Problem Solving Workshop #42 Tech Interviews and Competitive Programming Meetup
January 26, 2019 https://www.meetup.com/tech-interviews-and-competitive-programming/
Instructor: Eugene Yarovoi (can be contacted through the group Meetup page above under Organizers)

More practice questions: leetcode.com, geeksforgeeks.org
Books: Elements of Programming Interviews, Cracking the Coding Interview
Have questions you want answered? Contact the instructor on Meetup, or ask on Quora. You can post questions and follow the instructor and other people who write about algorithms.

Try to find optimized solutions, and provide a time and space complexity analysis with every solution.

Problem #1, "Car Rental" (Easy)

A car rental company is going to rent out a car, but there are multiple requests to rent it. Each person has a given (start time, end time) pair representing when they want to rent the car, and the car can only be rented to one person at a time. Given a list of rental requests, determine which people you should rent the car to, in order to maximimze the number of people you rent it to. (Return any solution if multiple are tied for optimal.)

Example Input: [(s = 4, e= 14), (s = 6, e = 8), (s = 9, e = 15), (s = 13, e = 17), (s = 19, e = 21)]
Output: [(s = 6, e = 8), (s = 9, e = 15), (s = 19, e = 21)]
Explanation: You can accept at most 3 of these rental requests (there's no way to do more requests without them conflicting). The above output is a way to honor 3 requests.

Follow-up **(Medium)**: what if each request has a different payoff? You're given (start, end, payoff) triplets. Maximize your total payoff.

Problem #2, "Labyrinth" (Medium)

You're trapped inside an MxN grid G, starting out in the upper left corner cell of the grid. To escape, you have to reach the lower right corner cell. Every time you move to an adjacent cell (you can move to any adjcent cell in the 4 cardinal directions), a ghoul appears and asks you a programming question. Each cell has a positive integer value (i.e. G is a 2D array and G[x][y] has some value for each in-bounds x, y), representing the difficulty of the programming problem you'll be asked. You are given the grid ahead of time to study, before you have to make any moves.

To get through, you need to solve each ghoul's problem. You reason that the smart way to approach it would be not necessarily to find the shortest path through the grid, but find the path that ***minimizes*** the difficulty of the ***hardest*** problem you'll encounter along the way. After all, it's easy to solve many easy problems, but even one difficult problem can imperil you. Find the best path throuigh the grid accoridng to this criterion.

The starting and ending cells don't have any ghouls in them (no problem to solve).

Example Input (provided as 2D array): G =

Start	3	8
4	7	8
6	3	End

Output: [(0, 0), (0, 1), (0, 2), (1, 2), (2, 2)]

Explanation: The optimal path is the output, in order from start to end, as (x, y) coordinate pairs, with left being 0 and top being 0 (image-coordinate system). The values on this path are 4, 6, 3 in that order. The most difficult problem on this path is the one with difficulty 6, at (0, 2). It's not possible to avoid a problem of at least that difficulty, as any other path encounters a difficulty of 7 or 8.

(a) Return any path if there's a tie.
(b) If multiple optimal paths are tied according to the least-max-difficulty criterion, return the shortest of them.

Problem #3, "Printing Directories"

Consider a command like `ls` in a shell. Suppose we want it to print all the files in a directory in multiple columns, so that the columns are aligned. For example, if we have the filenames A, Boo, Bzarr, and Gamma, they might get printed like this:

```
A    Bzarr
Boo  Gamma
```

The font is always monospace (each character takes up the same width), the width of a column is always the length of the longest string in that column plus 2 characters for padding (assume there's padding even for the rightmost column for simplicity), and the files are always printed in alphabetical order, going first from top to bottom, and then from left to right. If the N files are printed using R rows, then there must be ceiling (N / R) columns. The last column may have some empty spaces. For example, if we have files named A, B, C, D and decide to use 3 rows, we will print:

```
A  D
B
C
```

In a terminal window, the maximum width of a line is typically limited by the size of the window. We don't want to exceed the line width limit. We also want to print the directory files as compactly as possible -- that is, using the smallest possible number of rows.

Given a line width limit L, and a list of files in alphabetical order (they must also be printed in that order), what is the minimum number of rows you need to print all the files, aligned in columns as shown earlier, without exceeding the line width limit?
(To make the problem more interesting, don't assume L is a small number).

Complexity guidance: [Easy] O(N^2), **[Hard]** O(N log N)

Follow-up: What if you instead have to print in left-to-right, then top-to-bottom? E.g.

```
A   B   C
D
```

Complexity Goals

1. O(N log N) where N is number of requests, for both the main problem and the follow-up.
2. O(MN log(MN)), for both the original and the follow-up.
3. Stated in problem.