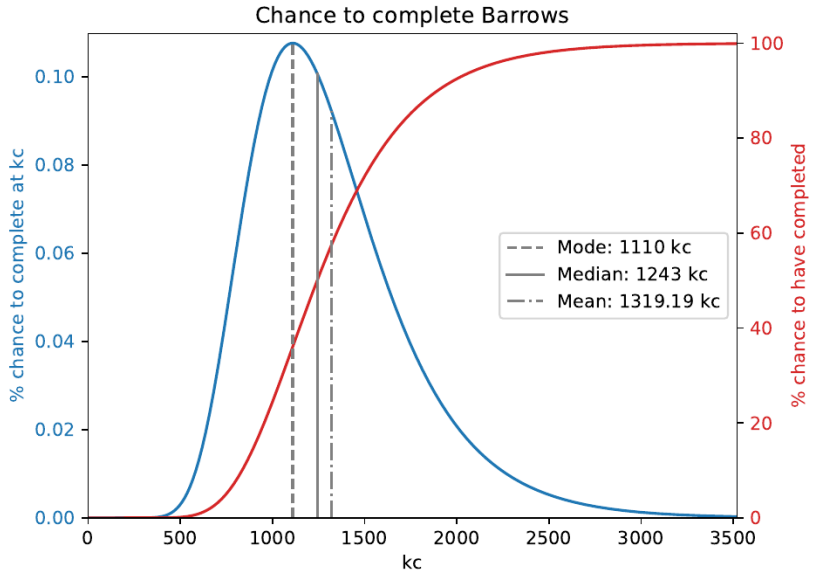
Calculating completion rates for all bosses in OSRS

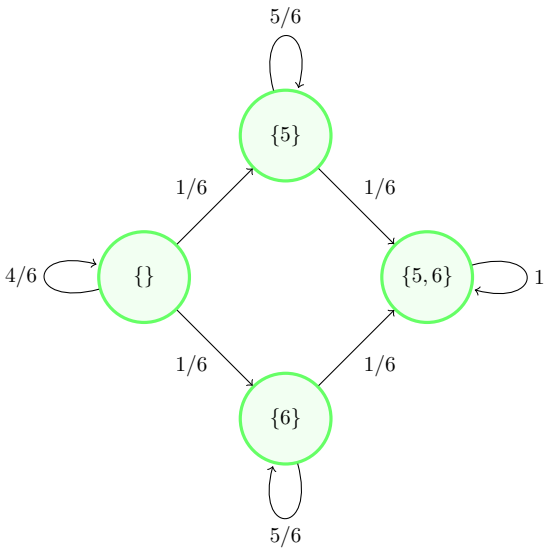


Introduction

The time it takes to complete a collection of different items is something know as the coupons collectors problem. This problem is very hard to solve in the general case. For us it gets even more complex when we introduce multiple drop tables and other weird behaviour. Which is why we convert the problem into an *Absorbing Markov chain*. They can produce the same or more results and are far easier to generalize. I've written some code to do this it can be found here

Die roll example

Imagine a six sided die, where we need to land both a 5 and a 6 before we've collected all 'items'. There's 4 possible states we can find ourselves in depending on whether we've rolled a 5 or 6 previously. We can model these states as a directed graph where the edges are the odds of transitioning to the state it's pointing to. This is known as a markov chain.



Using this graph we can rephrase the problem as follows; *How many rolls does it take to end up in the absorbing state* $\{5, 6\}$. We can solve this problem by creating the transition matrix P of the graph. Where every p_{ij} in the matrix is the probability of moving from state i to state j in the graph after 1 roll of the die. If we raise the matrix to a nth power we get the transition matrix given p_{ij}^n or the odds of moving from state i to j after n die rolls. We want to know the odds of moving from s_0 where we have no 5 or 6 to s_3 where we have both. The odds of this happening is given by $p_{0,3}^n$ (the upper right number in each matrix).

$$P = \begin{bmatrix} \frac{4}{6} & \frac{1}{6} & \frac{1}{6} & 0 \\ 0 & \frac{5}{6} & 0 & \frac{1}{6} \\ 0 & 0 & \frac{5}{6} & \frac{1}{6} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad P^2 = \begin{bmatrix} \frac{4}{9} & \frac{1}{4} & \frac{1}{4} & \frac{1}{18} \\ 0 & \frac{25}{36} & 0 & \frac{11}{36} \\ 0 & 0 & \frac{25}{36} & \frac{11}{36} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad P^{10} = \begin{bmatrix} \frac{1024}{59049} & \frac{968561}{6718464} & \frac{968561}{6718464} & \frac{20991751}{30233088} \\ 0 & \frac{9765625}{60466176} & 0 & \frac{50700551}{60466176} \\ 0 & 0 & \frac{9765625}{60466176} & \frac{50700551}{60466176} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In these matrixes we can see that after 2 rolls we have a 1/18 odds of having rolled a 5 and a 6, and after 10 rolls it's about 2/3. If we plot the completion odds against n the power of the matrix we get a cdf plot as shown for barrows on page 1. We can easily extract the mode and median from it. And the mean gets increasingly more accurate the greater the more iterations we do of matrix multiplication. This slight inaccuracy can be seen in some of my results where we're supposed to only get a single drop like callisto, but the mean is 0.01 off.

Applying this to OSRS bosses

To make a transition matrix for a boss we need to have a state for every possible combination of having or not having the items we collect. We do this representing the index of a state i as a binary number where each bit with index j signifies whether or not we have the jth item of the list of items we're collecting. Eg: for the first state $s_5 \to i = 5 = 1001_b$ signifies we have the first and fourth item. This lets us fill in the matrix by iterating over every state and every drop as follows: $p_{i,i+2^j-1} = D(j)$ where $D(j)$ is the droprate of the jth item.

When we need to collect items j and k in order, like for example the ring pieces from the Alchemical Hydra, we make sure that the odds of getting item k is 0 before we have item j.

Multiple drop tables can accounted for by creating a matrix for each table and multiplying the resulting matrices together, as long as we make sure that the indexes on every matrix represent the same items. This is needed to model rates for Bosses like Zulrah or GWD accurately because you can get multiple items in 1 kc.

Barrows

The previous approach used for all other bosses doesn't work for barrows, because there are just too many drops. We'd need 2^{24} states to model barrows. So for it we need to adjust the way we create the matrix. We can use the fact that every unique drop from barrows has the same droprate. Using it we can reduce the number of states to 25. Where a state s_i signifies that we've collected i different uniques from barrows. We can then calculate the odds of moving from s_i to s_{i+a} as below and fill in our matrix.

$$p_{i,i+a} = \sum_{k=a}^{7} \left(\binom{7}{k} \left(\frac{1}{102}\right)^k \left(\frac{101}{102}\right)^{7-k} \binom{k}{a} \prod_{l=1}^{a} \left(\frac{24-i-l}{24-l}\right) \prod_{m=1}^{k-a} \left(1 - \frac{24-a-i-m}{24-a-m}\right) \right)$$

Where i is the amount of items we already have and a the amount of items we already have. This calculations works by summing the odds of getting 1 through 7 new uniques from a single chest. The first half calculates the odds of getting k uniques, which we then multiply by the second part which calculates the odds of a of those k uniques being new. We need to account for the fact that each of the 7 rolls in the chest has a 1/102 chance of being a unique but those are never the same. Which is why the number of uniques in a chest affects the odds of them being new uniques. Eg: if you have no items yet and get 2 uniques they are guaranteed to both be new only if you got them from the same chest.