

Final. Due 11/12. You can spend at most 24 hours on this test. Note: your final should be carefully written and detailed. (For example, don't just say, "Oh yeah, and with dynamic programming, we can do this in time $O(\text{Blah})$." Explain how.) You must work alone, but may consult the course notes and other non-human resources (but may not of course just look up the answer on the web). This test is scored out of 100 points. (I realize that the maximum total possible score is 170 points, but anything over 100 will be extra credit.)

1. (25 points) Lempel-Ziv compression. Consider the sequence:

11011100101110111

The above is the concatenation of the binary representation of the numbers $1, 2, 3, 4, \dots$. Consider S_T to be the concatenation of the binary numbers 1 through T . About how long is the string S_T as a function of T (try not to be off by more than a constant factor)? About how long will the Lempel-Ziv encoding of S_T be? Which is longer?

2. (50 points) Online averaging. Consider the problem of estimating the average of a sequence of real numbers $x_1, x_2, \dots, x_T \in [-1, 1]$. In an online version, on period t , we see x_1, x_2, \dots, x_{t-1} and we choose a_t based on these numbers. Then we find out x_t . This repeats for periods $1, 2, \dots, T$. Let's say that we pay a cost of $(a_t - x_t)^2$. The goal is to have low regret, where regret is defined as:

$$\text{regret} = \sum_{t=1}^T (a_t - x_t)^2 - \min_{a \in \mathbb{R}} \sum_{t=1}^T (a - x_t)^2.$$

- (a) (5 points) What is the value of $a \in \mathbb{R}$ that minimizes the quantity $\sum_{t=1}^T (a - x_t)^2$? (No need to prove correctness.)
- (b) (10 points) Explain how the online averaging problem can be put into Zinkevich's online convex optimization framework. What is the bound on regret of his algorithm and analysis as a function of T ? (I only care about the order, is it $O(T^{1/2})$, $O(T^{1/4})$, etc.)
- (c) (20 points) Consider the following algorithm: choose

$$a_t = \frac{x_1 + x_2 + \dots + x_{t-1}}{t}.$$

Prove that this algorithm never has a regret that is larger than $O(\log T)$. (Hint: you can do this either directly or by comparing yourself to the imaginary “be the leader” algorithm that chooses $a_t = \frac{x_1+x_2+\dots+x_t}{t}$, and arguing that this imaginary algorithm has regret ≤ 0).

- (d) (15 points) Consider the case where the cost is $|a_t - x_t|$ on period t . What is the best choice of a to minimize $\sum_{t=1}^T |a - x_t|$? What is the regret bound from using Zinkevich’s algorithm here? (Don’t worry about the fact that $f_t(a) = |a - x_t|$ is not differentiable at the single point $a = x_t$.) Extra credit: Can you do better? Prove yes or no.

3. (40 points) Rent or buy. Recall, from the first lecture, the rent or buy problem, where the cost of renting was R and the cost of buying was $B > R$. Each time you go skiing (and haven’t bought skis yet), you must decide to rent or buy. Once you buy, you no longer have to pay anything for skis. The problem is that you don’t know, in advance, how many times you go skiing.

If one follows the deterministic strategy of renting $B/R - 1$ times and then buying, one gets a competitive ratio of at most 2 (in the limit as B/R grows without bound, this is 2). Give a randomized algorithm that achieves $e/(e - 1)$ competitive ratio.

4. (15 points) k -server problem. Recall that caching (also called paging) can be thought of as a special case of the k -server problem, where k is equal to the size of the cache and the cost of moving from one page to another is 1. In class, we saw that for any problem, the deterministic work-function algorithm is $2k - 1$ competitive. On specific metric spaces, the work-function algorithm may have a better competitive ratio than on others. To within a constant factor, what is the competitive ratio of the work function algorithm we discussed in class for the special case of caching, as a function of k ? (In other words is it $O(k)$ or $O(\sqrt{k})$ or $O(\log k)$?) Why?
5. (40 points) Weighted majority. Recall the prediction from expert advice problem. Each period, we have to choose one of n experts. After we make our choice, each expert receives a reward of either 0 or 1, these rewards are revealed, and we receive the reward of the expert we chose.

The Weighted Majority algorithm maintains weights $w_i^t > 0$ for each expert $i \in \{1, 2, \dots, n\}$ and each period t . The update rule is that initially each weight is 1, $w_i^1 = 1$, and each time an expert receives a reward of 1, its weight is doubled, otherwise it remains the same. In other words, $w_i^{t+1} = 2w_i^t$ if expert i received reward 1 on period t and $w_i^{t+1} = w_i^t$ if expert i received reward 0 on period t . On period t , the algorithm chooses a random expert with probability of choosing expert i equal to $w_i^t / \sum_{j=1}^n w_j^t$. Consider a sequence where every expert receives exactly R total reward. So, the reward of the best expert on this sequence is exactly R . What is the worst kind of sequence of this type for the weighted majority algorithm? In other words, give a sequence of rewards that would cause the weighted majority algorithm to get the least possible total reward, subject to the constraint that each expert achieves total reward R . I don't care about the length of the sequence, but just the performance of weighted majority as a function of R . Prove that your answer is correct. Extra credit: evaluate the performance of weighted majority on this sequence.