CS 561, HW 13

Prof. Jared Saia, University of New Mexico

1. Exercise 34.5-2 (0-1 Integer Programming)

2. **KEYMASTER** You are trapped in a maze filled with keys and locked doors! Your goal is to gather keys to open corresponding doors and reach the exit. You are given an undirected graph $G = (V, E)$ representing the maze. There are $\ell \leq n$ types of keys, $\{k_1, \ldots, k_\ell\}$, and ℓ types of doors, $\{d_1, \ldots, d_\ell\}$. Each node $v \in V$ contains at most one item denoted $C(v)$, where $C(v)$ is either one of the key types, one of the door types, or the empty set. There is also a special start node s, and exit node t in V.

   There are two traversal rules. First, when you visit a node that contains a key type, you automatically pick up that key. Second, you can only visit a node containing a door type d_i if you already hold a key of type k_i, and on first visiting that node, both the door at the node and 1 key of type k_i that you hold disappear. For a given input, the problem KEYMASTER returns YES iff there is a walk starting at s and ending at t that follows the above rules.

   Prove that KEYMASTER is NP-complete.

3. Rock, Paper, Scissors is a simple 2 person game. In a given round, both players simultaneously choose either Rock, Paper or Scissors. If they both choose the same object, it's a tie. Otherwise, Rock beats Scissors; Scissors beats Paper; and Paper beats Rock. Imagine you're playing the following betting variant of this game with a friend. When Scissors beats Paper, or Paper beats Rock, the loser gives the winner \$1. However, in the case when Rock beats Scissors, this is called a **smash**, and the loser must give the winner \$10.

   (a) Say you know that your friend will choose Rock, Scissors or Paper, each with probability 1/3. Write a linear program to calculate the probabilities you should use to maximize your expected winnings. Let p_1, p_2, p_3 be the variables associated with your optimal

probabilities for choosing Rock, Scissors and Paper respectively. Note: If you want to check your work, there are several free linear program solvers on the Internets: check the Wikipedia page on linear programming.

(b) Now say that your friend is smart and, also, semi-clairvoyant: she magically knows the exact probabilities you are using and will respond optimally. Write another linear program to calculate the probabilities you should now use in order to maximize your expected winnings. Hint 1: If your opponent knows your strategy, her strategy will be to choose one of the three objects with probability 1. Hint 2: Review the linear program for the shortest paths problem.

4. *X-STREAM Dance Dance Revolution (XDDR)* is played on a Pogo Stick while blind-folded. At the beginning of each round, you can hop from your current square to any square. However, the target sequence, σ is not known in advance: the value $\sigma[i]$ is announced only at the end of round i, for each $i \in [1, n]$, where n is the length of σ.

   Your goal is to minimize *cost*: the number of rounds $i \in [1, n]$ in which you hop in a square different than $\sigma[i]$.

   Below is an example game; your cost is 5 because there are 5 rounds where the square you hop in does not match the target square in σ.

σ	A	C	A	D	C	D	D
Pogo position	A	B	B	D	A	C	A
Cost?	0	1	1	0	1	1	1

   In round i, you let $\vec{x}_i$ be a length 4 vector giving a probability distribution over the 4 possible squares on which you will hop, i.e. $\vec{x}_i[1]$, $\vec{x}_i[2]$, $\vec{x}_i[3]$, $\vec{x}_i[4]$ are the probabilities of hopping into squares A, B, C, D respectively.

   Then, for round i, you define $f_i(x_i)$ as your expected cost in round i. In round i, let c_i be a length 4 vector giving the cost outcome: $c_i[j] = 0$ when j matches the square given by $\sigma[i]$ and $c_i[j] = 1$ otherwise. For example, in round i, if $x_i = [1/8, 1/2, 1/8, 1/4]$ and $c_i = [1, 1, 1, 0]$, then your expected cost for this round is $3/4$

   (a) Give the mathematical expression for f_i as a function of $\vec{x}_i$ and $\vec{c}_i$.

(b) Describe, algebraically, the convex search space κ. What is the diameter, D of κ?

(c) Zinkevich's theorem says that the cost of online gradient descent tracks the cost of the best offline solution, x^*. In particular, if OPT is the cost of the best offline solution, then the cost of our algorithm is at most $OPT + \sqrt{n}DG$. Give a precise 1 line definition of OPT for this problem using the c_i values.

(d) Now, you want to use some history. In particular, you notice that the current square in σ often depends on the last square. So you want to use the outcome of the last round to help set your probability distribution for the current round. To do this, how would you change the convex search space κ? How many dimensions does it now have? Will OPT, G and D likely increase or decrease? Will your algorithm's expected cost increase or decrease?