Challenging Students with Creative Assignments

Michael Mitzenmacher*

December 27, 2000

Although I still generally use standard problem sets in my theory classes, I find myself experimenting more and more with alternatives, including more open-ended assignments. One big motivation for this change is to put some *science* back in my computer science class, by making assignments more like laboratory exercises in chemistry and biology: students have to experiment to find an answer. I also find that open-ended assignments offers more room for student creativity, although it is not clear to me that students prefer this. (Indeed, it is clear that some students resent it.) Or perhaps, more pessimistically, it is just my fear that the solutions of all the standard problems are available somewhere on the Web.

In an effort to encourage a conversation in the community about the topic, I thought I would present one of my recent assignments. The full text of the assignment lies below. (I have made only small modifications, so please forgive its rough form.) In one sentence, the thrust of the assignment is that the students should model the game show "Who Wants to be a Millionaire?" and use their model to determine the best player strategy. The problem is harder than it might seem, and it raises many interesting questions about how to design a reasonable model.

I gave this assignment to my graduate seminar on Probabilistic Analysis and Randomized Algorithms, a class primarily meant for first and second year graduate students in theory and networks. I belive that beginning graduate students and advanced undergraduates leaning towards graduate school should be learning how to develop models for complex systems. Designing proper abstractions is a useful skill for both theoreticians and non-theoreticians; this problem has the advantage of being easily understood by everyone.

While I think the assignment is far too open-ended for an undergraduate class, a simpler variation might make an interesting project. (At the very least, underlying it all is a reasonably well-motivated dynamic programming problem.) Student opinion was mixed, with the primary complaint centering on open-endedness of the assignment; one said that they could spend an arbitrary amount of time on it without getting closer to the final solution. I thought this was an appropriate metaphor for graduate school. The next biggest complaint was that students were not familiar with the show, which is somehow disturbing and refreshing at the same time.

If anyone wants to use this assignment or some variation in a class, you have my permission and encouragement. I would ask that you let me know how it goes and if you have any suggestions for how to improve it.

*Harvard University, Computer Science Department. 33 Oxford St., Cambridge, MA 02138. Supported in part by NSF CAREER Grant CCR-9983832 and an Alfred P. Sloan Research Fellowship. E-mail: michaelm@eecs.harvard.edu.

Who Wants to be a Millionaire?

1 Goals of the assignment

One of the hardest parts of research is coming up with reasonable models of complex systems and processes. Let's try modelling and solving an apparently basic but potentially complex problem from the world of TV game shows.

Let me clarify that this is the first time I have given the assignment, and I have no idea of what the "right" answer is. The assignment is deliberately open-ended. You may work in small groups if you wish.

2 The game

Let's come up with an optimal strategy for the game "Who Wants to be a Millionaire?"

Recall the game (in fact, go watch an episode or two if you have not seen it): there are fifteen questions, increasing in difficulty. The amount you have won after answering k questions correctly is given in Figure 1. Note that after correctly answering 5 questions, you are guaranteed to leave with \$1,000; after correctly answering 10 questions, you are guaranteed to leave with \$32,000. At each point, you are given a multiple choice question with four possible answers. You may choose to not answer the question, at which point you leave with the amount you have won. If you answer the question but are incorrect, you lose, and your winnings are reduced either to \$0, \$1,000, or \$32,000, as appropriate. If you answer correctly, Regis congratulates you, and you go on.

The game is made more complex by the use of lifelines. You have three lifelines available, each of which can be used at most once. The lifelines are:

1. 50/50: Remove two of the four choices. We assume that the two choices removed are selected randomly.
2. Phone a friend: You can call a friend, who will give you their opinion of the right answer.
3. Ask the audience: The audience is polled on what they think the right answer is.

3 General approaches

As a start, we might consider optimizing for expected winnings. (Already, this is a modeling assumption that is highly debatable. Not everyone wants to optimize for expected winnings... This is worth thinking about more.) Dynamic programming is a natural approach. That is, suppose you figure out the optimal strategy and expected winnings given you have reached the last question. Then you should be able to figure out the optimal strategy and expected winnings given you have reached the second to last question, and so on down the line back to the first question.

There are many aspects that need to be modelled, and hence may require parameters. For example:

Question	Winnings
1	100
2	200
3	300
4	500
5	1000
6	2000
7	4000
8	8000
9	16000
10	32000
11	64000
12	125000
13	250000
14	500000
15	1000000

Table 1: Table of winnings. After 5 questions, you are guaranteed to leave with $1,000; after 10 questions, you are guaranteed to leave with $32,000.

- How do we model what a question looks like? For example, perhaps there is some probability that we can discount one or two of the choices. Perhaps we think each answer is correct with some probability once we see the question. Perhaps questions should be divided into types (corresponding to sports, entertainment, etc.) and we have different abilities for each type. There is a lot of room here for creativity as well as mathematics.
- How do we model increasing difficulty of the questions?
- What is the effect of lifelines for our model?

An interesting question is how complex a model is appropriate. Specfically, since you will have to use this model, there may be a tradeoff between the complexity of the model and the difficulty in using it. One possible way to handle this is to develop a simple model and a more complex variation. If the two give nearly the same answers, then the simpler model is probably more useful; if there are deviations, then perhaps the complex model is necessary to capture important aspects of the game.

4 The assignment

Your assignment consists of many parts. The first:

- Develop one or more appropriate models. (From now on we will refer to your model, but you may have more than one.)

Given your model, you should test to see if it provides reasonable answers. This will require an implementation. Parameters to your model should be input parameters.

- Create an implementation based on your model.

Given your implemenation, you should develop some insight into the game. Remember that you should form hypotheses, test your hypotheses, and possibly **change your hypotheses or your underlying model** based on your results. Here, for example, are some possible hypotheses, but you should think of some on your own.

1. It is worth using up lifelines in order to reach the $32,000 level.
2. On TV, it seems that about 90% of the time people will win between $32,000 and $125,000. Based on this, most players appear to be playing near optimally.
3. The probability of winning one million dollars decreases substantially if you use even one lifeline early on.

- Create and test hypotheses based on your model.

Finally, based on your empirical results, you should develop some insight into how good your model is. What are its strengths? If you had more time, what might you improve? What results is your model good at deriving, and where are the potential holes?

- Describe the strengths and weaknesses of your model.

You will be assessed based on the elegance and richness of your model, the successful implementation of your model, the novelty of the results you find based on your model, and the insight you show in describing the strengths and weaknesses of your model.