twenty48

Kishore Hari

2022-10-31

Making the game 2048

The strategy briefly

We need a matrix printed, 4X4. Start it with random numbers, highly sparse (i.e., mostly zeros). Print this matrix in the console. Then take input continuously until 2048 appears. With each input we need to update the matrix properly.

Functions that can together achieve the strategy

Here are the questions that you need to answer:

1. Write a function to print a blank matrix using 4 vectors, each of length 4. If any of the elements is zero, print a ".". For example, the vector c(0,0,2,0) will be printed as ". . 2 ."
2. Write a function to generate four random vectors, each of length 4. The elements of the random vectors must be taken from a seed vector that is passed as an argument. That is, if the seed vector is c(0,2), all elements of the vectors that you generate must either be 0 or 2. How can you make it so that there are more zeros than 2's?
3. Write a function (say p) that takes four vectors and a function (call it f) as arguments (so that there will be 5 arguments, one of them is another function. Google this idea and see how this is implemented, if you are confused). The function p does the following tasks:
   i. Take an input from the user. The inut must be one of the four letters "w", "A", "S", "D".
   ii. Call f, with the 4 vectors and the letter input. Assume that the output of f would be the four vectors in a list.
   iii. clear the console. (The expected effect is that the console becomes empty. Find out how to do this by googling).
   iv. Print the new vectors using the function you have written as answer to question 1.
   v. return the new vectors. (How do you return four vectors?)
4. Now you need to write f. This is the trickiest part of coding the game. The logic is as follows:
   i. There are two operations: stacking and sum. If same numbers face each other, they sum. if Different numbers face each other, they stack.
   ii. Upon pressing W, All numbers except for the first row will move up. Each number can skip the zeros in its way. When it encounters a non-zero number, it will decide whether to stack or to sum.
   iii. If the number in the last row moves up to the first row, you need to fill the slots from the 2nd row to last row in that column with numbers randomly generated from a seed vector.
   iv. The seed vector needs to be updated to include the newly emerging numbers in the summing up.

While this is the general idea, the kicker is putting these steps in a programmable way. Think about how to do that and see if you can write a function that takes the 4 vectors and a user input, updates the vectors using the logic above, and returns the updated set of vectors.

5. Finally, combine these functions and call them in the right order so that you can get the game running. Note that you need to keep calling the function until the number 2048 appears, or the vectors don't change no matter what direction is given, i.e., no sum operations are possible anywhere.