ISC4221C - Final Project:
Luck-based Story Game

by: Andres Candido

Introduction:

I have always enjoyed making games as final projects for my programming classes. I think they allow me to show mastery over the contents of course in fun and creative ways.For other previous classes I made a “pong” style tennis game and a 2-D platformer. This time I wanted to try making something different, a luck-based story game, where what happens in the story is determined by rolling a dice. I wanted the player to get a negative,neutral, or positive outcome depending on the number they roll. I also wanted them to be able to see how their overall luck compares to the luck of other players, is it good, bad, or average?

I thought this type of game would be fitting as a final project for this class since many of the ideas and mathematical methods used in this game are related to the “Random process” module of the course.

Below I will explain the ways in which I was able to achieve all of these things.

Methods:

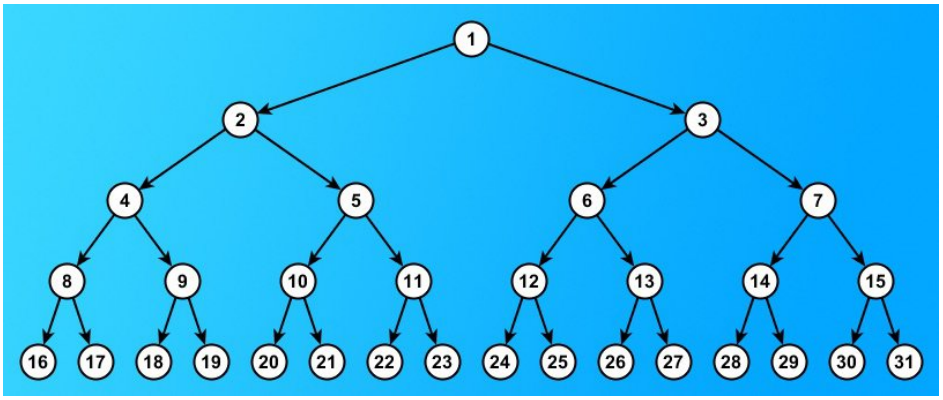
The major challenges of this project can be divided into three parts:

1. Story structure (Branching or Block-based)
2. Mathematical Methods (CDF and Normal distribution of “player” scores)
3. Game output (More specifically plotting)

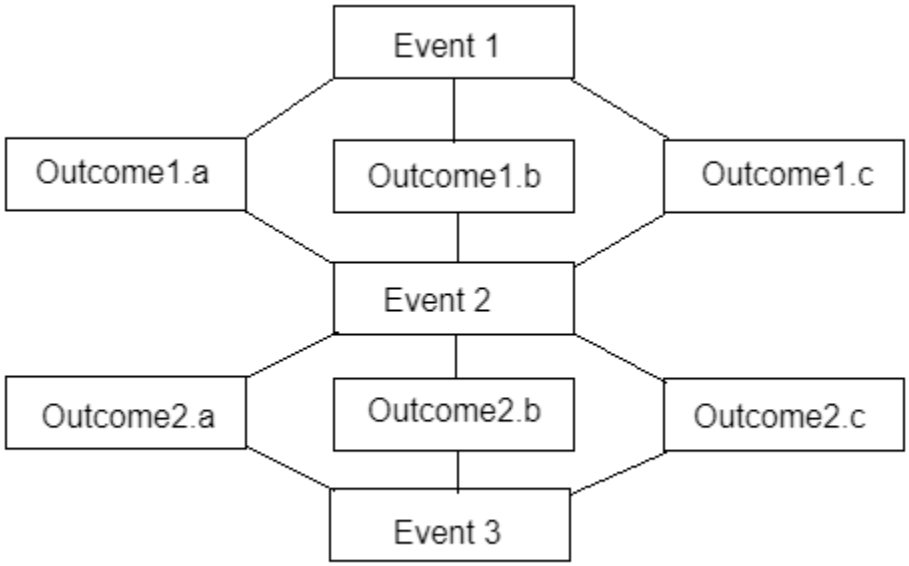
I will explain all methods related to each one of them, starting with:

1. Story structure:

One of the first challenges I came across at the beginning of the project was deciding which structure the story should follow, a branching structure or a block-based structure.
A branching structure means that the story changes (branches) based on the outcome of the dice roll. This type of story-telling would require a lot of writing since I would need to make a dedicated set of paragraphs for each possible order of outcomes. The structure would look something like this:



On the other hand, a “Block-based” structure is far less complex. In this case the story can also change based on the outcome of the dice roll. But instead of going in diverging paths, all outcomes re-converge into the same next event. I opted to choose this structure for my game in order to keep the story simple. This type of structure would look something like this:



2. Mathematical Methods:

The main idea of the game is that what happens in the story is determined by rolling a dice (Random integer between 1 and 6). Rolling a 1 or 2 leads to bad luck, a 3 or 4 to neutral luck, and a 5 or 6 leads to good luck.
In order to keep track of all of the player’s rolls we add them in a variable called “luck_score.”

There are two mathematical methods used as part of the output of the game. The first one is a cumulative representation of the player’s luck score, similar to a CDF, it shows how the luck score increased as the game went on. For this method we used:

```
for i in range(len(luck_matrix)):
    if i == 0:
      cdf[i]=luck_matrix[i]
    else:
      cdf[i] = cdf[i-1]+luck_matrix[i]
```

Which is similar to the sum formula:

$$\sum_{i=1}^{n} x_i = x_1 + x_2 + \cdots + x_n$$

The other mathematical method used was comparing the player's luck score to the luck score of other players. For this, I decided to simulate a sample population of 100000 players, make a normal distribution out of their luck scores and then fit the real player's luck score into the graph to see how their luck compares. For this I wrote code in python that would work similarly to the normal distribution formula:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

3. Game output:

The output of the game includes graphs of the two previously mentioned Mathematical Methods. In order to display both of them at the same time, we need to create a plot matrix and then put the two plots into their corresponding slots in the matrix. For this we used the following lines of code:

```
plot1 = plt.subplot2grid((1, 4), (0, 0), colspan=2)
plot2 = plt.subplot2grid((1, 4), (0, 2), colspan=2)

plot1.plot([1,2,3,4],cdf)
plot1.set_title("Your cumulative luck score")

plot2.plot(dice,prob_trials)
plot2.vlines(luck_score,0,0.12,color='r')
plot2.set_title("Your luck score compared to others")
```

```
# Packing all the plots and displaying them
plt.tight_layout()
plt.show()
```

Numerical Details:
The game program was developed using Python 3.8.16 and Google Collaboratory. However, after testing, It also works on Jupyter Notebook (v6.5.2) and IDEs such as Visual studio Code 2022 (v1.73.1).

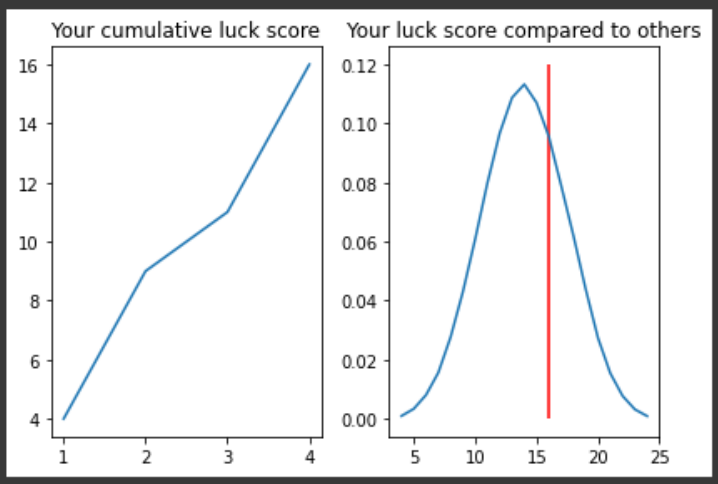
Results and Discussion:
Program output (I will just include the game summary and some of the story since the entire story is long… and contains spoilers):

```
You turn and see the creature stands on its hind legs and lets out a loud growl as the car mo
The ranger turns to you and says: "You are very lucky I found you there and then son, that wa
You feel fortunate to have escaped from that forest.

The end.

---------------------------------- Game Summary ------------------------------------

 Your luck score was: 16   Your luck was average!
```



Above are the two graphs showing the previously mentioned cumulative luck score and luck score compared to a normal distribution of the luck scores of 10000 simulated players. The red line represents where the real player's luck score falls in the distribution. At the top of the "Game Summary" section we can also see a fraction of the end of the story this particular player obtained, there is a total of three different endings.

Conclusion and Outlook:

Overall, I am very satisfied with the results of this project. Since at the end of the day this was meant to be a game, there weren't any major findings. However, I did come across some limitations and possible ways to improve it. One of the main limitations I came across was trying to use the Tkinter library to create GUI elements for the game. Initially I wanted to display the game on a separate window after the player started the program, unfortunately Tkinter has some conflicts with Google Collab notebooks and Jupyter notebooks so I decided not to use it for this project.

As for ways to improve the game, I would have liked to integrate sound effects to play after the player rolls the dice, a different sound effect could play depending if the outcome is positive, negative, or neutral.