Assignment-1 6369
Aishwarya Pothula
1001743470

1a) The equation of the utility function for the problem as a function of the starting floor of the elevator (f) and the probability distribution that persons will call the elevator from a particular floor, (s_prob) and want to exit at any other floor, (es_prob) is. S is the call floor and e is the exit floor.

$- \sum_{s,e} (s\_prob * es\_prob * ((abs(s - f) + abs(s - e) + 1)* 7))$

b) Formulate an n-armed bandit problem to determine the ideal (fixed) starting floor for the elevator. Indicate all the parts of your formulation of the n-armed bandit problem.

It is a 6-armed bandit problem, with each of the arms representing the 6 starting floor possibilities for the elevator.

The utility function for each floor will be the penalty calculated for each floor. The concept of total probability is used to calculate the penalty for each floor; the summation of all possible combinations of call floor and exit floor for each starting floor are multiplied with the probability of calling and probability of exit floor given calling floor gives the penalty for each floor.

c)
i) in the early morning all persons come in on the first floor and exit on each of the other floors (2-6) with uniform probability;

I have solved for the starting floor of the elevator by considering the s_prob of calling floor1 as 1 and se_prob of exit floors as 1/5. For this case, a starting floor of 1 would yield the lowest penalty for the elevator.

ii) at noon people call the elevator from floors 2-6 with a uniform distribution and all exit on the first floor

I have solved for the starting floor of the elevator by considering the s_prob of calling floors as 1/5 and se_prob of exit floor1 as 1 . For this case, a starting floor of 4 would yield the lowest penalty for the elevator.

iii) in the afternoon, 50% of the time the elevator is called from the second floor where people want to go to the other floors with a uniform distribution while the other 50% of the time the

elevator is called from floors 2-6 with a uniform distribution with persons always exiting on the 1st floor.

For this case I have calculated the penalties of each floor for each subcase and averaged them to represent the 50% chance of either subcase taking place. I have considered the probabilities of c(i) and c(ii) for subcases 1,2 respectively.

For this case, starting floor of 2 would yield the lowest penalty for the elevator.

d) Implement an incremental averaging solution to your n-armed bandit. Persons call the elevator from each of the floors with a uniform probability and are going to travel to other floors with a probability distribution where when going down the probability to travel to each of the other floors is proportional to the number of floors that they
Make sure to submit the code as well as a graph showing the estimated utilities for the actions in your n-armed bandit at each iteration (i.e. after each person used the elevator) for the first 500

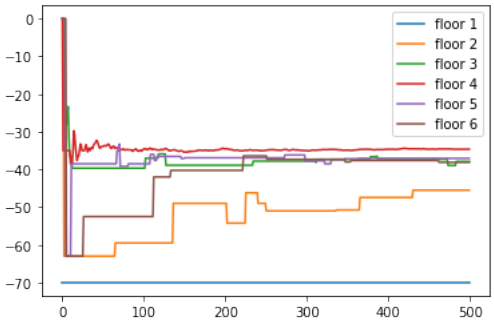
I have used the (1 - 1.0 / N) * mean + 1.0 / N * x equation to perform incremental averaging where N is the number of time a particular floor has been selected as the starting floor and x is the latest penalty.

The probabilities I have considered according the given information are

```
if s == 0:
    e = np.random.choice([0,1,2,3,4,5], p = [0, (5/25), (5/25), (5/25), (5/25), (5/25)])
if s == 1:
    e = np.random.choice([0,1,2,3,4,5], p = [(1/21), 0, (5/21), (5/21), (5/21), (5/21)])
if s == 2:
    e = np.random.choice([0,1,2,3,4,5], p = [(2/18), (1/18), 0, (5/18), (5/18), (5/18)])
if s == 3:
    e = np.random.choice([0,1,2,3,4,5], p = [(3/16), (2/16), (1/16), 0, (5/16), (5/16)])
if s == 4:
    e = np.random.choice([0,1,2,3,4,5], p = [(4/15), (3/15), (2/15), (1/15), 0, (5/15)])
if s == 5:
    e = np.random.choice([0,1,2,3,4,5], p = [(5/15), (4/15), (3/15), (2/15), (1/15), 0])
```

I have used epsilon-greedy algorithm for the explore-exploit dilemma in this learning problem. I have considered epsilon value of 0.1.

This is the graph of the rewards for each actions for the first 500 iterations



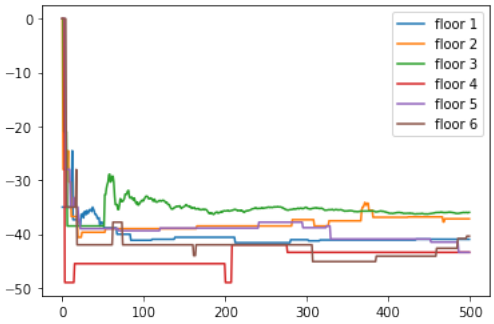
For this distribution, starting floor of 4 seems to be giving the lowest penalty when tested over 500 iterations.

e) Use your learning implementation from part d) to solve the same problem for 3 more distributions of your own choice and discuss the behavior of the n-armed bandit learner.

I have considered the following distributions. The probability distribution for call floors is always uniform = 1/6

Distribution1: probability of exit floor given start floor is 2 ^ (floors to cross -1) * p, 0 when same as start floor, 2 ^ 5 * p to travel to floors above.
Calculated probabilities are

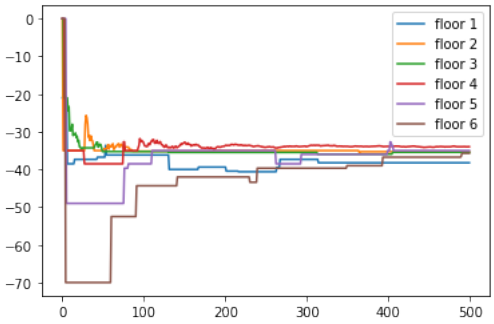
```
if s == 0:
    e = np.random.choice([0,1,2,3,4,5], p = [0, (16/80), (16/80), (16/80), (16/80), (16/80)])
if s == 1:
    e = np.random.choice([0,1,2,3,4,5], p = [(1/65), 0, (16/65), (16/65), (16/65), (16/65)])
if s == 2:
    e = np.random.choice([0,1,2,3,4,5], p = [(2/51), (1/51), 0, (16/51), (16/51), (16/51)])
if s == 3:
    e = np.random.choice([0,1,2,3,4,5], p = [(4/39), (2/39), (1/39), 0, (16/39), (16/39)])
if s == 4:
    e = np.random.choice([0,1,2,3,4,5], p = [(8/31), (4/31), (2/31), (1/31), 0, (16/31)])
if s == 5:
    e = np.random.choice([0,1,2,3,4,5], p = [(16/31), (8/31), (4/31), (2/31), (1/31), 0])
```



For this distribution, starting floor of 3 seems to be giving the lowest penalty when tested over 500 iterations.

Distribution2: probability of exit floor given start floor is 2 * p, 0 when same as start floor
Calculated probabilities are

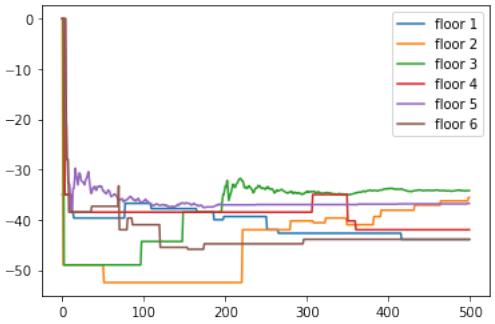
```
if s == 0:
        e = np.random.choice([0,1,2,3,4,5], p = [0, (2/10), (2/10), (2/10), (2/10), (2/10)])
    if s == 1:
        e = np.random.choice([0,1,2,3,4,5], p = [(2/10), 0, (2/10), (2/10), (2/10), (2/10)])
    if s == 2:
        e = np.random.choice([0,1,2,3,4,5], p = [(2/10), (2/10), 0, (2/10), (2/10), (2/10)])
    if s == 3:
        e = np.random.choice([0,1,2,3,4,5], p = [(2/10), (2/10), (2/10), 0, (2/10), (2/10)])
    if s == 4:
        e = np.random.choice([0,1,2,3,4,5], p = [(2/10), (2/10), (2/10), (2/10), 0, (2/10)])
    if s == 5:
        e = np.random.choice([0,1,2,3,4,5], p = [(2/10), (2/10), (2/10), (2/10), (2/10), 0])
```



For this distribution, starting floor of 4 seems to be giving the lowest penalty when tested over 500 iterations.

Distribution3: probability of exit floor given start floor is floor no * p, 0 when same as start floor
Calculated probabilities are

```
if s == 0:
        e = np.random.choice([0,1,2,3,4,5], p = [0, (2/20), (3/20), (4/20), (5/20), (6/20)])
    if s == 1:
        e = np.random.choice([0,1,2,3,4,5], p = [(1/19), 0, (3/19), (4/19), (5/19), (6/19)])
    if s == 2:
        e = np.random.choice([0,1,2,3,4,5], p = [(1/18), (2/18), 0, (4/18), (5/18), (6/18)])
    if s == 3:
        e = np.random.choice([0,1,2,3,4,5], p = [(1/17), (2/17), (3/17), 0, (5/17), (6/17)])
    if s == 4:
        e = np.random.choice([0,1,2,3,4,5], p = [(1/16), (2/16), (3/16), (4/16), 0, (6/16)])
    if s == 5:
        e = np.random.choice([0,1,2,3,4,5], p = [(1/15), (2/15), (3/15), (4/15), (5/15), 0])
```



For this distribution, starting floor of 3 seems to be giving the lowest penalty when tested over 500 iterations.

2. Consider that this is also the case for the utility of waiting time and assume that the perceived ”penalty” of waiting instead behaves quadratic in terms of the wait time

a) Assuming the same elevator scheduling problem as in Question 1 and the quadratic ”penalty” for waiting, design a new utility function for the problem as a function of the starting floor(f) of the elevator and the probability distribution that persons will call the elevator from a particular floor, (s_prob) and want to exit at any other floor, (se_prob).

$-\sum_{s,e}$ np.power((abs(s - f) + abs(s - e) + 1)* 7, 2

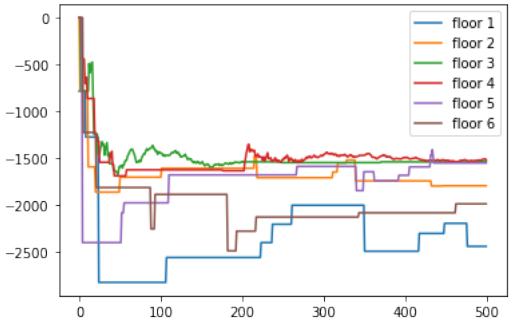
b) Modify your n-armed bandit code to use this utility function and re-run the experiments from parts 1 d) and 1 e). Show the results and discuss the differences in behavior of the elevator. Again, make sure to submit the code as well as a graph showing the estimated utilities for the actions in your n-armed bandit at each iteration (i.e. after each person used the elevator) for the first 500 iterations.

I have considered the following distributions. The probability distribution for call floors is always uniform = 1/6

Distribution1: probability of exit floor given start floor is 2 ^ (floors to cross -1) * p, 0 when same as start floor, 2 ^ 5 * p to travel to floors above.
Calculated probabilities are

```
if s == 0:
        e = np.random.choice([0,1,2,3,4,5], p = [0, (16/80), (16/80), (16/80), (16/80), (16/80)])
    if s == 1:
        e = np.random.choice([0,1,2,3,4,5], p = [(1/65), 0, (16/65), (16/65), (16/65), (16/65)])
    if s == 2:
        e = np.random.choice([0,1,2,3,4,5], p = [(2/51), (1/51), 0, (16/51), (16/51), (16/51)])
    if s == 3:
        e = np.random.choice([0,1,2,3,4,5], p = [(4/39), (2/39), (1/39), 0, (16/39), (16/39)])
    if s == 4:
```

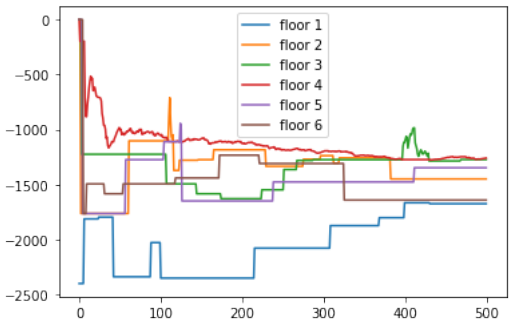
```
    e = np.random.choice([0,1,2,3,4,5], p = [(8/31), (4/31), (2/31), (1/31), 0, (16/31)])
if s == 5:
    e = np.random.choice([0,1,2,3,4,5], p = [(16/31), (8/31), (4/31), (2/31), (1/31), 0])
```



For this distribution, starting floor of 4,5 seems to be equally giving the lowest penalty when tested over 500 iterations.

Distribution2: probability of exit floor given start floor is 2 * p, 0 when same as start floor
Calculated probabilities are

```
if s == 0:
    e = np.random.choice([0,1,2,3,4,5], p = [0, (2/10), (2/10), (2/10), (2/10), (2/10)])
if s == 1:
    e = np.random.choice([0,1,2,3,4,5], p = [(2/10), 0, (2/10), (2/10), (2/10), (2/10)])
if s == 2:
    e = np.random.choice([0,1,2,3,4,5], p = [(2/10), (2/10), 0, (2/10), (2/10), (2/10)])
if s == 3:
    e = np.random.choice([0,1,2,3,4,5], p = [(2/10), (2/10), (2/10), 0, (2/10), (2/10)])
if s == 4:
    e = np.random.choice([0,1,2,3,4,5], p = [(2/10), (2/10), (2/10), (2/10), 0, (2/10)])
if s == 5:
    e = np.random.choice([0,1,2,3,4,5], p = [(2/10), (2/10), (2/10), (2/10), (2/10), 0])
```

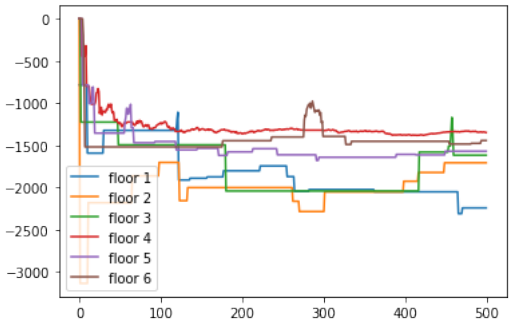


For this distribution, starting floor of 3,4 seems to be equally giving the lowest penalty when tested over 500 iterations.

Distribution3: probability of exit floor given start floor is floor no * p, 0 when same as start floor

Calculated probabilities are

```
if s == 0:
    e = np.random.choice([0,1,2,3,4,5], p = [0, (2/20), (3/20), (4/20), (5/20), (6/20)])
if s == 1:
    e = np.random.choice([0,1,2,3,4,5], p = [(1/19), 0, (3/19), (4/19), (5/19), (6/19)])
if s == 2:
    e = np.random.choice([0,1,2,3,4,5], p = [(1/18), (2/18), 0, (4/18), (5/18), (6/18)])
if s == 3:
    e = np.random.choice([0,1,2,3,4,5], p = [(1/17), (2/17), (3/17), 0, (5/17), (6/17)])
if s == 4:
    e = np.random.choice([0,1,2,3,4,5], p = [(1/16), (2/16), (3/16), (4/16), 0, (6/16)])
if s == 5:
    e = np.random.choice([0,1,2,3,4,5], p = [(1/15), (2/15), (3/15), (4/15), (5/15), 0])
```



For this distribution, starting floor of 4 seems to be giving the lowest penalty when tested over 500 iterations.

Note1: The graphs and ,accordingly, ideal starting floors of elevators change everytime the programs are run because of sampling.

Note2:To run the programs type python3 <filename.py>

Note3: In 1-c, please change the case to “morning”, “noon” or “afternoon” in the program (in this line c_1 = run_experiment('case_name') to run different cases,