Chapter 8 Exercises

Brady Field

16 June 2016

Partners in Crime

Thom - helped with probabilities Matthew - helped with recurrence relation and probabilities

Typos

Exercise 9 on page 292 - "efficiciencies" should be "efficiencies" Exercise 12 on page 292 - "a games" should be "a game"

Solution to Exercise 9 on page 292

I would use Pascal's Triangle to get around the multiplications. I can generate the triangle up to the nth row and simply select the kth item as my coefficient.

The time efficiency is much better than the formula method since addition is much cheaper than the multiplication, or factorials, required in the formula.

At first glance, it may seem that the space efficiency is bad since we have to store Pascal's Triangle, but if we consider than we only ever need to store two rows of the triangle (for computing to the next row from the previous) and only half each row (since the other half is mirrored), the memory cost isn't that bad. The cost will only be seen for large values of n and is well worth the price for the incredible gain in time efficiency in many cases.

Solution to Exercise 12 on page 292

a. P(i,j) = p * P(i - 1,j) + (1 - p) * P(i,j - 1), the first recurrence is the probability of Team A winning the game. This leaves one less game for them to win and Team B still needs the same amount. The second recurrence is the opposite where Team A loses and has to win the same amount of games, but Team B has one less game to win. The initial conditions are P(0,j) = 1 since Team A has 0 games left to win, and P(i,0) = 0 since Team A cannot win when B has already won.

b. Team A needs to win at least 4 games to get the series. Using the recurrence relation, we get a final probability of .29.

c. C++

```
float P(float p, int i, int j) { if (i == 0) return 1; else if (j == 0)
    return 0;
return p * P(p, i - 1, j) + (1 - p) * P(p, i, j - 1); }
```

Solution to Exercise 8 on page 297 (NOTE wording change: Give two reasons versus Explain)

Is expensive with memory compared to a top-down algorithm.

The speed gain achieved with a memory function, in this case, is less than the speed gains with more elegant algorithms. Using Pascal's Triangle would be much faster and more space efficient.

Solution to Exercise 5 on page 303

False, we provided a counter example in class demonstrating this.

Solution to Exercise 7 on page 312

$$\begin{bmatrix} 0 & 2 & 3 & 1 & 4 \\ 6 & 0 & 3 & 2 & 5 \\ 10 & 12 & 0 & 4 & 7 \\ 6 & 8 & 2 & 0 & 3 \\ 3 & 5 & 6 & 4 & 0 \end{bmatrix}$$

Same as the extra credit quiz done in class