Journal of Geek Studies

ISSN 2359-3024

The Infinite Fish Playing Pokémon Theorem

João V. Tomotani
Universidade de São Paulo; São Paulo, Brazil.
Email: t.jvitor@gmail.com

THE INFINITE MONKEY THEOREM

I will start by presenting this little famous theorem. The Infinite Monkey Theorem states that if you have a monkey hitting keys at random on a typewriter keyboard for an infinite amount of time, eventually it will end up reproducing some famous text, like the complete works of William Shakespeare, The Lord of the Rings, or the very article you are currently reading (the process used to write this article is not that far from it, actually).

This is the sort of theorem that clearly became famous because of its funny name. And, while I do find the name amusing, the theorem itself is also quite interesting; well, at least more interesting than the real-life experiment that the University of Plymouth decided to do, which remarkably showed that a monkey with infinite time would probably be able to defecate infinitely and destroy an infinite amount of typewriters (BBC News, 2003). Also, for some reason, friends of mine tend to state the Infinite Monkey Theorem as having an infinite amount of monkeys with an infinite amount of typewriters, instead of the infinite time stuff. Though this would solve the small problem of having to find an immortal monkey, maybe an infinite amount of monkeys would also pose a problem (for more information, you can read "what would happen if you were to gather a mole ($6x10^{23}$) of moles" from Randall Munroe, 2014) – let me simply say that this would make Planet of the Apes a lot more literal.

In any case, recently a small event reminded me of the Infinite Monkey Theorem. I am talking, of course, about Grayson Hopper, and his quest to become a pokémon master. Grayson is a fish who rose to absolute stardom after his owners decided to make him play Pokémon. By swimming in his aquarium, Grayson's position is detected by a camera and a command is sent to the game. You can follow his play on Twitch. Not much is going on right now (or ever).

Grayson is trying to prove by himself that, through a generator of random movements and a lot of time, one should be able to finish a game like Pokémon, which you can play by pressing only one button at a time and not having to rely on timing or stuff like that. I mean, if a ten year old boy can do it, why can't a fish?

And what the Infinite Monkey Theorem states is that he can do it, right? The fish might very well be able to beat the game in his lifetime, since a really large number of combinations of buttons will be generated, and one of them MUST be the correct one. Well, meet Route 22 (Fig. 1).

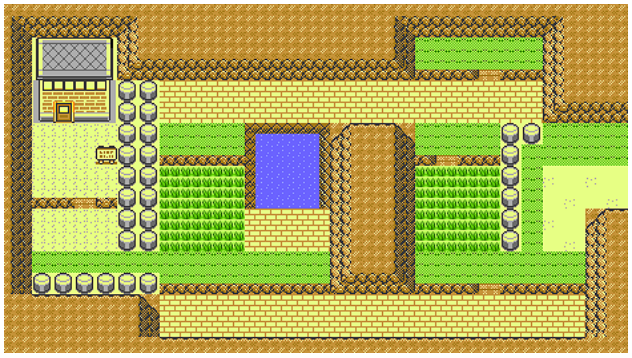

Figure 1. Why, hello there, Route 22. Image taken from: Pokémon Gold/Silver/Crystal.

From now on, I will consider that all of you had a childhood and are thus capable of following the game mechanics.

Route 22 could be called the nightmare of random walking (for more information on random walking, or "why a drunk always come back home while a drunk bird may be lost forever because of extra dimensions and stuff", see Math Explorer´s Club, 2009). The game mechanics of one-side-crossing-only ledges clearly makes this route goddamn awful. Also, Grayson tends to stay still for some seconds on the same area of the aquarium, thus repeating the same command a lot (a "down" command one time more than necessary can be fatal here). So, is it impossible?

Only one way to find out…

THE INFINITE FISH PLAYING POKÉMON SIMULATION MODEL

To simplify, let's say that the movement of the fish in the aquarium is a random process that can be categorized as a Markov chain. A Markov chain is one of the simplest stochastic processes, where the next entry in the chain depends only on the current position (or state), and not on the history of entries (this lack of history is called the Markov property). The random walk is an example of a Markov chain. Though the movement of the fish depends on the history (usually he will keep swimming on the same direction), let's not waste time trying to model this further, because time is precious and we wouldn't like to waste it on useless stuff like a useless model.

Grayson's aquarium is divided into nine squares (Fig. 2). Each square has a specific command, with one exception, the "randomize!" command, which randomly chooses one of the other eight. Let's consider that one command is chosen after every second.

Thus, the Markov chain describes a process where the random walker is on a position among nine, and has a random probability of going to any of the nine positions (including staying on the same one) after one second.

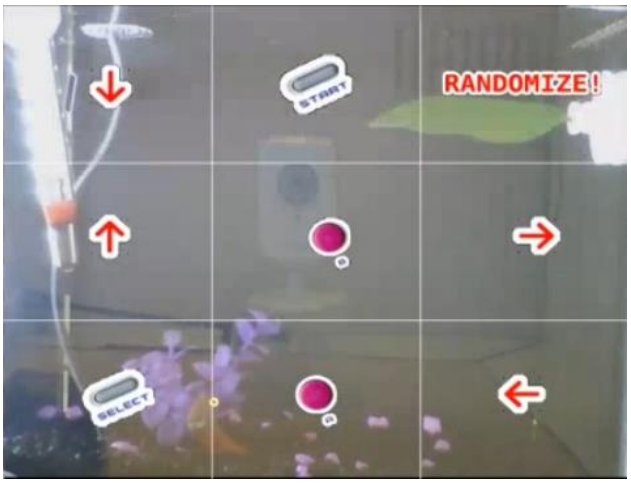

Figure 2. Grayson's aquarium and control scheme. Screenshot taken from: Twitch – Fish Plays Pokémon.

The matrix that gives the probability of transition to each of the states in a Markov chain is called the transition matrix. The transition matrix can be dynamic or stationary in time. For example, giving the fish more food would make it dynamic, with a higher probability of him going to the surface of the aquarium. But let's consider a stationary matrix for simplicity.

After watching Grayson playing Pokémon for 2 minutes (boy, that was fun!), I generated some numbers for the transition matrix (Table 1). I decided that the probability of him staying still in the same position for one second would be 80% in any of the positions (though I must say that it is probably much higher). Also, I considered that the probability of him going to an adjacent area was higher than crossing the aquarium from one instant to the other. Since the position of the commands changes over time on the twitch play, I decided to generate them randomly and leave them as such. Admittedly, the fish spends a lot of time near the surface, but let's not focus on the details since the randomization of the commands also contributes to the randomization of the process.

Table 1. Position matrix.

1	2	3
4	5	6
7	8	9

Table 2. Probabilities of the transition matrix.

	1	2	3	4	5	6	7	8	9	
1	80.0%	5.0%	2.0%	5.0%	3.5%	1.0%	2.0%	1.0%	0.5%	100.0%
2	4.0%	80.0%	4.0%	2.0%	4.0%	2.0%	1.0%	2.0%	1.0%	100.0%
3	2.0%	5.0%	80.0%	1.0%	3.5%	5.0%	0.5%	1.0%	2.0%	100.0%
4	4.0%	2.0%	1.0%	80.0%	4.0%	2.0%	4.0%	2.0%	1.0%	100.0%
5	2.0%	3.0%	2.0%	3.0%	80.0%	3.0%	2.0%	3.0%	2.0%	100.0%
6	1.0%	2.0%	4.0%	2.0%	4.0%	80.0%	1.0%	2.0%	4.0%	100.0%
7	2.0%	1.0%	0.5%	5.0%	3.5%	1.0%	80.0%	5.0%	2.0%	100.0%
8	1.0%	2.0%	1.0%	2.0%	4.0%	2.0%	4.0%	80.0%	4.0%	100.0%
9	0.5%	1.0%	2.0%	1.0%	3.5%	5.0%	2.0%	5.0%	80.0%	100.0%

By multiplying the transition matrix by itself a number "n" of times, you can find the accumulated transition probabilities after n events. For instance, if I multiply the matrix by itself twice, I will have the probabilities of Grayson transiting between two states after two seconds.

Since the transition matrix is stationary, aperiodic (no time limitations from transiting from one state to the other) and has no recurrent state (states with probability of 0, transition state, or 1, absorbing state), it is also said to be ergodic, that is, the system has the same average behavior over a long period of time. Multiplying the matrix by itself 128 times, I get to the following matrix (Table 3).

That is, the probability of Grayson being in one of the states after a long period of time is not dependent on his initial state, only on the transition matrix. Looking at this matrix, I can see that the probability of the fish spending some time on the center of the aquarium is slightly higher than on the edges, given the numbers I decided for the transition matrix. Since the distribution among the states is quite similar, I am happy with the transition matrix chosen. After modeling the fish movement, it was time to model the map of Route 22 (Fig. 3).

Table 3. Transition matrix after 128 seconds.

	1	2	3	4	5	6	7	8	9	
1	9.5%	11.5%	9.5%	11.5%	15.9%	11.5%	9.5%	11.5%	9.5%	100.0%
2	9.5%	11.5%	9.5%	11.5%	15.9%	11.5%	9.5%	11.5%	9.5%	100.0%
3	9.5%	11.5%	9.5%	11.5%	15.9%	11.5%	9.5%	11.5%	9.5%	100.0%
4	9.5%	11.5%	9.5%	11.5%	15.9%	11.5%	9.5%	11.5%	9.5%	100.0%
5	9.5%	11.5%	9.5%	11.5%	15.9%	11.5%	9.5%	11.5%	9.5%	100.0%
6	9.5%	11.5%	9.5%	11.5%	15.9%	11.5%	9.5%	11.5%	9.5%	100.0%
7	9.5%	11.5%	9.5%	11.5%	15.9%	11.5%	9.5%	11.5%	9.5%	100.0%
8	9.5%	11.5%	9.5%	11.5%	15.9%	11.5%	9.5%	11.5%	9.5%	100.0%
9	9.5%	11.5%	9.5%	11.5%	15.9%	11.5%	9.5%	11.5%	9.5%	100.0%

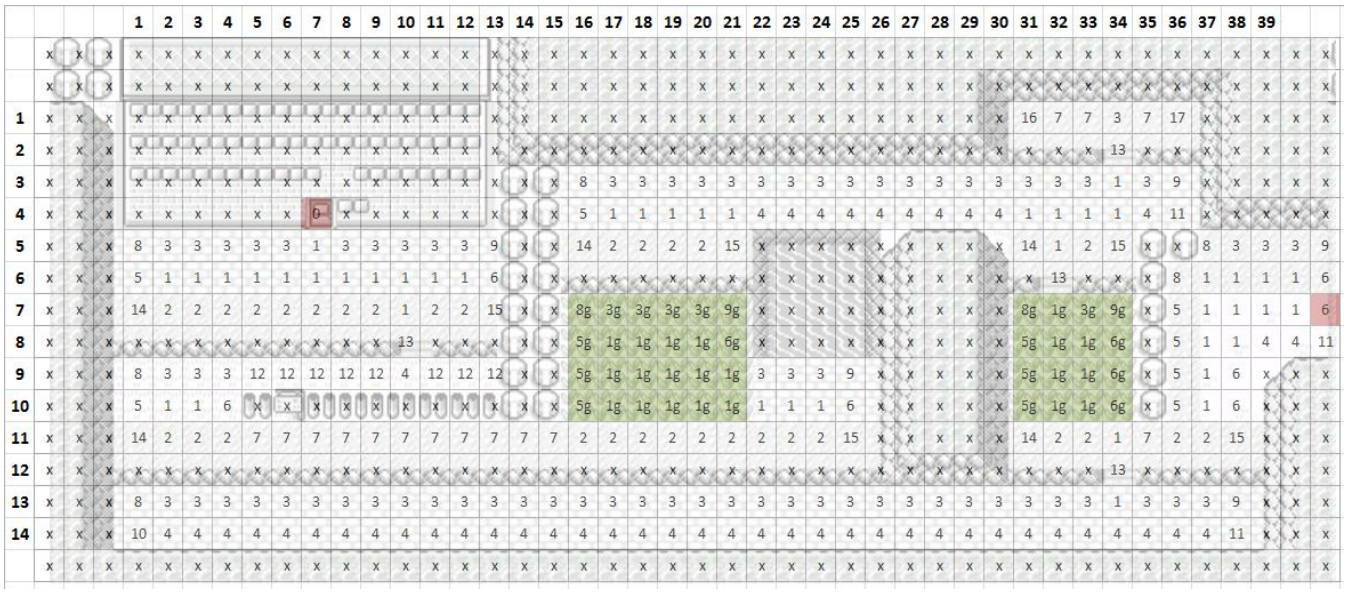

Figure 3. Modelled map of Route 22.

As can be seen on Figure 3, for each coordinate on the map, a number was attributed. This number states movement restrictions or other characteristics. Coordinates with the number “1” have no movement restriction, while coordinates with the number “3” has restrictions with the “up” direction, and coordinates with the number "2" have no restrictions, but the "down" direction skips a square. Once again, to simplify the model I considered that Grayson cannot go to the right of the map, leaving the area of Route 22.

The green area is the “wild grass” area. I decided to count the steps taken on “wild grass” areas to estimate how many pokémon battles Grayson would face in his epic quest. The probability encounter formula, according to Bulbapedia (2014) is defined as P = x / 187.5, where x is the encounter rarity variable. Considering a common encounter rate (x = 8.5), Grayson should face S*0.05 battles, where S is the number of steps taken on wild grass.

The desired results are the number of total commands necessary for Grayson to arrive at his

destination. Other interesting results are the number of battles and the number of commands given while in "paused" state.

As a limit of commands (thus, time), at first I thought that the simulation should go no longer than the average life span of a goldfish. Considering a life span of 25 years, this would be the same as 788,400,000 seconds/commands (if he keeps pressing the A or select button, or moving while on paused state, it still counts as a command for the simulation).

This number of commands, though, is unreasonably high and would probably only increase the entropy of the universe. So, I decided to just let it run for a couple of minutes and see what would happen after the equivalent of one day for the fish.

The model was written in VBA and the program ran on my personal computer.

RESULTS AND DISCUSSION

The results are as follows:

- Number of steps to complete: 86400 (1 full day, didn't complete, ended in coordinate 14 x 18);
- Number of steps on "wild grass" areas: 351;
- Estimated number of battles: approximately 18;
- Number of steps taken on paused state: 47796.

Thought I only simulated the equivalent of one "fish day", whenever I stopped to watch the simulation, it was clear that the character in-game was struggling to get out of the lower part of the map. It was expected, almost inevitable, for Grayson to spend most of his time randomly walking around the 76 coordinates that composed this artificial cage (with only one way out, coordinate 13 x 34, and many ways in); more than 99% of the time was spent there. Positioning the character right in front of the exit and moving "up" twice was a very specific command that happened only twice during the simulation, with the character going back right after. Also, more than half of the commands were given while on paused state.

A DIFFERENT APPROACH

The greatest limitations of the study above are the short simulation time and the fact that the simulation was run only once. Considering how the stochastic process of the fish swimming impacted the results, it would be necessary to run the simulation dozens of times to achieve better and more valid results.

With that in mind, I now propose a different take on The Infinite Fish Playing Pokémon experiment. Instead of simulating the random process of a fish swimming, I will propose some simplifications in the previous model, so that the problem can be solved on a deterministic way. Though the simplification will reduce considerably how well the model represents reality, the results presented will offer valuable information on the magnitude of the problem.

EXPLANATION OF THE NEW APPROACH

On my first attempt to model the Fish Playing Pokémon, I considered that the movement of the fish in the aquarium was a random process which could be represented as a Markov chain. Since the position of the fish in the aquarium was considered a Markov process, this stochastic process was what defined which command was sent to the Pokémon game.

Since the probability of each command being chosen depended on the position of the fish in the aquarium, the modeling of the Pokémon game was a complex process that needed simulation.

This time, I will simply ignore the position of the fish and consider that the command sent to the game is a completely random process, with each of the directional commands having the same probability of being chosen. This way, the movement made on the game depends only on the current position of the character in the map, that is, the game itself can now be modeled as a Markov chain. With this simplification, the probability of the character being in each position can be defined on a deterministic way.

IMPLEMENTATION

Once again, the first step is to attribute a number to each coordinate in the map, and define which the possible movements for them are. There are 305 different possible coordinates in Route 22 (Fig. 4). Considering four possible commands, each command has a 25% chance of being chosen.

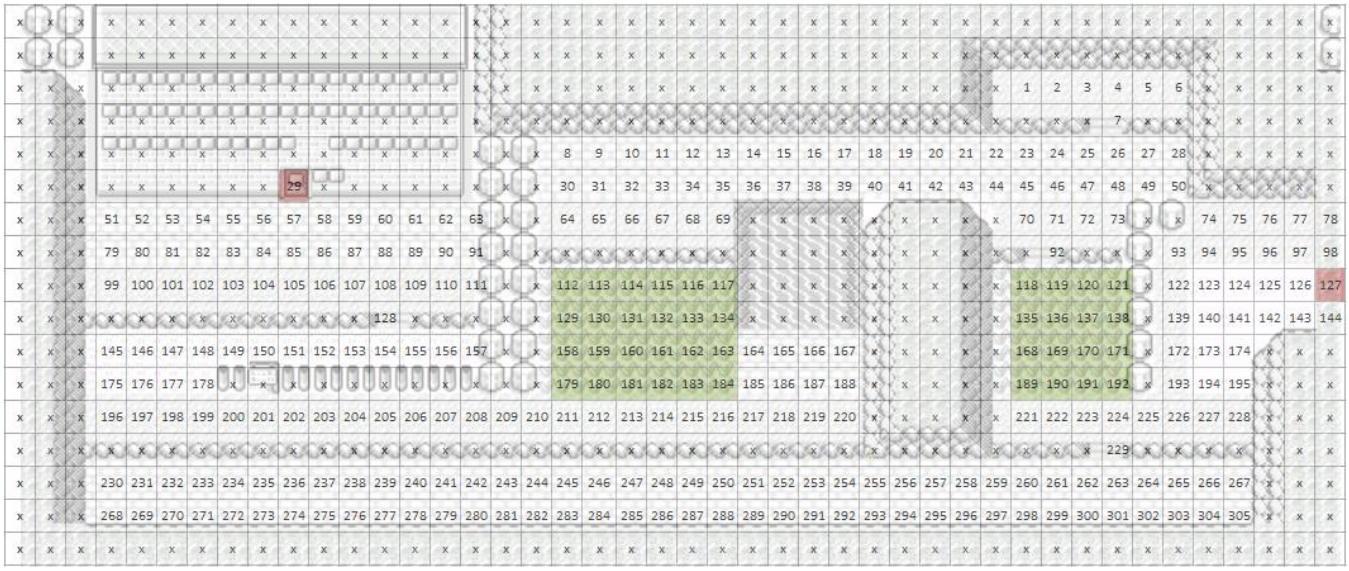

Figure 4. Route 22 with coordinates.

If the chosen command is impossible to be made, the character stays on the same coordinate. For example, if the character is on square 305, there is a 25% chance of him going to coordinate 267, 25% of going to 304, and 50% chance of staying in the same place. These probabilities, as stated, do not vary.

Having defined the coordinates and the possible movements, I can define the transition matrix. Since I have 305 possible positions, the transition matrix will be a 305 x 305 matrix (Figure 5). From each position, the character can do, at most, 4 different movements. That way, at least 301 movements will have a zero (for instance, the probability of going from position 127 to 29 in a single step is zero).

By multiplying the transition matrix by itself a number “n” of times, you can find the accumulated transition probabilities after n events.

Another simplification adopted is that the end of route 23 is not an absorbing coordinate, that is, if the character arrives at the end, it is possible for him to go back. This simplification is necessary for the transition matrix to be ergodic, and thus stabilize after a long period of time.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0.50	0.25																				
2	0.25	0.25	0.25																			
3		0.25	0.25	0.25																		
4			0.25	0.25	0.25		0.25															
5				0.25	0.25	0.25																
6					0.25	0.50																
7				0.25			0.50															
8								0.50	0.25													
9								0.25	0.25	0.25												
10									0.25	0.25	0.25											
11										0.25	0.25	0.25										
12											0.25	0.25	0.25									
13												0.25	0.25	0.25								
14													0.25	0.25	0.25							
15														0.25	0.25	0.25						
16															0.25	0.25	0.25					
17																0.25	0.25	0.25				
18																	0.25	0.25	0.25			
19																		0.25	0.25	0.25		
20																			0.25	0.25	0.25	
21																				0.25	0.25	0.25
22																					0.25	0.25
23																						0.25
24																						
25																						
26							0.25															

Figure 5. An example of a small subset of the transition matrix.

Multiplying the matrix by itself 64000 times (since there were many coordinates, it took a while for the probabilities to stabilize), I get to the matrix shown in Figure 6. This stabilized matrix shows that, after a long time, the probability of the character being in each coordinate does not depend on the starting point. This is true because of the ergodic property of the system. When stabilized, it is said that the system achieved stationary regime.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
2	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
3	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
4	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
5	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
6	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
7	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
8	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
9	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
10	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
11	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
12	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
13	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
14	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
15	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
16	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
17	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
18	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
19	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048
20	0.000003	0.000005	0.000013	0.000033	0.000013	0.000007	0.000074	0.000004	0.000005	0.000007	0.000010	0.000015	0.000023	0.000035	0.000048

Figure 6. A small part of the transition matrix, after 64000 commands.

RESULTS OF THE DIFFERENT APPROACH

Another way of interpreting the stationary regime matrix is that, instead of the probability of the character being in each state, it represents the percentage of time the character will spend in each of the 305 coordinates. That way, it is possible to calculate how many iterations are necessary, on average (after achieving stationary regime), for the character to pass through a defined coordinate.

The probability of the character being in coordinate 29, thus, is $4.11*10^{-13}$, that is, a little

more than 0.000000000041%. If we divide 1 by this probability, we see that, on average, we would need $2.43*10^{12}$ iterative steps to pass through coordinate 29.

Since each command (or step) is made every 1.5 seconds, this number of steps would take around 115.700 years to be made. This time should be at least doubled, if we considered the possibility of the commands that are not movement commands (the "A", "B" and "Pause" commands).

Interestingly, the percentage of time spent in the lower part of the map (coordinates 230 to 305) is 96.36%, and the time spent in the initial part of the map (up to coordinate 92) is 99.42%.

CONCLUSION

What are the odds of the universe taking form as it did? What are the odds of you being conceived? These questions were made many times by many different scientists from different areas. Binazir (2011) shows through an interesting chart, which went viral on the internet, that the odds of you existing are the same as two million people throwing a trillion-sided dice and all of them getting the same result. As usual, the best answers are from Douglas Adams (1979), who shows that the whale and the bowl of petunias are not impossible as you initially thought, just as impossible as anything else.

The more specific the event, the more impossibly low are the odds of it happening.

This study only started as a joke, but the more I thought about it, more it made sense in this world of impossible improbabilities happening.

Grayson will clearly never finish this game, not before the heat death of the universe. He probably won't even get close to the Safari Zone, from where he would never come out as well anyway. But maybe, only maybe, something impossible might happen, and the fish might be able to achieve his impossible dream of becoming a Pokémon master.

REFERENCES & FUTHER READING

For more information on Markov chains, queue theory and lots of Operations Research concepts, read: **Winston, W.L.** (2003) Operations Research: Applications and Algorithms, 4th ed. Cengage Learning, Boston.

Adams, D. (1979) The Hitchhiker's Guide to the Galaxy. Pan Books, London.

BBC News. (2003) No words to describe monkeys' play. Available from: http://news.bbc.co.uk/2/hi/3013959.stm (Date of access: 14/Aug/2014).

Binazir, A. (2011) What are the chances of your coming into being? Available from: http://blogs.law.harvard.edu/abinazir/2011/06/15/what-are-chances-you-would-be-born/ (Date of access: 14/Aug/2014).

Bulbapedia. (2014) Tall Grass. Available from: http://bulbapedia.bulbagarden.net/wiki/Tall_grass (Date of access: 14/Aug/2014).

Math Explorer's Club. (2009) Random walks. Available from: http://www.math.cornell.edu/~mec/Winter2009/Thompson/randomwalks.html (Date of access: 14/Aug/2014).

Munroe, R. (2014) What If?: Serious Scientific Answers to Absurd Hypothetical Questions. Houghton Mifflin Harcourt, Boston.

Twitch. (2014) Fish Plays Pokémon. Available from: www.twitch.tv/fishplayspokemon (Date of access: 14/Aug/2014).