FFXIV: Mini Cactpot

Overview

Mini Cactpot is a daily activity of the famous MMORPG Final Fantasy XIV. Players will be able to win Manderville Gold Saucer Points(MGP) through this lottery system. Similar to a scratch-off lottery ticket, you'll reveal numbers, and depending on the sum of the numbers the rewards you can obtain will differ.

In Mini Cactpot, every ticket has nine spaces, each numbered randomly from one to nine. At the start, however, eight of these nine spaces will be hidden. To begin, select three numbers to uncover from the eight hidden on your ticket.

Next, select one of eight lines, vertical, horizontal, or diagonal. When selecting a line, the sum of the three numbers in that line will determine the amount of MGP you receive.

After you have selected a line, all the numbers are uncovered and you will receive MGP based on the sum of the line you chose.

Q7 often struggles with which line is the optimal choice given the 4 uncovered numbers. Can you design an algorithm to help Q7 choose the line with the highest expected amount of MGP?

Q1: Naive Solver

In this part, you are a *cheater* that can choose a best line with all the numbers uncovered.

Your should implement the `solveq1` method with a list as its argument denotes the 9 numbers, e.g. [[7, 6, 9], [4, 5, 3], [2, 1, 8]]. It should return the maximum amount of MGP the player can win by choosing an optimal line.

In this case, the answer should be 306.

Explanation: The optimal line to choose is the 3rd column, the sum is 20 and the corresponding reward is 306 MGP.

Q2: Real Solver

In this part, you no longer have all the numbers uncovered but only have 4 instead.

Your should implement the `solveq2` method with a list as its argument denotes the 9 numbers, e.g. [[0, 6, 0], [4, 0, 3], [2, 0, 0]]. It should return the maximum expected amount of MGP (only keep the integer part) the player can win by choosing an optimal line.

In this case, the answer should be 580.

Explanation: The expected rewards that can be obtained by choosing three rows are 389,226,180, the rewards that can be obtained by three columns are 118,389,168, and those of the two diagonals are 580,180. So the optimal choice is the main diagonal, and the corresponding expected reward is 580 MGP.

Grading Criteria

- The assignment has two parts, with the first part `Naive Solver` accounting for 30 points and the second part `Real Solver` 70 points.
- Online submissions close at 11:59 on Sunday, October 17. No late submission is allowed.
- You can submit up to 20 times. After this threshold, each additional submission is penalized by 10% of points. Only the last submission will be recorded for your final score.
- The code you submit will be archived and kept by the university, so please be mindful of academic integrity.
- No third-party library is allowed, but you can use built-in libraries such as `copy` and `itertools`.

Code Template

```
#!/usr/bin/env python3
score_mapping = {6: 10000, 7: 36, 8: 720, 9: 360, 10: 80, 11: 252, 12: 108, 13:
72, 14: 54,
                 15: 180, 16: 72, 17: 180, 18: 119, 19: 36, 20: 306, 21: 1080, 22:
144, 23: 1800, 24: 3600}


def solveq1(data):
    ans = 10_000

    # implement your algorithm here

    return ans


def solveq2(data):
    ans = 10_000

    # implement your algorithm here

    return ans
```

```
def main():
    print(solveq1([[7, 6, 9], [4, 5, 3], [2, 1, 8]]))
    print(solveq2([[0, 6, 0], [4, 0, 3], [2, 0, 0]]))


if __name__ == "__main__":
    main()
```

Hint

1. For Q2, it may be difficult for a math noob like me to calculate the expected reward directly. An easier way is to consider all the possible permutations of the remaining numbers, convert the problem into the form of Q1, and just add up all the possible results and average them. You might need this: https://docs.python.org/3.9/library/itertools.html#itertools.permutations

```
In [1]: from itertools import permutations

In [2]: for perm in permutations([1,2,3]):
   ...:     print(perm)
   ...:
(1, 2, 3)
(1, 3, 2)
(2, 1, 3)
(2, 3, 1)
(3, 1, 2)
(3, 2, 1)
```

2. How to modify `b` but keep `a` unchanged? Check the tutorial slides!

```
In [1]: a = [[1,2,3],[4,5,6],[7,8,9]]

In [2]: b = a

In [3]: b[0][0] = 233

In [4]: b
Out[4]: [[233, 2, 3], [4, 5, 6], [7, 8, 9]]

In [5]: a
Out[5]: [[233, 2, 3], [4, 5, 6], [7, 8, 9]]
```