MAT 3772 Project 3

December 19, 2019

0.1 MAT 3772 Project 3 : Chute and Ladders

0.1.1 ABSTRACT

Chute and Ladders is a very common game where the goal is to reach the end of grid start from position 1. A dice is thrown and whichever number appears on the dice will be used to make progress. For example if you throw the dice and a 6 comes out, therefore you can move up for 6 steps. However, there are traps and shortcuts to make you arrive faster or slower if you fall into a trap. An example of the game can be shown looking at picture of cell below

0.1.2 Introduction

The goal of this report is to used montecarlo simulations to see in average how many turns are needed in order for 1 player to finish the game.

The default number of simulation that will be used is 10000.

****TABLE OF CONTENTS****

[12]:
```
# import numpy for random number and computations
import numpy as np

# Matplotlib to graph results
import matplotlib.pyplot as plt
```

We define a function that unique goal is to throw a dice and return the result of the throw.

[13]:
```
# The function uses numpy.random.randint() to generate random integers from a␣
↪specified range
def throw_dice():
    return np.random.randint(0,6) + 1
```

The number of simulations being done is defined by n

[14]:
```
n = 10000
```

Use **dictionaries for mapping** and **check if for a given position, there is an associated chute or ladders** We will define different number of chutes and ladders to generate different simulations

[15]:
```
chutes_ladders1 = {# Ladders   # Chutes
                    1:38,       16:6,
                    4:14,       87:24,
                    9:31,       47:26,
                    21:42,      49:11,
                    28:84,      56:53,
                    36:44,      62:19,
                    51:67,      64:60,
                    71:91,      98:78,
                    80:100,     93:73}
```

[16]:
```
chutes_ladders2 = {# Ladders   # Chutes
                    1:38,       16:6,
                    9:31,       47:26,
                                49:11,
                                56:53,
                                64:60,
                                98:78}
```

[17]:
```
# This list will contain number of turns per game for 10000 games
overall_nb_turn    = []
```

[18]:
```
def simulation(chute_ladder):

   # We keep playing as long as we have not reached n
   for counter in range (n):

      # How many times a player need to play in order to win
      nb_turns_per_games = 0
      curr_pos           = 0

      # While a game is not over
      while curr_pos < 100:

         curr_pos += throw_dice()
```

```
        nb_turns_per_games += 1

        # If the position correspond to either a chute or ladder
        if chute_ladder.get(curr_pos) != None:
            curr_pos = chute_ladder.get(curr_pos)

    overall_nb_turn.append(nb_turns_per_games)
    counter += 1
```

[19]:
```
overall_nb_turn = []
simulation(chutes_ladders1)
```

[0]:
```
# A game last in average
np.mean(np.array(overall_nb_turn))
```

Plot of the histogram

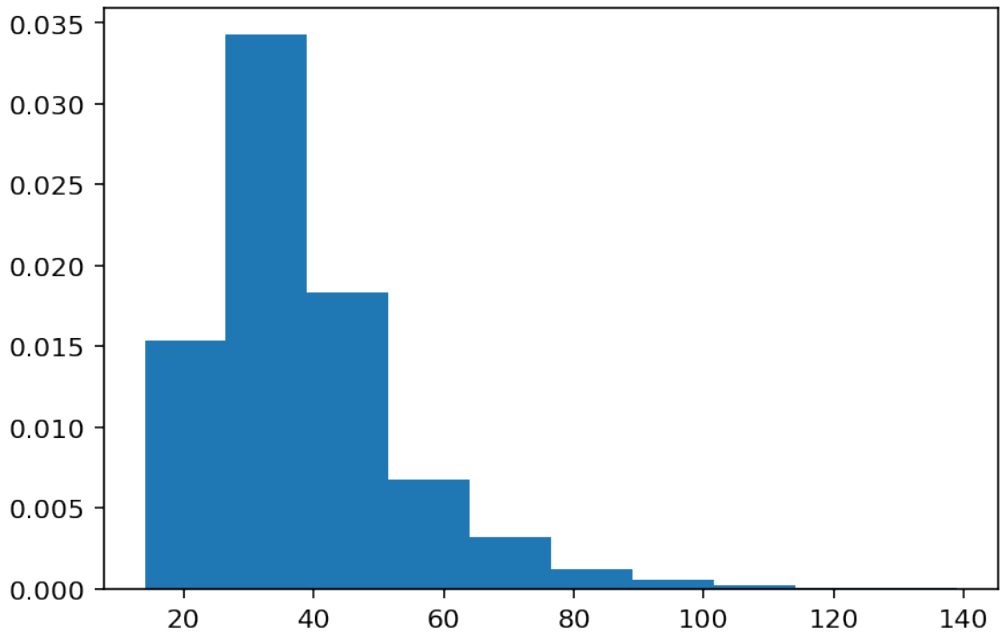
[0]:
```
plt.hist(overall_nb_turn, density = True)
```

[98]:
```
overall_nb_turn = []
simulation(chutes_ladders2)

plt.hist(overall_nb_turn, density = True)
```

[98]:
```
(array([1.5368e-02, 3.4248e-02, 1.8288e-02, 6.7680e-03, 3.1840e-03,
        1.1840e-03, 5.7600e-04, 2.5600e-04, 8.8000e-05, 4.0000e-05]),
 array([ 14. ,  26.5,  39. ,  51.5,  64. ,  76.5,  89. , 101.5, 114. ,
        126.5, 139. ]),
 <a list of 10 Patch objects>)
```

[98]:



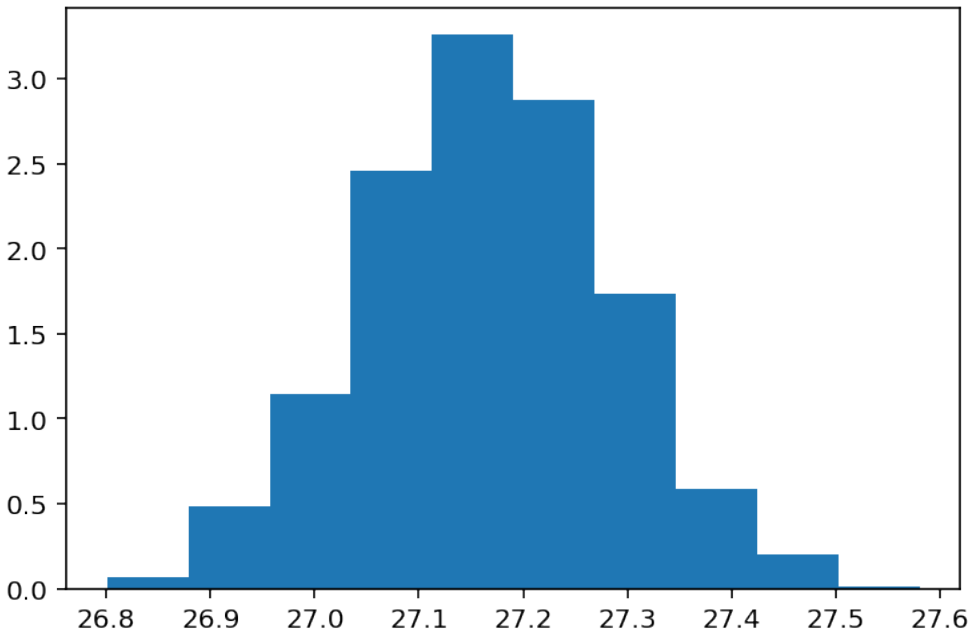
[11]:
```
# Avg stored the average overall number of turns of 1000 games for 100␣
 ↪simulations
avg = []

# Proceed to 1000 simulations of 1000 games
for i in range(1000):
   overall_nb_turn = []
   simulation(chutes_ladders1)
   avg.append(np.mean(np.array(overall_nb_turn)))

# Plot the result
plt.hist(avg, density = True)
```

[11]:
```
(array([0.06414368, 0.48749198, 1.14175754, 2.46311738, 3.25849904,
        2.87363695, 1.73187941, 0.59012187, 0.20525978, 0.01282874]),
 array([26.8008 , 26.87875, 26.9567 , 27.03465, 27.1126 , 27.19055,
        27.2685 , 27.34645, 27.4244 , 27.50235, 27.5803 ]),
 <a list of 10 Patch objects>)
```

[11]:



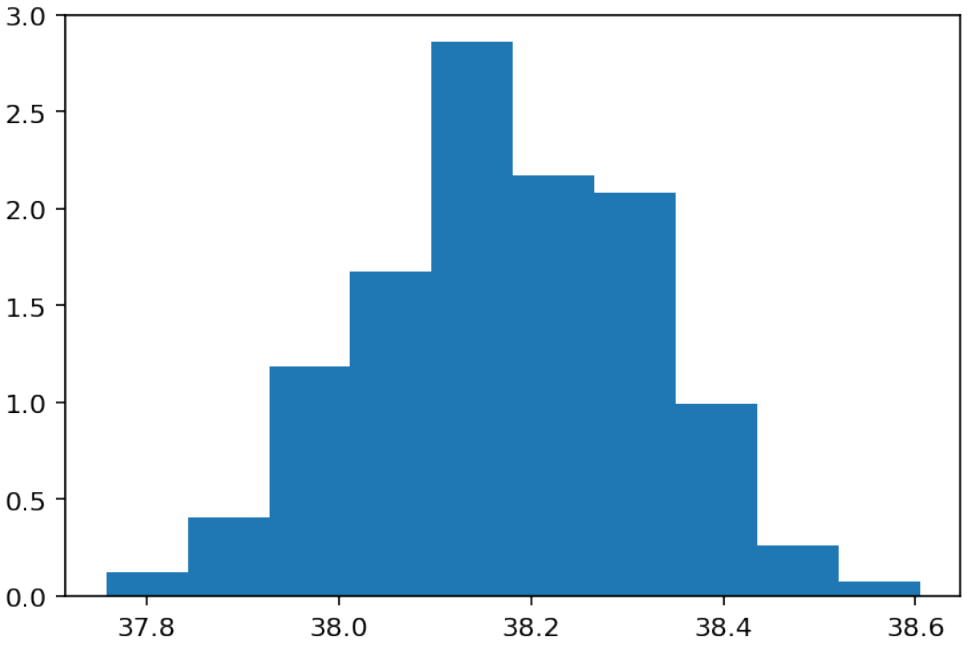
[103]:
```
# Avg stored the average overall number of turns of 1000 games for 100␣
 ↪simulations
avg = []

# Proceed to 1000 simulations of 1000 games
for i in range(1000):
   overall_nb_turn = []
   simulation(chutes_ladders2)
   avg.append(np.mean(np.array(overall_nb_turn)))

# Plot the result
plt.hist(avg, density = True)
```

[103]:
```
(array([0.11813349, 0.40165387, 1.18133491, 1.67749557, 2.85883048,
        2.17365623, 2.07914944, 0.99232132, 0.25989368, 0.07088009]),
 array([37.758  , 37.84265, 37.9273 , 38.01195, 38.0966 , 38.18125,
        38.2659 , 38.35055, 38.4352 , 38.51985, 38.6045 ]),
 <a list of 10 Patch objects>)
```

[103]:



1 Conclusion

To conclude we were able to simulate chutes and ladders game for a 10 x 10 grids. We notice the average number of turns per games in order to finish for 1 player varied from one simulation to another. That average depends on how chutes and ladders there were.