Computer Science Team Week 14

Henry Gustafson Julian Bauer

The College Preparatory School

Computer Science Team
March 25, 2024

ACSL 3

How was it?

Also, do programming problems!!! Shout out to Aidan and Julian for doing it.

Fun coding problems

Theme: Royal Game of Ur

Problem Dice

Problem Dice Joshua is playing the Royal Game of Ur but doesn't want to use real-life dice because they hurt their fingers. Help them out by creating a function that accurately simulates Royal Game of Ur dice rolls (for those who don't know, each Royal Game of Ur dice roll consists of flipping four coins and adding up the amount of heads). **Extra fun:** Only generate one random number in your function, while keeping the probability distribution accurate.

```
def roll() -> int
```

Problem Dice

Example

```
assert roll() in [0, 1, 2, 3, 4]
```

Problem Wand

Problem Wand You're playing a simplified, solo version of the Royal Game of Ur where you're trying to get all your pieces off of the board (which is shaped like a straight line). Methuselah gives you a magic wand that can alter future dice rolls. Given *length*, the length of your board, and *positions*, a list of indices that tells you the position of each of your pieces on the board, output the minimum number of turns needed to get all your pieces out (you roll once every turn, one piece can move the amount you rolled every turn, pieces can only move to the right, no pieces can be on the same square, a piece escapes if its position is $\geq n$)

```
def escape(length: int, positions: list[int]) -> int
```

Problem Wand

Example

```
assert escape(5, [0, 1, 2]) == 4
# turn 1: roll a 3, piece at 2 goes to 5 and escapes
# turn 2: roll a 4, piece at 1 goes to 5 and escapes
# turn 3: roll a 4, piece at 0 goes to 4
# turn 4: roll a 1, piece at 4 goes to 5 and escapes
# all pieces have escaped
```

Problem Flower

Problem Flower You put on your apple vision pro with built-in glasses and realize that some of the tiles on your board are Rosette tiles. In the Royal Game of Ur, when a piece lands on a Rosette tile, you can roll and move a piece again in the same turn! Create a function that takes *board*, a string of 1's or 0's where 1 means that the tile at that index is a Rosette, and the same *positions* variable, and output the minimum number of turns needed to get all your pieces off the board.

Problem Flower

Example

```
assert escape("01000", [0, 1, 2]) == 3
# turn 1: roll a 3, piece at 2 goes to 5 and escapes
# turn 2: roll a 4, piece at 1 goes to 5 and escapes
# turn 3:
#    roll a 1, piece at 0 goes to 1 which is rosette
#    roll a 4, piece at 1 goes to 5 and escapes
# all pieces have escaped
```

The End

Questions? Comments? Remarks?
Considerations? Confusions?