CS 202 Assignment 2 (2021 Term 2)

1. You have gone rock climbing and want to reach the top following an easiest path. Looking at the rock wall you can form a mental model of the easiness of the next step in climbing up the wall. You divide wall into a grid of blocks as below:

From each block you can go up left, up straight, or up right. There is a cost of each cell that you have estimated below. You are interested in the lowest overall cost to go from the bottom row (starting any place in bottom row) to the top row (anywhere in top row). Below an example is shown in bold

2	8	9	**5**	8
4	4	6	**2**	3
5	7	5	6	**1**
3	2	5	**4**	8

This question asks you to solve this problem using dynamic programming. Index the rows from 1 to n, with row 1 as bottom row and row n as top row. Similarly, column 1 is left column and column m is the right column (note: $m \neq n$ in general). **Define** $L(i,j)$ to be the cost of the cheapest (least dangerous) path from the bottom to the cell (i,j).

   a. (0.5 mark) Remember you want to solve lowest overall cost from the bottom row to the top row. State the problem solution value in terms of $L(i,j)$.

   b. (1 marks) Let the cost of each cell be $C(i,j)$. Write a recursive formula for $L(i,j)$. Hint: it might be helpful if you add two columns 0 and m+1 filled with ∞ and if you add a row indexed 0 with all 0s for columns 1 to m.

   c. (2.5 marks) Code up your solution in python producing both the lowest overall cost and the path from bottom row to top. We have provided you one q1.py file with the two functions (find_cost, get_opt_path) that you should implement. The code must use iterative dynamic programming.

      Important instructions

      i. Please ensure that you implement your codes with the 2 functions given in q1.py.
      ii. You can introduce more helper methods within q1.py (optional)
      iii. Please ensure your code can run with the given q1_tester.py to avoid penalty.
      iv. Please ensure that the output of your q1.py conforms to the format of the expected result too to avoid penalty.

v. This part will be auto-graded with other test cases upon submission.

vi. Python 3.9 will be used to run your code and no external libraries are allowed.

vii. Please zip up your q1.py file for submission and name the zip file according to the following format. (e.g. if your email address is john.lee.2019@sis.smu.edu.sg -> your zip file should be named as john.lee.2019.zip

2. You have started a business selling rolls of paper. The paper rolls come out of your machine as a single roll of length n. You have an option of cutting the rolls into smaller ones. Rolls of different length sell for different prices; of course, you want to maximize your revenue. You want to solve this problem using dynamic programming. For example, here is table of length and price

Length	1	2	3	4	5	6	7	8	9	10
Price (p_i)	1	5	7	9	10	17	17	20	23	30

a. (0.5 marks) Write a recursive formulation for the optimal solution of the problem above. Follow the convention that when you cut the roll the first piece is the left side of the roll and then the rest of roll to the right is further cut up as required. Use the notation R_n to denote the optimal revenue from roll of length n

b. (1 mark) Write an iterative pseudo code implementing the above recursive formulation.

c. (1.5 marks) Now, someone imposes an additional constraint that limits the number of rolls of each size that can be produced. For example, as given below

Length	1	2	3	4	5	6	7	8	9	10
Price	1	5	7	9	10	17	17	20	23	30
Limit	2	1	2	2	2	1	1	1	1	1

Explain (in less than 30 words) why the same recursive formulation that you wrote in part (a) does not work anymore. Be precise and write the precise reason

d. (1.5 marks) As the last problem became too restrictive, you are now allowed to cut the roll into at most K parts overall (no limit per size of roll). Write a recursive formulation for the optimal solution that will be able to solve this problem.

3. There are 5 apartments A1, …, A5 and people from these apartments order food from 3 restaurants R1, …, R3. The demand from the 5 apartments are as follows (think orders/hour):
   A1: 4, A2: 9, A3: 2, A4: 10, A5: 15
   The restaurants can supply (think orders/hour)
   R1: 10, R2: 10, R3: 10
   There are further constraints on how much can each restaurant deliver to each apartment (based on availability of delivery people, distance from apartment, etc.). The constraints are as follows:
   R1 can deliver 2 to A1, 3 to A2, 4 to A3, 3 to A4, 2 to A5
   R2 can deliver 2 to A1, 3 to A2, 3 to A3, 4 to A4, 5 to A5
   R3 can deliver 1 to A1, 2 to A2, 3 to A3, 4 to A4, 5 to A5
   Based on the above information answer the following question:
   a. (1 mark) It is desired to satisfy the overall demand as much as possible. Formulate this maximum demand satisfying problem as a maximum flow problem. Clearly show the graph visually, the edge capacities, and any other detail.
   b. (0.5 marks) Which apartment's demand can never be satisfied completely?