6.854 Online Algorithms Solutions

Contact: David Zhang – dzhang [at] cs [dot] toronto [dot] edu
All questions are taken from Karger's 6.854 Advanced Algorithms Homework (link):

1. You have reached a river (modeled as a straight line) and must find a bridge to cross it. The bridge is at some integer coordinate upstream or downstream

   (a) Give a 9-competitive deterministic algorithm for optimizing the total distance travelled up and downstream before you find the bridge. This is optimal for deterministic strategies.

   The algorithm is as follows:

   – Choose an initial direction;
   – Start from $i = 0$. Then, execute *step* i. If we don't find the bridge then increment i and repeat.

   On *step* i, travel 2^i units in the current direction and terminate immediately if we reach the bridge. Otherwise flip around and travel 2^i units in the opposite direction to return to the origin. In each step, we travel 2^{i+1} units.

   In the worst case, we fall just short of discovering the bridge at position b. That is, we need to travel twice the distance on the other side before returning to the origin and finding b in the next step. Hence, the distance that we travel following the step that almost reaches b is $< 5b$. Suppose that we almost reach b in step j. Then, $2^j < |b|$ so $j < \log b$, i.e. we take at most $\log b$ steps to almost reach b. Hence, the maximum distance that we will ever travel is bounded by $5b + \sum_{i=0}^{\log b} 2^{i+1} = 5b + 2(2^{\log b+1} - 1) < 9b$, yielding the desired ratio.

   (b) Give a randomized 7-competitive algorithm for the problem.

   Our algorithm suffers if we choose the wrong initial direction. Remedy this by flipping a fair coin to decide whether to explore up or downstream first. Suppose that $2^i < |b| \leq 2^{i+1}$. In the worst case, step i explores in the same direction of b and we remain 9-competitive. If we choose the right direction, then step i explores in the opposite direction and we will reach b during step $i + 1$. Hence, the total distance that we travel is bounded by $b + \sum_{i=0}^{\log b} 2^{i+1} = b + 2(2^{\log b+1} - 1) < 5b$. We traverse at most $5b/2 + 9b/2 = 7b$ units in expectation, so this 1 bit randomized algorithm is 7-competitive.

2. Consider an online version of the Knapsack problem where you get items one at a time and have to decide whether to put them into your Knapsack. If you run out of space you need to discard something and can never get it back later. What sort of competitive ratio can you achieve versus the offline (optimal) knapsack? WLOG, assume the knapsack has size 1 and each item has size no greater. Consider the following two algorithms: MAX, which simply keeps the highest value item, and GREEDY, which gives each item a *density* equal to its value divided by its size, and keeps as many of the highest density items as possible.

   (a) Prove that neither GREEDY nor MAX is competitive.

   – MAX: Send $1/\epsilon^2$ items, each with $s_i = \epsilon^2$ and $v_i = \epsilon$, followed by a single item of unit size and value. MAX will pack the unit item while an optimal knapsack packs the $1/\epsilon^2$ items to achieve $1/\epsilon$ profit. MAX is not competitive since decreasing ϵ will make the ratio arbitrarily large.
   – GREEDY: Send a single job with unit size and value, followed by a job with ϵ^2 size and ϵ value. A unit capacity knapsack can only accept one of the two jobs – GREEDY accepts the one with ϵ value (having density $1/\epsilon$) while an optimal knapsack accepts the item with unit value. Again, decreasing ϵ makes the ratio arbitrarily bad. Hence, GREEDY is not competitive.

   (b) Consider *fractional GREEDY* which is able to discard or keep arbitrary *fractions* of items (of high density). Prove that it does at least as well as the (integer) optimum.

   In the fractional setting, if the total weight of items that OPT packs is < 1, then the total weight across all items must also be < 1. In this case, our algorithm would just need to accept every item whole to be optimal. Thus, it is more interesting to consider when the sum of all item weights is ≥ 1. In this case, OPT will maximize the sum of densities of its accepted items, which will just be the total value that it packs since its final weight sum will be 1.

To maximize density, accept the densest items until the packed weight exceeds 1. Then, remove fractions (or whole amounts) of the least dense items to reduce the total weight to 1. Fractional GREEDY can do this in an online fashion by removing fractional (or whole) amounts of items with the smallest density, whenever it encounters an item whose density is greater than the smallest density in the set. Hence, the solution that fractional GREEDY produces is identical to the optimal offline solution, which is at least as good as the integer OPT since the fractional variant is a relaxation of it.

(c) Prove that randomly choosing between GREEDY and MAX is a 2-competitive strategy. **Hint:** compare to fractional GREEDY. Which will have how many fractional items?

Note that GREEDY and fractional GREEDY will accept the same items, but GREEDY will completely discard the least dense item while fractional GREEDY will keep a fraction of it. Further, note that the value provided by this least dense item is at most the value of the item that MAX accepts, i.e. it is bounded by the maximum. Hence, the sum of values from items accepted by GREEDY or MAX is $> OPT$. Since we choose each strategy with $1/2$ probability, it follows that we capture at least half of the optimal knapsack's value in expectation.