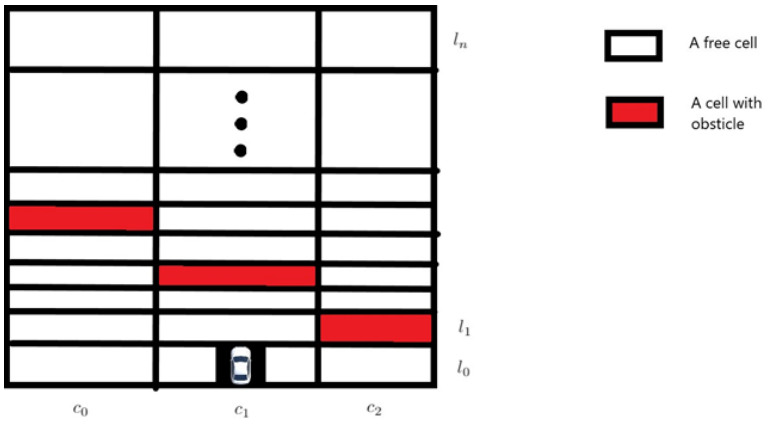
Homework 3 is due April 18 (Thursday), 23:30.

Traffic lane problem Consider a roadway with three traffic lanes, each consisting of cells as shown in the figure below.



The lanes are denoted by columns c_1, c_2 and c_3. Each lane is marked with levels l_1, l_2, ..., l_n, so the cells are denoted by pairs of lanes and levels. For instance, (l_1, c_2) denotes the cell at level 1 of the traffic lane 2. The roadway has some obstacles; the cells covered by these obstacles are colored as red in the figure.

Initially, there is a car at cell $(0, 1)$ of this roadway. The car can move forward with the goal of reaching a cell at level l_n, obeying the following rules:

Rule 1 The car can not move through the cells covered by obstacles.

Rule 2 The car has to move forward to a neighboring cell at the next level. For instance, if the car is located at a cell (l_i, c_0) then it has to move straight to (l_{i+1}, c_0) or diagonally to (l_{i+1}, c_1).

Therefore, the car cannot move between the cells at the same level, and it cannot go back to a cell at the previous level.

Rule 3 The car cannot jump over cells. Therefore, it cannot go from c_0 to c_2 or from c_2 to c_0 in one step. It cannot jump from level l_i to l_j where $j > i + 1$ or $j < i - 1$.

Note that, according to these rules, the car can move to one of the three cells at the next level only if it is at c_1.

The traffic lane problem aims to find a route for the car to reach one of the cells at the last level, so as to minimize the number of traffic lane changes.

Submit PDF Write a report including the following:

(a) Recursive formulation of the traffic lane problem: Identify the subproblems, observe the optimal substructure property and the overlapping computations, and then define the problem recursively.

(b) Pseudocode of a naive recursive algorithm based on your recursive formulation, and its complexity analysis (i.e., the asymptotic time and space complexity).

(c) Pseudocode of a recursive algorithm that builds solutions to your recurrence from top down with memoization, and its complexity analysis (i.e., the asymptotic time and space complexity).

(d) Pseudocode of an iterative algorithm that builds solutions to your recurrence from bottom up, and its complexity analysis (i.e., the asymptotic time and space complexity).

(e) Experimental evaluations of these three algorithms: plot the results in a graph, and discuss the results (e.g., are they expected or surprising? why?)

Submit Python Code, and Benchmarks

(a) Implement in Python the three algorithms above, designed to solve the traffic lane problem.

The roadway will be presented as an input matrix of size $n \times 3$. The entry at the i'th row and j'th column of the matrix is 0, if the cell (l_i, c_j) is a free cell (i.e., not occupied by an obstacle); otherwise, it is 1.

(b) Create a benchmark suite to test the correctness of your Python programs: take into account the functional testing methods (e.g., white box and lack box testing) while constructing the instances, and test your programs with these instances.

(c) Create a benchmark suite to test the performance of your Python programs: construct instances of different sizes (e.g., relative to the number of levels), evaluate the performance of your programs in terms of computation times, and plot the results within a graph.

Demos Demonstrate that your Python programs correctly compute solutions for your benchmark instances, and for the instances that will be provided by us. Demo day will be announced.