CISC/CMPE-365
Fall 2019

Week 6 Lab Problem: The Stopping Problem

Not to be handed in for grading

Suppose you are working in a chocolate factory – where the pay is in chocolate (my personal idea of the perfect job). Each day the factory produces exactly n boxes of chocolate, each with a different number of chocolates. The boxes pass down a conveyor belt past your work-station. As each box passes, you can see the number of chocolates it contains. Once per day you are permitted to pick up the box currently in front of you and keep it as your pay. Once a box goes past you on the conveyor belt it is immediately loaded on a delivery truck; you have lost your chance to choose that box.

Like any normal person you want to get as much chocolate as you can – so your goal is to pick the box with the most chocolates in it. The problem is that there is no upper limit on the number of chocolates in a box so you can never tell if you are looking at the best box of the day.

What algorithm should you use to make your decision?

A popular approach is to adopt a strategy that may seem counter-intuitive:

Choose a value k, where $0 \leq k < n$
Let the first k boxes go by, regardless of how good they look. Remember the best box you have seen so far – call this number of chocolates best_num
As the next n-k-1 boxes go by, choose the first box you see that contains more than best_num chocolates
If Box_n is going by and you have not yet chosen a box, choose Box_n

Obviously this strategy cannot guarantee that you get the best box: that box might be among the first k. Or box k+1 might be better than the first k boxes (so you take it) even though box k+1 might not be the best overall.

(PS: If you don't like chocolate, you have my sincere sympathy. Substitute "that thing you love".)

The value of k is crucial. If k = 0, you always take the first box. If k=n-1 you always take the last box. In both of these cases your probability of getting the best box is simply $\frac{1}{n}$.

But if k=1, you will get the best box whenever Box_1 is better than all the boxes that come between it and the best box. (Example: suppose the best box has 300 chocolates. If the sequence is 183, 96, 37, 120, 300, ... You will let Box_1 go by and remember 183. Then you ignore the next three boxes because they are all < 183. You take the 300 because it is the first box you see that is > 183. So in this case you get the best box. But if the sequence is 183, 96, 37, 184, 300 ... then you don't get the best box because you choose 184.)

It's non-trivial to work out the probability of getting the best box when k = 1 , but it can be done ... and it's greater than $\frac{1}{n}$!! So letting Box_1 go by and then taking the first box that beats Box_1 is a better strategy than simply taking the first box.

So the obvious question is – what is the optimal value of k? How many boxes should we let go by in order to maximize the probability that the box we eventually choose is actually the best? The wonderful thing is that somebody has worked this out – and the answer is $\frac{n}{e}$

Yes, that e – the natural logarithmic base, first named by Leonhard Euler.

$$\frac{n}{e} \approx 0.37 * n$$

Example: if n=500, we should let the first 185 boxes go by, then choose the first one that is better than all the boxes we have seen.

And just how good is this algorithm with $k = \frac{n}{e}$? It turns out that the probability of choosing the best box is at least $\frac{1}{e} \approx 0.37$ ... and this is true no matter how large n is!

It may feel counter-intuitive that we maximize our probability of getting the best result by automatically rejecting more than a third of the choices – but the math is sound.

Or is it? In this lab you will gather experimental evidence that will either confirm or refute this analysis. You will experiment with different set sizes and with different values of k.

Part 1:

Implement the WW algorithm, which I have named after the most important Chocolate Factory owner of all time. (It is more properly called the Odds Algorithm, and as such was formulated by Bruss in 2000.)

Here is some pseudo-code:

```
WW(n, Box[], k):
        # n is the number of chocolate boxes
        # Box[] is a 1-dimensional array.  Box[i] is the number of chocolates in the i^th box.
        # My chocolate boxes are numbered from 1 to n.  If you insist, yours can be
        # numbered from 0 to n-1.
        # k is the number of boxes that will be skipped

        best_number = 0
        for i = 1 to k:
                if Box[i] > best_number:
                        best_number = Box[i]
        for i = 1 to n-k-1:
                if Box[i] > best_number:
                        return i
        return n
```

Test your implementation on a small set of values Test with the largest value in different locations, and with different values of k. Part of your task here is to select test data.

Part 2:

Conduct the following experiment:

```
for n = 100, 200, 400, 800, 1600:  #you may wish to test for a wider range of values of n

        let count_wins be a 1-dimensional array indexed from 35 to 39, all set to 0
        # you may wish to test for a wider range of values of k_percent

        for tests = 1 to 500:
                let Box be a randomly ordered 1-dimensional array of n randomly chosen
                        distinct integers from the range [1 ...  1000000]
                let x be the position in Box of the largest value (i.e. Box[x] is the largest
                        value in Box)
                for k_percent = 35 to 39:
                        p = WW(n,Box, k_percent*n/100)
                        if p == x:                  # the algorithm found the best box
                                count_wins[k_percent] += 1

        print n
        for k_percent = 35 to 39:
                print k_percent, count_wins[k_percent], count_wins[k_percent]/500
```

Part 3:

Question 1:

Do your experimental results support the theoretical result that k = 0.37*n is the value of k that gives the greatest probability of choosing the best box?

Question 2:

Do your experimental results support the theoretical result that with k = 0.37*n, the probability of choosing the best box is at least $\frac{1}{e}$?

Reminder: This lab will NOT be graded.

Part 4: (completely optional!)

If you find this problem interesting, here are some further readings:

Knowing When to Stop: https://www.americanscientist.org/article/knowing-when-to-stop

The Stopping Problem: https://en.wikipedia.org/wiki/Optimal_stopping

Optimal Stopping and Applications: https://www.math.ucla.edu/~tom/Stopping/sr1.pdf

The Best Choice Problem: https://en.wikipedia.org/wiki/Secretary_problem

The Odds-Algorithm: https://en.wikipedia.org/wiki/Odds_algorithm

Part 5: (even more optional!)

After working for a few years at the chocolate factory you realize that your obsession with choosing the best box every day is causing you a lot of stress. You decide to relax a bit and count each day as successful if you choose a box that is in the top 10% of all boxes produced that day.

Conduct experiments to determine the value of k that maximizes the probability that your chosen box will be in the top 10% for that day. Is k consistent for all values of n?