MAS115: SEMESTER 1 MINI-PROJECT

160175114

1. Initial investigation

1.1. **The Python code.** The initial investigation involved creating a piece of code using Python that would simulate Annie the ant's movement a large number of times, count the number of times she ends on each vertex, and then print out the percentage of times she ends up on each vertex. In order to do this I wrote two programs. The first to check that Annie's journey was simulated correctly; the second to simulate the journey a large number of times and make the recordings. Both pieces of code are in most places the same so I'll explain their differences and then talk through the code and how it works.

For the first program I included various print commands in order that the process that took place between Annie's first and final step could be seen. I needed to check that the code met the following conditions:

- Annie must always start her journey on the vertex 0.
- The journey must last a duration of exactly 7 steps.
- In each step she must move to an adjacent vertex
- In each step she must move either clockwise or counter-clockwise with equal proabability.

In the second program I removed the print statements so that once the number of trials was increased there wouldn't be pages of steps to read through, but that only the final result would be presented. I then increased the number of trials so that the experiment would be repeated a great number of times in order that the perecentages taken would be more accurate.

So, in order to simulate Annie's journey I first imported the random module which I used to make sure Annie moved either clockwise or anticlockwise with equal probability. I did this by generating a random integer x such that $1 \leq x \leq 10$ and using *if* and *else* commands. If the integer was > 5 the ant would move clockwise, else it would move anti-clockwise. Since each integer is equally likely to be chosen and there are 5 integers both less than 5 and greater than 5, each direction is equally likely.

I used a list in order to simulate the pentagon, which I called *vertices.* Each vertex is an expression in the list. I decided to use integers so that the ant could move between them by adding or subtracting 1 from the index. I ran into a problem in the event where 1 was added 5 times in a row, and there isn't a 5th index in the list. I fixed this by adding another *if* command which changed the index back to 0 if it was 5. And likewise for subtracting, the index is changed to -1 when it becomes -6.

I used a for loop which repeated 7 times to simulate Annie's 7 steps, and this was within another for loop which repeated the journey a great number of times and each time recorded the ending position.

In order to record which vertex Annie landed on at the end of each 7 step journey and print out the percentage of times it had landed there, I created another list named *recordings*. At the end of each set of 7 steps I used the *.append*(x) method to add whichever vertex Annie was on to the end of the list. Then at the end of the simulation I used the *.count*(x) method to return the number of times each vertex occured in the list, and converted this number to a percentage, which was then printed.

1.2. **Findings.** After simulating the experiment and recording the result, I found it seemed that the percentage of the number of times Annie ends on vertex 1 and vertex 4 are equal, and likewise vertex 2 and vertex 3 share the same percentage. Annie lands most often on the vertices 1 and 4, followed by vertices 2 and 3 and lands least often on the starting vertex 0. I decided to calculate what the exact percentage would be for the chance Annie had of landing on each vertex after 7 steps. I did this by starting with the first step, in which Annie has a 50% chance of landing on either 1 or 4, on the second step, if she is on 1 she has a 50% chance of moving to either vertex 0 or 2, since Annie is already on vertex 1 50% of the time, she will then move to 0 and 2 25% of the time. Likewise she will move to vertices 0 and 3 25% of the time from vertex 4. So in total after two steps Annie lands on vertex 0 50% of the time, vertices 2 and 3 25% of the time and vertices 1 and 4 0% of the time. After 3 steps we have 0% on 0, 37% on 1 and 4 and 12.5% on 2 and 3. Here we begin to see the relationship between vertices 1 and 4 and vertices 2 and 3. Carrying on in the same way I found that after 7 steps she lands on vertex 0 10.9375% of the time, 27.34375% for 1 and 4, and 17.1875% for 2 and 3. The percentages that are returned from the Python code are consistently very close to these values.

Another interesting observation I found after working out the percentages is that the percentage of times Annie ends on vertex 0 is always the same as the percentage for 1 and 4 in the previous step. This can be explained in the following way: The only connections to vertex 0 are 1 and 4 and I assume that vertices 1 and 4 always have an equal percentage. Each of these vertices is passing on half of it's percentage to vertex 0 during each step. So vertex 0 is recieving two lots of half of the percentage 1 and 4 had the previous step, therefore vertex 0 always has the same percentage as that of vertices 1 and 4 the previous step.

2. Number of steps variation

2.1. **The code.** For the first variation I experimented with the number of steps that Annie would take. This involved very minimal changes to the code, simply changing the number in the range command in the for loop seen on line 104.

2.2. **Findings.** By changing the number of steps I found, for lower numbers, an odd number of steps meant that vertex 0 had the highest percentage, followed by vertices 1 and 4, and vertices 2 and 3 had the lowest percentage. For an even number of steps this was reversed, vertices 2 and 3 had the highest percentage, followed by vertices 1 and 4, and vertex 0 had the lowest percentage. The exception to this is step 1 in which vertices 0, 2 and 3 all have 0%.

I also found that as the number of steps tends towards infinity, the percentages for the number of times Annie ends on each vertex all tend to 20%.

3. Number of vertices variation

3.1. **The code.** After investigating how the number of steps changed things, I then investigated what happened when the number of vertices was changed. Similar to the number of steps variation, this involved minimal changes to the code for the initial investigation. I simply added more expressions into the list *vertices* to determine how many vertices there would be. I also then had to change the if statements on lines 150 and 156 which stopped the index becoming out of range. Also I added another print command with a new $.count(x)$ method for each new vertex.

3.2. **Findings.** For 6 vertices, I found that for an odd number of steps, Annie ends on vertices 0, 2 and 4 0% of the time. Likewise on an even number of steps vertices 1, 3 and 5 are not hit. Also as the number of steps increase, the percentage for each of the 3 vertices hit tend to $33.\dot{3}\%$ In fact the same will be observed for any even number of vertices. As every even vertex (and vertex 0) only connects to an adjacent odd vertex, and likewise any odd vertex only connects to an adjacent even vertex. Since Annie starts on vertex 0, she then moves to an odd vertex and is no longer on any even one. In the next step she will move off whichever odd vertex she was on to an even vertex. And this continues in the same way, meaning that for an odd number of steps she will always end on an even vertex (or vertex 0), and for an even number of steps she will always end on an odd vertex. Furthermore, as the number of steps tends to infinity, the percentage of times that she will land on this vertex tends to $\frac{100}{\frac{1}{2}(number\ of\ vertices)}$.

For 7 vertices, I found that the percentages were the same for vertices 1&6, 2&5 and 3&4. For the lower integers, with an odd number of steps, vertices 1&6 are landed on most often, followed by 3&4, then 2&5, and vertex 0 is landed on least often. For an even number of steps this order is reversed. As the number of steps tends to infinty, the percentage of times that Annie lands on each vertex is 14.2857...%. In fact, for any polygon with n vertices, where n is odd, for a lower number of steps, vertices $(0+1)$ & $(n-1)$, $(0+2)$ & $(n-2)$, $(0+3)$ & $(n-3)$, ... , $(0+(\frac{n-1}{2}))$ & $(n-(\frac{n-1}{2}))$, will be landed on by Annie with the same percentage. Then as the number of steps tends towards infinity, the percentage of times each vertex is landed on by Annie will tend to $\frac{100}{(number\ of\ vertices)}$.

4. Cube variation

4.1. **The code.** For the next variation, I investigated what would happen if Annie was moved onto a cube rather than a polygon. The main difference with a cube compared with a polygon is that each vertex is connected to another three vertices, unlike a polygon which is only connected to two. To simulate this I created a variable x which generated a random integer between 1 and 9, and split this into 3 groups: 1, 2 and 3; 4, 5 and 6; and 7, 8 and 9. I used three *if* statements that simulated Annie's movement to one of the 3 adjacent vertices depending on which group the variable x was in. This meant that Annie's movement was random.

In order to simulate the movement between vertices, I again used a list as in the code for the pentagon. I thought of the cube as two connected squares, one with vertices 0, 1, 2 and 3 and one with vertices 5, 6, 7 and 8. For each step Annie can either move clockwise along the current square, anti-clockwise along the current square, or across to the other square. I used similar code to the initial investigation for moving clockwise and anti-clockwise, and added 5 to the index to simulate Annie moving square, with an *if* statement for each vertex in the case that Annie moves from the second square to the first. I needed an expression, 4, in the list *vertices* so that there was a distinction between the last vertex 3 in the first square and the first vertex 5 in the second. Since to avoid an index error I had to change 4 back to 0 for the ant moving clockwise around the first square, if 4 had been the first vertex of the second square, Annie could have never landed there.

4.2. **Findings.** I found that after 7 steps Annie landed on vertices 0, 2, 6 and 8 with the same percentage, and that Annie never ended on vertices 1, 3, 5 and 7. For an even number of steps vertices 0, 2, 6 and 8 will not be hit and vertices 1, 3, 5 and 7 will be hit the same percentage of times. As the number of steps tends towards infinty, the percentage of times that Annie lands on each vertex will be 25%.