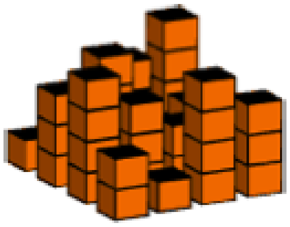
Computational Problem Solving CSCI-603
Skyscrapers Lab 4

08/17/2022

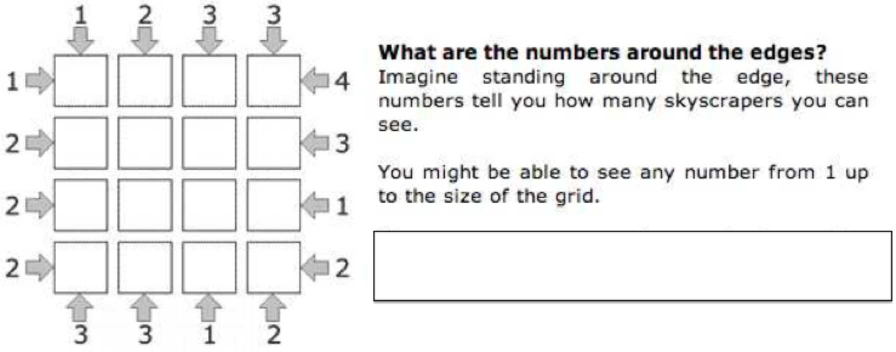


1 Introduction

Imagine a city block of skyscrapers whose area is surrounded by streets. Skyscrapers of various heights are evenly distributed into the rows and columns of the block. The goal is to organize the skyscrapers into a valid configuration that meets all the criteria for the puzzle.

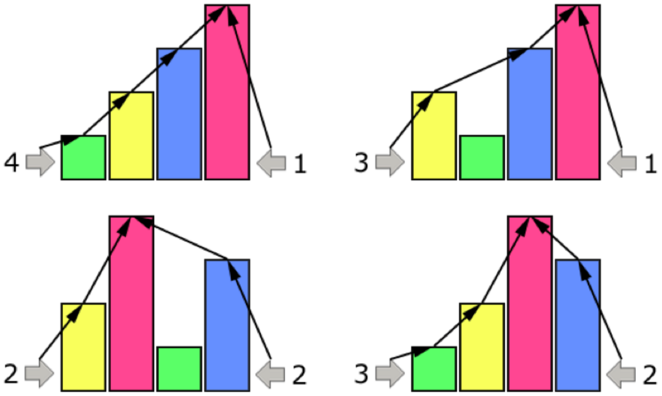
The rules for the puzzle are as follows:

i Complete the grid such that every row and column contains the numbers 1 to the size of the grid. Those numbers represent the height of the skyscrapers.
ii Each row and column contains each number only once.
iii The rules around the outside tell you how many skyscrapers you can see.
iv You can't see a shorter skyscraper behind a taller one.

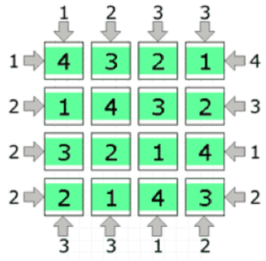


What are the numbers around the edges?
Imagine standing around the edge, these numbers tell you how many skyscrapers you can see.

You might be able to see any number from 1 up to the size of the grid.



Given the rules, here is the solution to the following puzzle.



	1	2	3	3	
1	4	3	2	1	4
2	1	4	3	2	3
2	3	2	1	4	1
2	2	1	4	3	2
	3	3	1	2	

2 Implementation

You will implement a program, `skyscrapers.py`, which reads in a filled skyscraper puzzle and checks whether the puzzle is correctly solved according to the rules.

2.1 Command Line

The program should execute on the command line as:

```
$ python3 skyscrapers.py filename
```

Where filename is the name of the text file containing the grid and clues as one row per line, with single integers 0-9, separated by spaces. If the filename is not provided, or there are too many arguments, the program should display the following message to standard output and exit:

```
Usage: python3 skyscrapers.py filename
```

If the filename is provided, it is guaranteed to be correctly formatted.

3 Input and Output files

You have been provided with some test files to help you verify the correctness of your implementation. Download those files from here. Here is a brief description of what is provided to you:

1. **data**: A folder containing different solved skyscrapers puzzles.
2. **output**: A folder containing the expected output when the program run with the **data** files.

Notice that developing your program to work with these test files is not enough; you should create your own test files to verify it thoroughly.

3.1 Input Specification

The format of the file is explained with comments "//" at the end of each line (the comments are not in the actual file).

```
4       // the square dimension of the block
1 2 3 3 // the clues at north looking south values, left to right
4 3 1 2 // the clues at east looking west values, top to bottom
3 3 1 2 // the clues at south looking north values, left to right
1 2 2 2 // the clues at west looking east values, top to bottom
4 3 2 1 // row 1 values
1 4 3 2 // row 2 values
3 2 1 4 // row 3 values
2 1 4 3 // row 4 values
        // newline (nothing should be read beyond here)
...     // comments about this test file
```

3.2 Program Execution

These are the steps the program should follow.

1. Check the command line arguments and display/exit if the number of arguments is incorrect.
2. Read the file, save it, and display the puzzle along with the clues.
3. Verify whether the puzzle has been solved correctly. If the puzzle is correct, print `The puzzle is valid!`. Otherwise, display a description of every rule violated.

If a puzzle contains several errors, the program must display each error in a separate line on the standard output.

3.3 Output Specifications

All output should happen to standard output. Refer to the sample runs provided on the `output` folder for examples of all the output messages.

4 Grading

The assignment grade is based on these factors:

- 20%: results of problem-solving
- 10%: Design
- 60%: Functionality
  - 10%: Proper selection and use of data structures
  - 5%: File reading
  - 30%: Detect correctly invalid puzzles and display the violated rule
  - 15%: Detect correctly valid puzzles
- 10%: Code Style and Documentation

5 Submission

Create a ZIP file named **lab4.zip** that contains your `skyscrapers.py` file. Submit the ZIP file to the MyCourses assignment before the due date (if you submit another format, such as 7-Zip, WinRAR, or Tar, you will not receive credit for this lab).