CS 3353 Fall 2024

Programming Homework 4 – Dynamic Programming and "19 keys"

Due date: 12/17 (Tue) 11:59pm. NO LATE PASS ALLOWED.

The goal of the homework is to apply dynamic programming to devise best strategy to "probabilistic" games and events. For this program, you are going to devise the strategy of variation of the UK game show "19 keys".

19 keys

The following is an edited version of the rule of "19 keys" as described at Wikipedia:

"Four contestants stood within a large, transparent plastic cube, one at each corner. They faced the host, who stood next to a safe and a tray of 19 numbered keys in the center. Only one key would open the safe. Before each contestant was a panel displaying the key numbers.

A cash jackpot was at stake, starting at zero and increasing at a continuous rate of £500 per minute for 15 minutes. If the jackpot reached its maximum of £7,500, it then began to decrease at a rate of £2,500 per minute for the next 3 minutes until it reached zero again. A game could thus last no more than 18 minutes.

Throughout the game, a contestant could eliminate wrong keys from their own panel by answering questions correctly, while a miss would relight them. The specific keys to be eliminated/relit were revealed only after an answer had been given

…..

At any time, a contestant could press a red button on their podium. They then had 10 seconds to select a key and try to open the safe with it, with the timer and money counter still running. If successful, they won the entire jackpot; if not, they were eliminated from the game. If the jackpot decreased all the way to zero before the safe was opened, all four contestants left with nothing."

A video of a game played is available for your viewing pleasure: Bing Videos (You don't have to watch this to understand the task, as the game will be significantly simplified for the homework).

Modelling the game (assumptions)

We can use dynamic programming to analysis the best strategy for the game.

To start out, we make a few assumptions to simplify the analysis:

- We describe the game as if it is a single contestant game.
- Instead of having a time limit, we assume in each game a total of 10 questions will be asked.
- The game starts with 9 keys and a jackpot of $0.
- The jackpot (all are positive integers) will increase after each of the first 6 questions is being answered (regardless of whether the answer is corrected or not). They will be denoted as j_1, j_2, … j_6. You can assume each of the $j_s > 0$ and $j_s < j_t$ if $s < t$. (The numbers are provided to the program). We can assume $j_0 = 0$

- After the 6th question, the jackpot will start decreasing by a fraction of f ($0 < f < 1$) for each question. So the jackpot for the 7^{th} question will be $f * j_6$, the 8^{th} question will be $f^2 * j_6$ etc.
- The contestant will start by answering the first question. If he/she answers the question correctly, one incorrect key will be removed from the panel. However, if he/she answers the question wrong, an incorrect key will be added by to the panel
- When a key is added, the keys that are still on the panel are shuffled, so the position of the correct key may change.
- If there are already 9 keys on the panel, no new keys will be added, even if the question is answered incorrectly (or in any other circumstances where keys are to be added – see extra credit).
- If there is only 1 key left on the panel, the key is not removed. (The key will be the correct key). Also (in extra credit) if at a step there are more keys to be removed that there are keys in the panel, 1 key (the correct key) will be left on the panel.
- After answering any question (and with keys added/removed), the contestant has the choice to randomly pick one key from the panel to try to open up the safe. If the safe opens, the contestant takes the current jackpot and win the game. Otherwise, the contestant leaves with 0 and the game ends. If the contestant does not choose, then he/she will move on to answer the next question, and the same criteria of adding/removing key applies for each subsequent question.
- If the contestant answered all questions (and with keys added/removed), he/she MUST randomly pick one key from the panel to try to open the safe. He/she wins the jackpot at that moment if the key is the correct one, and 0 otherwise.
- For each of the 10 questions, we assume the probability that a contestant chance of answers the questions correct are p_1, p_2, … p_{10} respectively. We assume $0 < p_i < 1$. However, there are no relationships between the p_is

Devising the strategy

Our goal is to figure out the optimal strategy for the game. In this homework we assume that we want to maximize the expected amount being won. We assume we are given the value of f, j_t ($1 <= t <= 6$) and p_i ($1 < i < 10$) already.

Notice that the amount of money that can be won depends on the number of questions you have answered (denote it as q) and number of keys left on the panel (denote it at k). Denote the expected amount to be won as Opt[q][k] ($0 <= q <= 10$, and $1 <= k <= 9$)

The following questions will help you come up with the recurrence relation for Opt[q][k] and also lead you to the correct algorithm. (Write all your answers in terms of p's and j's).

1. What is Opt[10][1]?
2. What is Opt[10][k] for $k > 1$?

3. Suppose we have answered q (q > 0) question, and there is k keys left on the panel.
    a. What is the expected amount I can win if I decide to pick a key?
    b. If I decided to answer a question, what will be the updated value for q and k if I
        i Answered correctly?
        ii Answered incorrectly?
    c. Based on the above, can you write down the recurrence relation for Opt[q][k] (for q > 0)?
4. Repeat part 3, for q = 0.

What to do (Base case)

You will be given a program (Prog4Main.java) that will read in a file that contains information about the game. The file has three lines

- The first line contains 10 numbers which are p_1, p_2, … p_{10} respectively
- The second line contains 6 numbers which are as j_1, j_2, … j_6
- The third line contains one number, which is f

You can assume all numbers are valid. If the numbers are not and your program crashed, you are not responsible.

You are also given a public class, Prog4Return which contains two 2D arrays:

- expected[s][t]: stored the expected amount that one would win if one have answered s questions, and with t keys
- action[s][t]: store what a contestant should do if one has answered s questions, and with t keys out there. 0 means one should answer the next question, 1 means the contestant should pick a key and open the safe.

You will need to implement the method in Prog4 class (Proj4.java)

```
public static Prog4Return Prog4(double [] p, double [] j, double f)
```

Where p, j, f are as described above, It should return the Prog4Return objects where all entries of the arrays are filled with the correct values.

Extra Credit

Early bird bonus (20%):

You get a 20% bonus of your score (including other extra credit) if you hand in your program by 12/15 (Sun) by 11:59pm

Extra Credit : (20 POINTS)

In many similar game shows, there are “lifelines” to help contestants. In this case, we assume there is 1 lifeline available:

- Safety key: You can use this lifeline once in your game, after you answered a question and keys are added/removed. You immediately select a key to try to open the safe. If it is the correct key, you win the corresponding money in the jackpot. However, if the key is not correct, then the game continues, but 1 extra incorrect key will be added, and the key one chose will also be returned to the panel. The contestant will then move on to answer the next question. Also, you cannot use lifeline once you answered all 10 questions.

You are also given a public class, Prog4ReturnE which contains two 3D arrays:

- expected[s][t][u]: stored the expected amount that one would win if one has answered s questions, and with t keys, u = 0 corresponds to the case that there is no lifeline available, 1 corresponds to the case that there is.
- action[s][t][u]: store what a contestant should do if one has answered s questions, and with t keys out there. 0 means one should answer the next question, 1 means the contestant should pick a key and open the safe, 2 means that the contestant should use the lifeline.

You will need to implement the method in Prog4 class (Proj4.java)

public static Prog4ReturnE Prog4E(double [] p, double [] j, double f)

Where p, j, f are as described above, It should return the Prog4ReturnE objects where all entries of the arrays are filled with the correct values.