1 Absolute Competitiveness

We say that an algorithm $\mathcal{A}$ is *absolutely C-competitive* if

$$cost_{\mathcal{A}}(I) \leq C\, cost_{\mathrm{OPT}}(I)$$

for any input I.

2 Ski Rental

You really don't like to ski. However, a business associate of yours has invited you to go skiing, and you cannot refuse. It is possible that, in the future, there will be more such invitations, but you have no way of estimating how many.

It costs one dollar to rent skis, and it costs N dollars to buy skis. Should you rent or buy? If you buy, you will never need to rent, but if you rent, you will faced with the same decision if and when you are invited to go skiing again.

In this section, when we refer to competitiveness, we shall mean absolute competitiveness.

Let T be the total number of times you will be invited skiing. The optimal strategy is to rent skis if $T < N$, and to buy skis if $T > N$. Either strategy is optimal if $T = N$. But you don't know T.

Since the adversary (your business associate) could invite you skiing N times, the worst case cost cannot be less than N. On the other hand, by buying skis upon the first invitation, you can ensure that you will pay no more than N. Thus, the minimum worst case cost is N, and the strategy to achieve this is to buy skis upon the first invitation.

The *cruel adversary* will stop inviting you as soon as you buy skis. You might feel very foolish every time you open your closet and see that pair of almost unused skis. The competitiveness of the "buy immediately" strategy is N, namely your cost, N, divided by the optimal cost, which is only 1, since you should have rented.

2.1 A 2-Competitive Strategy

Let $\mathcal{A}$ be the following online algorithm. Upon the t^{th} invitation to go skiing, buy skis if $t \geq N$, otherwise rent.

Lemma 2.1 *$\mathcal{A}$ is 2-competitive.*

Proof: Let T be the total number of invitations. Then, assuming N is an integer[1]

$$cost_{\mathcal{A}} = \begin{cases} T & \text{if } T < N \\ 2N-1 & \text{otherwise} \end{cases}$$

$$cost_{\text{opt}} = \begin{cases} T & \text{if } T < N \\ N & \text{otherwise} \end{cases}$$

[1]The proof can easily be adapted to the case that N is not an integer.

Thus, the largest possible value of $\frac{cost_{\mathcal{A}}}{cost_{\text{opt}}}$ is $\frac{2N-1}{N} < 2$. □

Since N could be very large, we say that the ski-rental problem is 2-competitive.

2.1.1 Randomized Strategies for the Ski Rental Problem

You might ask, how could the adversary know that you bought skis? The usual assumption, based on the philosophy of considering the worst case, is that the adversary is omniscient and has unlimited computational power; therefore, he knows when you will buy skis, even before he issues his first invitation. We call this adversary the *strong* adversary.

It can be argued that this is unrealistic. The *oblivious* adversary knows your algorithm, but does not see what you actually do. By reading the code of your algorithm, the oblivious adversary can, however, *predict* exactly what you will do, and thus is just as bad as the strong adversary – unless you use randomization.

Lemma 2.2 *There is a randomized algorithm for the ski rental problem that is* $\frac{e}{e-1} \approx 1.582$ *competitive against the oblivious adversary.*

Exercise 2.1 *Let $\mathcal{A}$ be the following randomized algorithm for the ski rental problem.*

When invited to go skiing for the t^{th} time, and you have not yet bought skis:

- *If $t \geq N$, then buy skis.*
- *If $t < N$, then buy skis with probability*

$$\frac{e^{t/N} - e^{(t-1)/N}}{e - e^{(t-1)/N}}$$

The above probability is conditional, i.e.,*it assumes that you have not already bought skis. Another way to express the strategy is that if you have already been invited t times, the probability that you have bought skis is*

$$p(t) = \begin{cases} \frac{e^{t/N}-1}{e-1} \text{ if } t \leq N \\ 1 \text{ if } t \geq N \end{cases}$$

Prove that $\mathcal{A}$ is $\frac{e}{e-1}$-competitive against the oblivious adversary.

3 Asymptotic Competitiveness

For problems where the possible optimal cost is unbounded, we usually measure competitiveness asymptotically. An online algorithm $\mathcal{A}$ is defined to be *asymptotically C-competitive* if there is some constant K such that, for any request sequence ρ,

$$cost_{\mathcal{A}}(\rho) \leq C\, cost_{\mathrm{OPT}}(\rho) + K$$

Asymptotic competitiveness is weaker than absolute competitiveness. In most online problems (the ski-rental problem is a notable exception) the word "competitiveness" is used to denote asymptotic competitiveness. Unless otherwise specified, we will always mean asymptotic competitiveness.

4 The "Cow" Problem

A cow is in a pasture bounded by a long fence, which contains one hole. Knowing that the grass is always greener on the other side of the fence, the cow wants to find the hole. The cow is facing the fence. The optimal behavior is to walk straight to the hole, and the optimal cost is the distance to the hole.

But the cow does not know whether the hole is to the left or right. Being nearsighted, the she cannot see the hole until she is right next to it. What should she do?

One strategy is for the cow to pick one direction and walk only in that direction until she finds the hole. If she guesses correctly, her cost is optimal, but otherwise, her cost is infinite.

Our problem is to minimize competitiveness, which is defined as the worst case ratio of the length of the cow's path to the distance to the hole. What is minimum competitiveness that can be achieved?

Exercise 4.1 *Let $\mathcal{A}$ be the following deterministic algorithm for the cow.*

1. *Pick a number $\lambda > 1$.*
2. *Walk one foot (American, Liberian, and Burmese cows only) or meter (all other cows) to the left.*
3. *Walk back to your starting point, then walk λ feet to your right.*
4. *Walk back to your starting point, then walk λ^2 feet to your left.*
5. *Continue these steps, alternating left and right excursions, where on each excursion, you walk λ times as far as your last one.*
6. *Break off the loop when you encounter the hole.*

Find the value of λ which minimizes the (asymptotic) competitiveness.

Going back to the definition of competitiveness, we cannot allow K to be zero, since if the adversary places the hole one nanometer to the right of the starting point, the cow pays two billion times optimal. We need to give the cow a little slack, and so we must pick a positive K. We will arbitrarily assume that $K = 1$.

Exercise 4.2 *Find a randomized algorithm for the cow problem whose competitiveness against the oblivious adversary is less than the best competitiveness of any deterministic algorithm.*