Code Jam 2008 - AMER Semifinal

Analysis: Test Passing Probability

The Problem

There are 4^Q points in the space, each of which contains an *answer combination* -- one answer for each of the **Q** questions. Because the answers to the questions are independent of each other, the probability of success at each point can be computed simply as the product of the probabilities from each problem.

So the simple mathematical model involves 4^Q points, each with a probability value. There is a single, hidden good point among the 4^Q points; you may pick up to **M** points; and you win the game if one of them is the good point. After you select a point, you learn if it is good or not, but that does not change the relative probabilities of the remaining points. The best strategy is simply looking at the points in the order of their probability. Your task in this problem is to compute the summation of the highest (up to) M among the 4^Q values.

Solutions

While Q looks moderate, 4^Q steps of computation can be a huge task that is probably not within your computer's capacity (or any computer's). The important restriction in this problem is the fact that M is also moderate, and we can compute the highest M values one by one.

One solution uses a priority queue **L** that keeps all the answer combination candidates for the *next* highest value -- after we computed the first **k** values. A bit of thought should convince your that the next-highest (the $k+1^{th}$) combination must be gotten by taking one of the first k combinations, and moving the answer to one of the questions one step "worse", to the next-most-probable answer. So in each step we can get the first element **S** (the one with the highest probability) from L, then add (up to) Q new candidates back to L. Each of the new candidates is created by moving one answer in S one step worse. The size of L will never exceed MQ, and the time complexity is O(MQ log MQ).

Another solution has an even simpler implementation: the idea is to add the questions one by one. For the first **k** questions, there are 4^k possible solution combinations, each with its own probability of answering all of the first k questions correctly. When we add the $k+1^{th}$ question, the new set of possible solution combinations, along with its probabilities, can be calculated by taking each of the 4^k previous values and multiplying it by the four probabilities for each answer to question k+1. We can easily compute all of these values and truncate to the top M, then repeat the process for the remaining questions. The truncation is O(M log M), and this process is repeated Q times, for a total time complexity of O(QM log M).

More information

Discrete Probability - Joint Distribution