159.271 Computational Thinking
Assignment 2

Semester 1, 2020

Your job for this assignment is to finish implementing a game we shall call "Collapse". Submit both your code and your answers to non-code questions via stream.

For the code questions (b) and (e), you only need to modify and submit `optimizer.py`.

Collapse

The game works as follows: The player starts in the center of a collapsing building, which consists of a quadratic grid of $(2n+1) \times (2n+1)$ rooms. Each room contains 0-9 units of supplies. With each move the player can go up, down, left or right and collect the supplies in the room entered. The entire row or column of rooms he just left then collapses behind him and can no longer be entered. After $2n$ moves no further moves can be taken and the game ends. If the player has collected enough supplies to survive till the rescue team gets him after D days, he wins. If not he looses.

A basic implementation of the game is provided on the course website. Run with the command:

```
> python collapse.py
```

To perform a speed test of your implementation, run

```
> python optimizer.py
```

Note: You can also specify the board size and easy/hard mode. E.g.

```
> python collapse.py 3 hard
```

will run the game with $n = 3$ using separate food and water supplies.

Tasks [20 points]

Currently the number of supplies needed to win the game is only based on n, and depending on how supplies are distributed (this is done at random) it may be very easy or impossible to win.

Your job is to calculate the maximum number of supplies that could be collected, and set the victory requirement (number of days till rescue) to this number. You will only need to edit `optimizer.py` for that.

(a) Show that a greedy strategy (starting in the center or at one of the corners) will not work. For this, give a 3×3 example with a unique optimal solution which includes neither the highest-valued side-room nor the highest-valued corner room. Use each number from 1 to 8 once (center is empty). [1 point]

To illustrate, in the example to the right, the unique optimal solution is 7-8 which includes both the highest-valued side-room (7) and the highest-valued corner room (8). Here a greedy algorithm will find the optimal solution. You need to create a different example where greedy fails.

1	2	3
4	@	5
6	7	8

(b) Implement an algorithm which calculates the maximum number of supplies collectable (as described above). This algorithm should be efficient (≤ 0.1 sec) for $n = 20$. [8 points]
Hint: Use dynamic programming techniques.

(c) Analyze the worst case time complexity of your `max_food` implementation, as a function of n. [1 point]
Reminder: There are $(2n+1) \times (2n+1)$ rooms. Not n.

To make the game more challenging, supplies are now separated into food and water. Each rooms contains either 0-9 units of food or 0-9 units of water. To survive, a player requires both food and water units equal to the number D of days till rescue.

(d) Consider the following greedy strategy for dealing with (food, water) pairs instead of just supplies:

> For each room simply store the partial solution for which min(food, water) is maximal. E.g., given the choice of (7,5) and (6,6), only the latter is kept.

Show that this strategy is not optimal, i.e., does not maximize min(food, water). [1 point]

Hint: A counter-example requires $n \geq 2$, but only one quadrant is relevant (rest can be empty). An example where the strategy does work is shown on the right:

		3W	2W	5F
		4W	5F	4W
		@	2F	4F

(e) Implement an algorithm which calculates the maximum number of food and water collectable (i.e., maximize min(food, water)). This algorithm should be efficient ($\leq$ 1 sec) for $n = 20$. [8 points]
Hint: When considering intermediate results, (food,water) pairs for which better pairs exist can be ignored. E.g. (7,5) is better than (7,4) but incomparable to (6,6).

(f) Analyze the worst case time complexity of your advanced algorithm, as a function of n. [1 point]

Speed Race

The two fastest (correct, $\leq$ 60 sec) solutions of the advanced version for a size of $n = 100$ will receive a Steam coupon for the *Dreamgate* game. Winners and their times will be published on Stream for bragging rights.