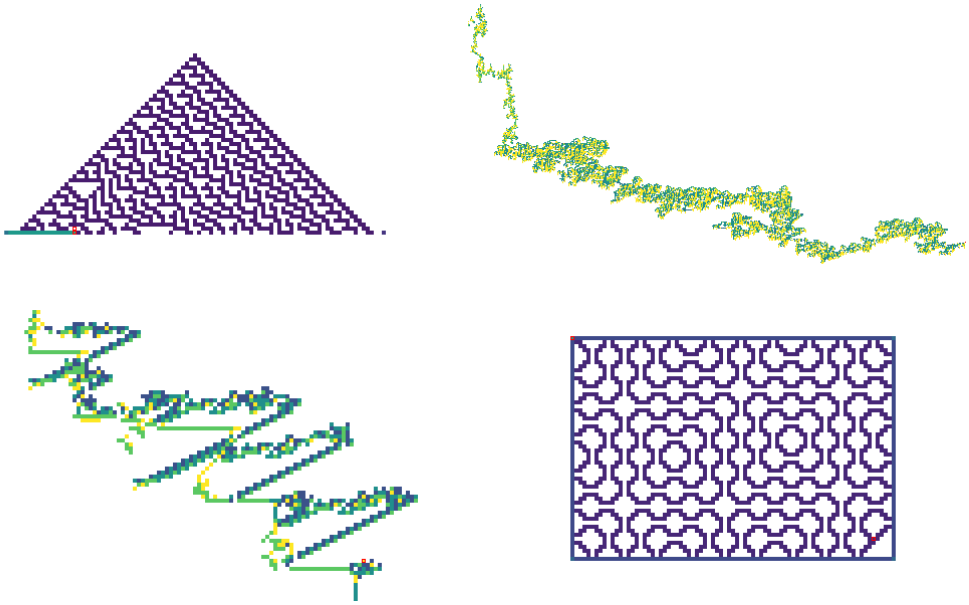
Dungeon Journey

2024 CMIMC Programming Contest New Language Round

1 Introduction

The twins Alice and Bob have found themselves trapped in a dungeon! The dungeon consists of an infinite two-dimensional square grid of rooms, and within each room there are a number of candles. Each day, the magical powers of the dungeon transports Alice and Bob an adjacent room of their choice. and erases their memory of what happened in the day. Luckily, the twins have telepathy skills which allow them to communicate with each other, regardless of their location in the dungeon. Using the candles and their telepathy skills, can the twins find the exit of the dungeon and escape?



2 Language Specifications

2.1 Overview

The dungeon is a two-dimensional square grid of rooms. Each room is indexed by a pair of integers $(x, y) \in \mathbb{Z}^2$. To be consistent with coordinate systems typically used in programming, moving up from the origin leads to the room $(0, -1)$ and down to the room $(0, 1)$. Of course, left is still $(-1, 0)$ and right is still $(1, 0)$. Within each room there is some number of indistinguishable candles that Alice and Bob may light up.

A Dungeon Journey program consists of a transition table with two inputs and four outputs. The inputs are two numbers, corresponding to the number of lit candles Alice and Bob see in their rooms. The outputs are two numbers and two directions, corresponding to what number of candles Alice and Bob wish to light in their current rooms. and to which room they want to move in next. The direction may be up/down/left/right/stay in current room.

2.2 Execution

The execution of a program begins with Alice and Bob in the room indexed by $(0, 0)$. The input to the program is written as finitely many lit candles that may appear in any room.

On each day, Alice and Bob wake up with no memory of their experiences in the dungeon. They read the number of lit candles, and write the number of candles to light. If they coincidentally are in the same room, Alice's choice is written. At the end of the day, they are simultaneously transported to the room they indicated.

The program halts when a day passes where there is Alice and Bob indicates to stay in their current rooms and the numbers on lit candles in their current rooms remain the same. The output of the program consists of the state of all candles in all rooms of the dungeon.

2.3 Coding

A Dungeon Journey program is formatted as a text file, where each line may be empty, a comment (start the line with the percentage symbol %), or a transition rule. A transition rule contains four non-negative numbers and two directions separated by spaces or commas, and possibly a comment at the end of the line (start the comment with the percentage symbol %).

The four numbers and two directions indicate that if Alice and Bob sees the first two numbers, they should write the next two numbers and move in the two directions. An output number may instead be an underscore _ which means to write the same output number as the input number. The direction is written as U/D/L/R/_, where the underscore means to stay in the current room.

It is worth noting that your transition table need not be total, that is, it need not address all possible inputs. However, if during execution you find yourself reading an input not adressed, then your program fails.

2.4 Examples

As an example, suppose that we have desire a program where we want to move Bob as far right as possible, under the condition that Bob eventually stops moving. We assume that all candles start unlit. Given three available candles in each room, a naive choice is to have Alice count up to 3 at the origin while having Bob move right. This is an example code.

```
0 0  1 _ _ R
1 0  2 _ _ R
2 0  3 _ _ R
3 0  _ 3 _ _
3 3  _ _ _ _ % Halt
```

However, we can do better. Consider the following example, where Alice counts up to 3 three times. After Alice counts to 3 at the origin, Alice will move up one room and count in room $(0, 1)$, and move back to the origin. In these two steps, Bob stops moving and the number of candles in Bob's current room is used to help keep track of where Alice should go.

```
% Inner loop
0,0 1,_,_,R
1,0 2,_,_,R
2,0 3,_,_,R
3,0 0,1,U,_ % Alice move to (0,-1), Bob marks 1
% Outer loop, Alice increment value at (0,-1), Bob clears mark
0,1 1,0,D,_
1,1 2,0,D,_
2,1 3,0,D,_
3,1 0,2,D,_ % Prepare to end, Bob marks 2
% End
0,2 0,2,_,_
```

Finally, here is another program that uses four candles, but illustrates a different important idea. It may be worthwhile coming back to this example later if you get stuck on task 3 or task 6.

```
% Set up
0,0 1,1,L,_
0,1 1,2,L,_
0,2 1,3,L,_
0,3 1,4,_,R
% Main body
1,0 2,_,_,R
2,0 3,_,_,R
3,0 0,_,R,R
% Stop
4,0 4,4,_,_
4,4 4,4,_,_
```

2.5 Testing

Problems will consist of an input and output. At the beginning of the program execution, all of the candles are unlit, except for those specified in the input. Your program should take this as input, execute, and halt. Upon halting, the number of candles lit should match the specified output in the rooms specified. In other rooms, the candles do not matter.

You are provided two programs, `dungeon-visual.py` and `dungeon-test.py`. To execute either program, you should run `python program.py -c mycode.txt -t mytest.json`. The format of test files are fairly self explanatory: each file contains a list of test cases, each of which has a list of input and output conditions in the form $[x, y, z]$ which implies the condition that room (x, y) has z candles lit. For both the visualizer and tester, the test file argument is optional. If not supplied, the program runs with an empty input, as needed in the examples above or some of the bonus tasks.

2.6 Logistics

Each normal problem is worth 10 points. You should also aim to reduce the size of the set of possible number of candles you light, hereinafter referred to size of program. This number is statically determined by your code. (For example, if you always light $0, 10$, or 20 candles, your size will be 3, not 20 or 21. This is for convenience of your coding)

Given N successful submissions, all successful submissions with size ranked at most $\lceil 0.5N \rceil$ smallest will be awarded 3 extra points, and all successful submissions size ranked at most $\lceil 0.2N \rceil$ smallest will be awarded another extra 3 points.

Your program will be capped to lighting at most 99 candles in any room. to exploring rooms (x, y) with $\max(|x|, |y|) \leq 100$. and at most 1,000,000 steps of execution. The test cases are designed so that you should be able to fit comfortably within these constrains.

CLARIFICATION: Your program should be theoretically correct on any appropriate input (such as x is any natural number), possibly arbitrarily large, and run reasonably efficient. However for testing you will only be tested up to the above constrains. Input and output conditions will all be contained in $\max(|x|, |y|) \leq 80$, and the extra padding is for you to perform computation. The idea is that for a theoretically correct on infinite grid program, you should not need to worry about the constrains.

Bonus problems are graded separately, and may have different program constrains. The points from the bonus problems will not be counted towards the overall contest ranking, but will be counted towards the new language round ranking.

3 Problems

For this test, natural numbers are strictly positive. Note that problems in each subsection are ordered by roughly increasing difficulty, but difficulty varies arbitrarily between subsections. You are intended and encouraged to move between subsections. Good luck!

3.1 Dungeon escape

You found yourself in a mysterious dungeon, and naturally you wish to escape the dungeon. Within the dungeon there unique room which leads to the exit, and that room is marked with one light candle. Find that room, and light a second candle to open the exit.

Task 1. Dungeon escape 1

Input: $(x, 0) = 1$ where x is a natural number
Output: $(x, 0) = 2$

Task 2. Dungeon escape 2

Input: $(x, y) = 1$ where x is a natural number, y is an integer with $0 \leq y \leq 3$
Output: $(x, y) = 2$

Task 3. Dungeon escape 3

Input: $(x, y) = 1$ where x is a natural number, y is an integer with $0 \leq 3y \leq x$
Output: $(x, y) = 2$

Task 4. Dungeon escape 4

Input: $(x, y) = 1$ where x is a natural number, y is an integer with $-x \leq y \leq x$
Output: $(x, y) = 2$

Task 5. Dungeon escape 5

Input: $(x, y) = 1$ where x, y are integers not both zero
Output: $(x, y) = 2$

3.2 Arithmetic

As you reached the exit, you discover that a password is required to open the exit. Luckily, you found that a series of equations have been left behind that can be used to compute the password. If you can compute these equations, perhaps you'll have the password?

Task 6. Doubling

Input: $(x, 0) = 1$ where x is some natural number
Output: $(2x, 0) = 1$, $(2x + 1, 0) = 0$

Task 7. Addition

Input: $(x, 0) = 1$ and $(y, 1) = 1$ where x, y are some natural numbers
Output: $(z, 2) = 1$, $(z + 1, 2) = 0$ where $z = x + y$

Task 8. Multiplication

Input: $(x, 0) = 1$ and $(y, 1) = 1$ where x, y are some natural numbers
Output: $(z, 2) = 1$, $(z + 1, 2) = 0$ where $z = x \times y$

Note that in these tasks, the first output condition is the relevant output. The second output condition is to prevent you from "cheating" by, say, writing 1's on the entire row.

3.3 Lists

Oh no! You have been caught by the dungeon guardian as you were cracking the password. Impressed by your cleverness, the dungeon guardian has asked you to do do some chores. Perhaps you will be allowed to leave if you can complete these tasks.

Task 9. Max

Input:	$(0, n+1) = 1$ $(x_m, m) = 1$ for all $1 \le m \le n$	where $n, x_1, \ldots, x_n$ are natural numbers
Output:	$(y, n+1) = 1$, $(y+1, n+1) = 0$	where $y = \max(x_1, \ldots, x_n)$

Task 10. Mode

Input:	$(0, n+1) = 1$, $(x_m, m) = 1$ for all $1 \le m \le n$	where $n, x_1, \ldots, x_n$ are natural numbers
Output:	$(y, n+1) = 1$, $(y+1, n+1) = 0$	where $y = \text{mode}(x_1, \ldots, x_n)$ is unique

3.4 Bonus: Draw

Throughout your journey in the dungeon, you have (hopefully) grown to appreciate the magical powers of the dungeon. Before you leave the dungeon, you wish to leave a trace of your journey for future adventurers to admire.

Bonus Task 1. Draw 1

Input:	Up to 5 non-zero inputs of your choice
Output:	Your drawing
Constrains:	Your program size must be at most 5

Bonus Task 2. Draw 2

Input:	Up to 10 non-zero inputs of your choice
Output:	Your drawing
Constrains:	Your program size must be at most 10

Bonus Task 3. Draw 3

Input:	Up to 20 non-zero inputs of your choice
Output:	Your drawing
Constrains:	Your program size must be at most 20

In this task, you are to start from a dungeon with your choice of input and draw whatever you want on the candles of the dungeon. For examples you can look at the title page (which in clockwise order from top left, satisfies tasks 3, 1, 2, 1, respectively).

If you so desire, you may increase the bounds of your program to exploring rooms (x, y) with $\max(|x|, |y|) \le 1000$. Your program does not need to halt. You may record the state of the dungeons during any point in the execution of your program.

Your drawing will be entered in a contest where all contestants will vote for the most interesting drawings. If you submit to any of the three tasks, you will receive 10 points. Within each task, the top 5 submissions will each earn 10 points.

Your deliverables should include, for each category you wish to submit to, your code, your initial input as a test file, your drawing, (optional) a short description of your program, and

the number of steps executed by the code when the drawing is captured. Your drawing may be image printed by using the provided `print_dungeon` function, or you may store the numpy array corresponding to the dungeon, and perform your own simple post-processing.

3.5 Bonus: Scavenger Hunt

In this bonus section, you are searching for exotic programs with interesting behavior. Your programs should all have size no greater than 5. These programs will run on empty input, exploring rooms (x, y) with $\max(|x|, |y|) \leq 1000$. You should change the `RADIUS` parameter in the testing programs to 1000, and run all programs until it halts or hits one of the limits above.

Bonus Task 4. Beaver

Find a program that:	lits at least 100 rooms before looping
Constrains:	Your program size must be at most 5

We say that a program loops if both the state of the dungeon and the location of Alice and Bob are eventually periodic. Note that a program halting is equivalent to looping with a period of 1. Your program should clearly be looping before 100,000 steps, and maintain that at least 100 rooms have non-zero lit candles throughout the loop.

Bonus Task 5. Methuselah

Find a program that:	eventually cycles with a period of at least 5000
Constrains:	Your program size must be at most 5

We say that a program eventually cycles if the number of lit candles Alice and Bob see are eventually periodic. This does not mean that the position of Alice and Bob loops. It is hard to argue formally that a given program does eventually cycle forever. A sufficient condition will be that Alice and Bob are drawing clearly repetitive patterns and they both head off to infinity in different directions, and for our purposes we will visually check this.

Bonus Task 6. Arrow

Find a program that:	Alice and Bob builds an arrow
Constrains:	Your program size must be at most 5

An arrow is a program where Alice and Bob's each builds a repetitive pattern (like an eventually periodic program), but they head in the same direction, their trajectories bounces off each other infinitely often, and the spacing of their intersections get further and further apart. Once again, it is hard to even define this formally, so we will inspect your program visually for this.

Bonus Task 7. Double pyramid

Find a program that:	both Alice and Bob builds an infinite pyramid
Constrains:	Your program size must be at most 5

Look back at tasks 3 and 4. It is likely that in your implementation, one of Alice and Bob "counts", while the other moves in a triangular shape to cover all the area specified by the task. In this bonus task, you should find a program where Alice and Bob simultaneously moves in a triangular shape, creating double pyramids.

Bonus Task 8. Chaos

Find a program that:	exhibits no clearly predictable pattern
Constrains:	Your program size must be at most 5

Most infinite programs in dungeon journey eventually cycles. Those that do not often contain a pyramid or arrow-like pattern, where the trajectory of Alice and Bob still has some local

repeating patterns. In this task, you should find a program that does not exhibit any clear patterns, even locally. Again, in general you do not know that a program will stay chaotic forever, but all we will check is the duration before the program hits $\max(|x|, |y|) > 1000$ or 1,000,000 steps.

You may find examples of all of the above cases in the `dungeon-journey-scav-examples` handout. Finally, if you find any other interesting patterns, especially predictable patterns that are not listed above (possible examples may include drawing a parabola or spiral), please submit them! Bonus points may be awarded.

Your deliverables should include, for each task, your code, the number of steps taken by the program, and the image printed by using the provided `print_dungeon` function in `dungeon-test` after the program either halts, hits the boundary of the dungeon, or uses up 1,000,000 steps (You should not stop the program earlier). Each task will be judged manually by the contest organizers, and each successful task awarded 2 points.