COSC 6111 Advanced Design and Analysis of Algorithms
Jeff Edmonds
Assignment: Random Algorithms

First Person:
Family Name:
Given Name:
Student #:
Email:

Second Person:
Family Name:
Given Name:
Student #:
Email:

Do three of the following (that you have NOT seen a solution for before).

Problem Name	If Done Old Mark	Check if to be Marked	New Mark
1 Recursive Probability			
2 Random Walks			
3 Triangles			
4 Shoot Game			
5 Uncertainty			

COSC 6111 Advanced Design and Analysis of Algorithms
Jeff Edmonds
Assignment: Random Algorithms

1. Recursive Probability. I posted on facebook that I had speakers to give away. Three friends, Kunle, Hilary, and Michael said they wanted them. I decided to flip a coin to fairly decide who gets them. The question is how to do this. A dice would have been easier. Why? If you are not up on probability, think about the space of probability one as being a piece of paper of area one. I have a way of taking any piece of paper and cutting its area in exactly half. This gives pieces of area $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots \frac{1}{2^r}$. Then I must give each of my friends a pile of pieces whose area sums to $\frac{1}{3}$. If, for example, I cut each piece r times then I have 2^r pieces of the same size. But 2^r can't be divided evenly into three. Problem. We will develop two different recursive algorithms for this problem.

   (a) The first algorithm will be recursive and will be called the "Sam gives it aways" algorithm.

      i. To warm up for this algorithm, lets pretend that Sam responded too. The following then would be the solution.

      algorithm *Choose*4()

      ⟨***pre−cond***⟩**:** There are no inputs, but we do flip coins.

      ⟨***post−cond***⟩**:** We output Kunle, Hilary, Michael, or Sam with equal probability

      begin
      flip a coin twice getting one of HH, HT, TH, TT.
      select:
      get HH: return(Kunle)
      get HT: return(Hilary)
      get TH: return(Michael)
      get TT: return(Sam)
      end algorithm

      Change this program into a recursive program that chooses fairly between the first three people. Hint: If Sam gets it, he can give it away.

      ii. What is the worst case number of coins that gets flipped? What is the probability that this happens.

      iii. If I get \$3 with probability p and \$7 with probability $1-p$, then my Expected[profit] is $3 \times p + 7 \times (1-p)$. Give a recurrence relation involving Expected[number of coin flips]. Solve this.

      iv. Suppose the program is

      algorithm *Choose*3()

      ⟨***pre−cond***⟩**:** There are no inputs, but we do flip coins.

      ⟨***post−cond***⟩**:** We output Kunle, Hilary, or Michael with equal probability

      begin
      flip a coin twice getting one of HH, HT, TH, TT.
      if(get HH) return(Kunle)
      else if(get TT)
      flip a coin twice getting one of HH, HT, TH, TT.
      if(get HH) return(Kunle)
      ... nothing else for Kunle
      end algorithm

      then Kunle is returned with the sequences HH and TTHH. The probability of this is $\frac{1}{4}+\frac{1}{4}\times\frac{1}{4} = \frac{1}{4}+\frac{1}{16} = \frac{5}{16}$. List the sequences of coin flips for which Kunle is returned in your program. What is the total probability of this? It should be $\frac{1}{3}$.

      v. The last calculation was a little tricky. Make it easier by giving a recurrence relation involving Pr[Kunle].

(b) The second algorithm will be iterative and will be called the "binary search" algorithm. This algorithm needs to work even more generally. The input to it needs to be a tuple $\langle p_1, p_2, \ldots, p_r \rangle$ of arbitrary probabilities such that $\sum_{i=1}^{r} p_i = 1$. Note r can be any positive integer. The algorithm can only flip fair coins. The output is the number $i \in [1, r]$ with probability p_i. As a hint, consider the line interval from 0 to 1. For each $j \in [0, r]$ put a mark at $\sum_{i=1}^{j} p_i$ in the line. This break the line into r pieces, the i^{th} of which has length p_i and is labeled with i. The algorithm will informally at random pick a point on the line and return the label i of the piece containing the chosen point. However, the algorithm can only flip a fair coin. Therefore, it does "binary search" to "find" the randomly chosen point. Like binary search, this algorithm could either be iterative or recursive. Give a sketch of the main ideas for this algorithm. You don't need to give all implementation details. Being an iterative algorithm, give all the loop invariant steps.

As a big hint, I will give you the loop invariant: Intuitively, we know that the point in the line we are looking for in the interval $[a, b]$. More formally, we know that this interval $[a, b]$ was chosen sufficiently randomly, so that if the algorithm would now choose uniformly at random pick a point within $[a, b]$, then the point obtained would have been uniformly at random picked from $[0, 1]$.

i. How is the loop invariant established?
ii. How is the loop invariant maintained?
iii. How does the loop invariant and the exit condition give the post condition, namely that i is returned with probability p_i?
iv. What is the measure of progress?
v. Suppose for example, $p_1 = \frac{1}{3}$ and $p_2 = \frac{2}{3}$. Is the algorithm guaranteed to stop? What is the expected number of coin flips?
vi. Given a general input $\langle p_1, p_2, \ldots, p_r \rangle$, what is the probability that it stops at or before t iterations? How does this probability (in the worst case) depend on r and on the p_i?

2. Consider a line (side walk with squares) with a wall at $i = 0$ and a wall at $i = n$. Each time step, the drunk man is standing at some square i and with probability $\frac{1}{2}$ stumbles one square forward and with probability $\frac{1}{2}$ stumbles one square backwards. Let p_i denote the probability that when starting at square i, he reaches $i = n$ before reaching $i = 0$. Set up a recurrence relation for the p_i. Solve it determining their values.

3. Choose three points A, B, and C uniformly at random within the unit circle. Let α be a random variable giving the angle ABC. Determine $\text{Exp}[\alpha]$. Hint: For any triangle (in the plane), the sum of its angles is 180. It is not hard. Can you as easily compute $\text{Exp}[\alpha^2]$?

4. James Bond 007: Consider the following game (similar to rocks-paper-scissors). Each round, each of the players must decide to *shoot*, *load*, or *shield*. Simultaneously, they reveal what action they took. When deciding what move to make in the current round, a player knows what has occurred in all previous rounds, but does not know what his opponent will do this round. If one player is shooting while the other is loading, then he gets a point. If they are both shooting at the same time, then they both get a point. The main challenge is that you cant shoot unless you have loaded since you have last shot.

Random(q_s, q_l): Consider the following opponent. Each round, after he has loaded and before he shoots, he shoots with probability $\frac{1}{q_s}$ and shields with probability $1 - \frac{1}{q_s}$. Each round, after he has shot and before he loads, he loads with probability $\frac{1}{q_l}$ and shields with probability $1 - \frac{1}{q_l}$. Alternatively, after he loads, he picks a random number $q \in [1, 2q_s - 1]$. Then he shields $q-1$ times and then shoots. Similarly, After he shoots, he picks a random number $q \in [1, 2q_l - 1]$. Then he shields $q-1$ times and then loads. With either strategy, the expected time from when he loads until he shoots is q_s and the expected time from when he shoots until he loads is q_l.

Probability Claim: For each round i (except near the beginning), the probability that Random(q_s, q_l) is shooting at time i is approximately $\frac{1}{q_s + q_l}$ and so is the probability that he is loading.

Proof: Suppose there are n rounds. The expected time from when he loads until he loads again is $q_s + q_l$. Hence, the expected number of times he loads is $\frac{n}{q_s+q_l}$ and so is the expected number of times he shoots. When these events occur is shifted earlier and later so that for any given i, it is random whether he is loaded or not (except near the beginning). The claim follows.

Tie Claim: Random(q_s, q_l) is expected to tie against any strategy.
Proof: Suppose I am shooting. The probability that Random(q_s, q_l) is loading is $\frac{1}{q_s+q_l}$ and the probability that he is shooting is $\frac{1}{q_s+q_l}$. Hence, the number of points I expect to get is $\frac{2}{q_s+q_l}$ and the number of points he expects to get is $\frac{1}{q_s+q_l}$. Suppose I am loading. The probability that he is shooting is $\frac{1}{q_s+q_l}$. Hence, the number of points he expects to get is $\frac{1}{q_s+q_l}$. I shoot and load the same number of times. Hence, we both to get $\frac{2}{q_s+q_l}$ points times the number of times I load/shoot.

Beating Random(q_s, q_l): Is the Tie Claim, actually true? If not, what false assumption is being made and when playing against Random(q_s, q_l) what strategy do you use to maximize the amount you win by. After a total of n rounds, what is the expected score?

Cheating: Name a strategy that might be considered to be cheating. If your proposed strategy from the last question does this then what is your best fair strategy against Random(q_s, q_l)?

Worst Opponent: Suppose that your opponent fixes his strategy to be what ever you proposed in the question for beating Random(q_s, q_l). Supposing that you know this, what strategy should you use to beat him.

Winning Strategy: If I know the opponent's strategy (or can figure it out), than I can win. Can you give a strategy that wins against every opponent?

5. Letter from a friend.
Jeff, I'm guessing you won't even remember this conversation, but it was about information theory, I guess is how you'd classify it, and it's been bugging me for... I guess it would be years now, actually.

Briefly, here's the question:

It seems to me that an attribute shared by two entities represents greater relatedness between those entities in inverse proportion to how common the attribute is among the members of the entity class as a whole. (Does that sound suitably abstract?) :0)

Here's an example of what I mean: if the great majority of people like the movie Up, then when I and some other person both hated it (I actually thought it was pure shit) doesn't that represent a greater similarity between me and this theoretical other person than does the fact that we both dislike some universally unpopular film - say, The Matrix Reloaded, for example? Conversely, if we both LOVED The Matrix Reloaded, wouldn't that mean more than if we both loved, uh... The Incredible? (one of my favorite movies ever, by the way).

(a) Use probability and conditional probability to answer his question.

(b) Use entropy to answer his question.