinking along the lines of the Von Neumann minimax theorem, we can consider mixed strategies for the row player to construct a randomized algorithm for the ski rental problem.

Let p_i be the probability of our randomized algorithm choosing row i. What is the expected cost of this algorithm? Suppose that the competitive ratio was at most c in expectation. The expected competitive ratio of our algorithm against each instance should be at most c, so this yields the following linear constraints.

$$\begin{aligned}
4p_0 + p_1 + p_2 + p_3 &\leq c \\
\frac{4p_0 + 5p_1 + 2p_2 + 2p_3}{2} &\leq c \\
\frac{4p_0 + 5p_1 + 6p_2 + 3p_3}{3} &\leq c \\
\frac{4p_0 + 5p_1 + 6p_2 + 7p_3}{4} &\leq c
\end{aligned}$$

We would like to minimize c subject to $p_0 + p_1 + p_2 + p_3 = 1$ and $p_i \geq 0$. It turns out that one can

do this by solving the following system of *equations*.[1]

$$\begin{aligned} 4p_0 + p_1 + p_2 + p_3 &= c \\ 4p_0 + 5p_1 + 2p_2 + 2p_3 &= 2c \\ 4p_0 + 5p_1 + 6p_2 + 3p_3 &= 3c \\ 4p_0 + 5p_1 + 6p_2 + 7p_3 &= 4c \end{aligned}$$

along with $p_0 + p_1 + p_2 + p_3 = 1$. Subtracting each line from the previous one gives us

$$\begin{aligned} 4p_0 + p_1 + p_2 + p_3 &= c \\ 4p_1 + p_2 + p_3 &= c \\ 4p_2 + p_3 &= c \\ 4p_3 &= c, \end{aligned}$$

This gives us $p_3 = c/4$, $p_2 = (3/4)(c/4)$, etc., and indeed

$$p_i = (3/4)^{3-i}(c/4) \text{ for } i = 0, 1, 2, 3.$$

Using $p_0 + p_1 + p_2 + p_3 = 1$, we get

$$c = \frac{1}{1 - (1 - 1/4)^4}.$$

Distribution for general B. We now consider general B. It suffices to consider the inputs $I_1, I_2, \cdots I_{B-1}$ and I_∞. This is because any algorithm ALG_i performs worse on instance I_∞ than instances $I_B, I_{B+1}, \cdots$. Given that the only relevant instances are as above, it suffices to consider the algorithms $\mathsf{ALG}_0, \cdots \mathsf{ALG}_{B-1}$. Note that the cost of ALG_i on instance I_j is j if $j \leq i-1$ and $i + B$ if $j \geq i$.

For each $i \in \{0, 1, \cdots B-1\}$ let p_i be the probability that the randomized algorithm follows ALG_i. If the (expected) competitive ratio is at most c then for each $j \in \{1, 2, \cdots B-1, \infty\}$,

$$\sum_{i=0}^{j-1} (i + B) \cdot p_i \; + \; j \cdot \sum_{i=j}^{B-1} p_i \;\; \leq \;\; c \cdot \min(j, B). \tag{3}$$

Setting the inequalities to equalities and taking successive differences, we obtain B equations in B variables:

$$B \cdot p_j + \sum_{k=j+1}^{B-1} p_k \;\; = \;\; c, \quad \forall 0 \leq j \leq B-1.$$

This can be solved to obtain a unique solution:

$$p_i = \frac{c}{B} \left(1 - \frac{1}{B}\right)^{B-1-i}, \quad \forall 0 \leq i \leq B-1.$$

Finally, note that we also need to ensure that $\sum_{i=0}^{B-1} p_i = 1$ to get a valid distribution. This implies that:

$$\frac{c}{B} \sum_{i=0}^{B-1} = c\left(1 - (1 - 1/B)^B\right) = 1,$$

[1] Why is it OK to set the inequalities to equalities? Simply because it works out: in essence, we are guessing that making these four constraints tight gives a basic feasible solution—and it turns out to be right.

which means

$$c = c_B = \frac{1}{1-(1-\frac{1}{B})^B} \leq \frac{e}{e-1}.$$

Thus, we have:

Theorem 3.1 *There is a randomized $\frac{e}{e-1}$-competitive algorithm for ski rental.*

4 Randomized Lower Bounds

In order prove limits (or lower bounds) on randomized algorithms, we need to show that *for every* randomized algorithm $\mathcal{R}$, *there exists* an instance on which $\mathcal{R}$ performs poorly. As randomized algorithms are much more complex than deterministic ones, it is often difficult to carry out such an argument. Recall that a randomized algorithm is a distribution over deterministic algorithms. We now discuss an alternative approach (called Yao's lemma) to proving such lower bounds. This involves showing that *there exists* a distribution on instances/inputs such that *every* deterministic algorithm performs poorly. It turns out that such a statement is often much easier to prove. We will also apply this approach to the ski-rental and online-bidding problems.

Let I denote instances of an online problem and A denote deterministic online algorithms for it. Let $C(I, A)$ denote the cost incurred by algorithm A on instance I. For any instance I, let $\mathsf{OPT}(I)$ denote the optimal value of I. Any *randomized* online algorithm is given by a probability distribution $\{p_A\}$ over deterministic algorithms, and its competitive ratio is

$$\max_I \frac{\sum_A p_A \cdot C(I,A)}{\mathsf{OPT}(I)}.$$

To avoid technicalities, we assume that the number of possible instances and algorithms is finite. So, we can write the following linear program to calculate the best randomized competitive ratio.

$$\begin{aligned} \lambda^* \quad = \quad \min \quad & \lambda && \text{(LP)} \\ s.t. \quad & \lambda \cdot \mathsf{OPT}(I) - \sum_A C(I,A) \cdot p_A \geq 0, \qquad \forall I && (4) \\ & \sum_A p_A \geq 1, \\ & \lambda, p \geq 0. \end{aligned}$$

Taking the dual, we obtain:

$$\begin{aligned} \max \quad & \mu && \text{(DLP)} \\ s.t. \quad & \mu - \sum_I C(I,A) \cdot y_I \leq 0, \qquad \forall A && (5) \\ & \sum_I \mathsf{OPT}(I) \cdot y_I \leq 1, && (6) \\ & \mu, y \geq 0. \end{aligned}$$

By (weak) LP duality it follows that any feasible solution of (DLP) has objective at most λ^*. Consider *any* probability distribution $\mathcal{D} = \{\rho_I\}$ over instances I. Note that the expected optimal

value (over $\mathcal{D}$) is $O = \sum_I \mathsf{OPT}(I) \cdot \rho_I$. Further, suppose that the expected cost of *every* deterministic algorithm A is at least μ times the expected optimal value. That is,

$$\sum_I C(I, A) \cdot \rho_I \geq \mu \cdot O, \qquad \forall A.$$

Then, we construct a solution (μ, y) to (DLP) as follows. Define $y_I = \rho_I / O$ for all instances I. Clearly, $\sum_I C(I, A) \cdot y_I = \frac{1}{O} \sum_I C(I, A) \cdot \rho_I \geq \mu$ for all A, by the above assumption. So (5) is satisfied. Moreover, (6) is satisfied as an equality by definition of O. So (μ, y) is feasible to (DLP), which implies $\lambda^* \geq \mu$. Note that this is a lower-bound on the competitive ratio.

We now discuss another (slightly different) way of getting lower bounds. Let us scale down each constraint of (4) by $\mathsf{OPT}(I)$ (note that this does not affect the optimal (LP) value λ^*) and write the corresponding dual:

$$\begin{aligned} \max \quad & \mu && \text{(DLP')} \\ s.t. \quad & \mu - \sum_I \frac{C(I, A)}{\mathsf{OPT}(I)} \cdot z_I \leq 0, \qquad \forall A \\ & \sum_I z_I \leq 1, \\ & \mu, z \geq 0. \end{aligned}$$

Consider *any* probability distribution $\mathcal{D} = \{\rho_I\}$ over instances I. Suppose that the expected competitive ratio (taken over $\mathcal{D}$) of *every* deterministic algorithm is at least μ. That is,

$$\sum_I \frac{C(I, A)}{\mathsf{OPT}(I)} \cdot \rho_I \geq \mu, \qquad \forall A.$$

Then, we construct a solution (μ, z) to (DLP') by setting $z_I = \rho_I$ for all instances I. Clearly, all constraints in (DLP') are satisfied. By weak LP duality, $\lambda^* \geq \mu$ which provides a lower-bound on the competitive ratio. To summarize,

Theorem 4.1 (Yao's lemma) *Consider any online problem Π. Suppose there is a distribution $\mathcal{D}$ over instances of Π such that one of the following holds:*

- *every deterministic online algorithm for Π has expected cost (over $\mathcal{D}$) at least μ times the expected optimal value $\mathbb{E}_{I \leftarrow \mathcal{D}}[\mathsf{OPT}(I)]$.*
- *every deterministic online algorithm for Π has expected competitive ratio (over $\mathcal{D}$) at least μ.*

Then, the competitive ratio of every randomized online algorithm for Π is at least μ.

4.1 Online Bidding

Recall that an instance of online bidding is specified by a single number $T \geq 1$ (we allow continuous values to keep calculations simple here). So an instance distribution is the same as a distribution on this number T. Let U be a large parameter and let $\mathcal{D}$ be the distribution with density function $\frac{W}{t^2}$ for $t \in [U, U^2 \ln U]$. We set $W \approx U$ so that $W \int_{t=U}^{U^2 \ln U} \frac{dt}{t^2} = 1$.

The expected optimal value (under $\mathcal{D}$) is $E = \mathbb{E}_{T \leftarrow \mathcal{D}}[T] = \int_U^{U^2 \ln U} t \cdot \frac{W}{t^2} dt \approx U \ln U$.

We now show every deterministic algorithm, given by a sequence $\langle b_1, b_2, \cdots b_k\rangle$, has expected cost (under $\mathcal{D}$) at least $\approx e \cdot E$. By Theorem 4.1, this would prove that no randomized algorithm for online bidding has competitive ratio smaller than e.

We now calculate the expected cost of the deterministic algorithm $\langle b_1, b_2, \cdots b_k\rangle$. We can assume that the minimum $b_1 = U$ and the maximum $b_k = U^2 \ln U$ as $U \le T \le U^2 \ln U$. Note that we incur the cost b_i for the i^{th} distance iff $T > b_{i-1}$. So the expected cost is:

$$A = \sum_{i=1}^{k} b_i \cdot \Pr[T > b_{i-1}] \approx U \sum_{i=1}^{k} \left(\frac{b_i}{b_{i-1}} - \frac{b_i}{U^2 \ln U} \right),$$

where we used $\Pr[T > \beta] = \int_{\beta}^{U^2 \ln U} \frac{W}{t^2} dt \approx U \cdot \left(\frac{1}{\beta} - \frac{1}{U^2 \ln U} \right)$.

Let h index the first number with $b_h > U^2$. For each $i \le h$, we have $\frac{b_i}{b_{i-1}} - \frac{b_i}{U^2 \ln U} \ge \frac{b_i}{b_{i-1}} \left(1 - \frac{1}{\ln U}\right)$ because $b_{i-1} < U^2$. So,

$$A \ge (1 - o(1)) \cdot U \sum_{i=1}^{h} \frac{b_i}{b_{i-1}}.$$

It remains to lower-bound $A' = \sum_{i=1}^{h} \frac{b_i}{b_{i-1}}$ over increasing sequences with $b_1 = U$ and $b_h \ge U^2$. This can be done by basic calculus. Fix any $h \ge 1$ and let $\delta_i := \ln(b_i / b_{i-1})$ for all $1 \le i \le h$. Note that all $\delta_i \ge 0$, $\sum_{i=1}^{h} \ln \delta_i = \ln \left(\frac{b_h}{b_1} \right) \ge \ln U$ and $A' = \sum_{i=1}^{h} e^{\delta_i}$. We want:

$$\min \sum_{i=1}^{h} e^{\delta_i} \ : \ \sum_{i=1}^{h} \delta_i \ge \ln U, \ \delta \ge 0.$$

By convexity of the objective, the minimum occurs when each $\delta_i = \frac{\ln U}{h}$. So the minimum value of A' (for a fixed h) is $h \cdot \exp((\ln U)/h) = h \cdot U^{1/h}$. Minimizing over h,

$$A' \ge \min_{h} (h U^{1/h}) = (\ln U) U^{\frac{1}{\ln U}} = e \cdot \ln U.$$

Therefore, we obtain $A \ge (1 - o(1)) \cdot U \cdot A' \ge (e - o(1)) \cdot U \ln U \ge (e - o(1)) \cdot E$, as claimed.

4.2 Ski Rental

Recall that it costs 1 to rent and B to buy. The time horizon T is unknown to the online algorithm. We consider distribution $\mathcal{D}$ on instances (for fixed B) by choosing T as follows:

$$\Pr[T = k] = \rho_k := \frac{1}{B} \left(1 - \frac{1}{B}\right)^k, \quad \forall k = 0, 1, \cdots .$$

To keep things clean, we consider an unbounded distribution: but it can be truncated beyond B at a small loss in the ratio. Let $r = 1 - \frac{1}{B}$. The expected optimal value is:

$$E = \sum_{k \ge 0} \rho_k \cdot \min(k, B) = B - \sum_{k=0}^{B} (B - k) \rho_k = B - \sum_{k=0}^{B} r^k + \frac{1}{B} \sum_{k=0}^{B} k r^k = Br - Br^{B+1} \approx (B-1)(1 - \frac{1}{e}).$$

Above, we used $\sum_{k=0}^{B} r^k = \frac{1 - r^{B+1}}{1 - r} = B(1 - r^{B+1})$ and $\sum_{k=0}^{B} k r^k = B^2 (r - 2r^{B+1})$.

We now show that the expected cost (under $\mathcal{D}$) of every deterministic algorithm is at least $\frac{e}{e-1} \cdot E$, which would imply (using Theorem 4.1) a randomized lower-bound of $\frac{e}{e-1}$.

Consider any deterministic algorithm, given by number i, that rents for up to i days and then buys. Its expected cost is:

$$cost(i) = \sum_{k=0}^{i} k\rho_k + (i+B)\sum_{k>i} \rho_k = \frac{1}{B}\sum_{k=0}^{i} kr^k + (i+B)\left(1 - \frac{1}{B}\sum_{k=0}^{i} r^k\right)$$

Note that $\sum_{k=0}^{i} r^k = B(1-r^{i+1})$ and

$$\sum_{k=0}^{i} kr^k = \frac{r(1-r^i)}{(1-r)^2} - \frac{ir^{i+1}}{1-r} = B^2 r(1-r^i) - iBr^{i+1}.$$

So,

$$cost(i) = Br(1-r^i) - ir^{i+1} + (i+B)r^{i+1} = Br = B - 1.$$

This shows that no randomized online algorithm for ski-rental can have competitive ratio better than $\frac{e}{e-1} - o(1)$.