Challenge 7 - Boozzle

Bob is really bad at Boozzle. He gets really angry when he loses against his friends and he has decided to do whatever it takes to win. He will pay a fortune for a machine that plays optimally.

Boozzle is played as follows. You are given a board of n rows and m columns. For each cell, there is a character, a score, and a score modifier. The goal is to find words in the board and achieve the maximum score possible in the given time. Words are built by joining adjacent cells in any direction (vertically, horizontally or diagonally).

The score for a word is given by

$$(S(c_1)*CM(c_1) + \ldots S(c_j)*CM(c_j)) * max(WM(c_1),\ldots,WM(c_j)) + j$$

where

- $c_1,\ldots,c_j$ is the path of cells that make up the word
- $S(c_i)$ is the score for the character of the cell i
- $CM(c_i)$ is the multiplier modifier for the score of the character ($WM(c_i)>1 \Rightarrow CM(c_i)=1$)
- $WM(c_i)$ is the multiplier modifier for the whole word ($CM(c_i)>1 \Rightarrow WM(c_i)=1$)

To help Bob, you have to develop an algorithm to control a robot that will play Boozzle optimally, taking into account the following:

- The interface with which the robot interacts is a touchscreen, so it has to select the words by using its "fingers", exactly like a human would.
- Because of that, it takes time to submit a word. Specifically, **it takes 1 second to select each of the cells that make up a word, plus 1 second more to press the "submit" button**.
- There is no extra delays between different words. (ie, it takes 11 seconds to submit two words of 4 and 5 characters, respectively)
- The same word cannot be submitted twice, even if there are different paths for the same word.
- All the cells building a word must be different (i.e., you cannot use the same cell twice for the same word).
- The only valid words are the ones in this dictionary.
- Time spent computing the best words is not taken into account.

Input and limits

- First line of the input contains the number of problems, **N**
  $1 \leq \mathbf{N} \leq 10$
- Scores for each caracter follow
  {'A': 1, 'C': 3, 'B': 3, 'E': 1, 'D': 2, 'G': 2, 'F': 4, 'I': 1, 'H': 4, 'K': 5, 'J': 8, 'M': 3, 'L': 1, 'O': 1, 'N': 1, 'Q': 5, 'P': 3, 'S': 1, 'R': 1, 'U': 1, 'T': 1, 'W': 4, 'V`: 4, 'Y': 4, 'X': 8, 'Z': 10}
- Next, the duration of the game in seconds, **W**
  $1 \leq \mathbf{W} \leq 300$
- Next, the number of rows, **n**

$1 \leq \mathbf{n} \leq 50$

- Next, the number of columns, **m**
  $1 \leq \mathbf{m} \leq 50$
- And finally, the board: **n** rows with **m** cells. Each cell is represented by **Wxy** where W represents the character, x is the multiplier type and y is the multiplier value ($1 \leq y \leq 3$):
  If x=1 (CM multiplier) then CM(ci) = y, WM(ci) = 1, $1 \leq y \leq 3$
  If x=2 (WM multiplier) then CM(ci)=1, WM(ci) = y, $1 \leq y \leq 3$

Output

For each problem a line with the maximum score the robot could achieve.

Sample input

```
2
{'A': 1, 'C': 3, 'B': 3, 'E': 1, 'D': 2, 'G': 2, 'F': 4, 'I': 1, 'H': 4, 'K': 5, 'J':
8, 'M': 3, 'L': 1, 'O': 1, 'N': 1, 'Q': 5, 'P': 3, 'S': 1, 'R': 1, 'U': 1, 'T': 1,
'W': 4, 'V': 4, 'Y': 4, 'X': 8, 'Z': 10}
9
2
2
B11 B11
I11 P11
{'A': 1, 'C': 3, 'B': 3, 'E': 1, 'D': 2, 'G': 2, 'F': 4, 'I': 1, 'H': 4, 'K': 5, 'J':
8, 'M': 3, 'L': 1, 'O': 1, 'N': 1, 'Q': 5, 'P': 3, 'S': 1, 'R': 1, 'U': 1, 'T': 1,
'W': 4, 'V': 4, 'Y': 4, 'X': 8, 'Z': 10}
10
2
2
B11 B11
I11 P11
```

Sample output

```
16
22
```

Submit & test your code

To test and submit code we provide a set of tools to help you. Download contest tools if you haven't already done that. You will then be able to test and submit your solution to this challenge with the challenge token.

```
Challenge token: gt79x_o_mZqGBngfQrwS
```

To test your program

```
./test_challenge gt79x_o_mZqGBngfQrwS path/program
```

A nice output will tell you if your program got the right solution or not. You can try as many times as you need.

To submit your program to the challenge

```
./submit_challenge gt79x_o_mZqGBngfQrwS path/source_pkg.tgz path/program
```

Note that you first need to solve the test phase before submitting the code. During the submit phase, in some problems, we might give your program harder questions, so try to make your program failsafe.

Important: In this phase, you must provide the source code used to solve the challenge and, if necessary, a brief explanation of how you solved it.

Remember **you can only submit once!** Once your solution is submitted you won't be able to amend it to fix issues or make it faster, so please be sure your solution is finished before submitting it.

If you have any doubts, please check the info section.

Go ahead

I'm done! :)

Once you have submitted your code, hit refresh and continue to next challenge.

I'm stuck! :(

Be sure you follow the Tuenti Engineering twitter for updates and possible hints during the contest.

If this challenge is too hard and you are blocked, you will be able to skip it after two hours. Note that **you won't be able to complete it later**, and you have a limited number of challenges to skip.

Finally, if you run out of skips but are still really stuck with one problem, you will be able to skip it after 24 hours.

Challenge status:

Test case	Not done
Solution submitted	Not done
Skip	Skip this challenge :((You have 4 skips)

Refresh status

Tweet about this! #TuentiChallenge3

Share Follow @Tuentieng