Question 1

You are playing a video game, where you control a character in a grid with m rows and n columns. The character starts at the square in the top left corner $(1, 1)$, and must walk to the square in the bottom right corner (m, n). The character can only move one square at a time *downwards or rightwards*. Every square (i, j), other than the starting square and the ending square, contains a known number of coins $a_{i,j}$.

1.1 [10 marks] Design an algorithm which runs in $O(mn)$ time and determines the maximum number of coins that your character can accumulate by walking from $(1, 1)$ to (m, n) using a combination of downwards and rightwards moves.

Answer:

According to the topic, the character only move one square to downwards or rightwards at one time, therefore, for each square view, the character will only enter from the up side or left side, so the maximum amount of coins when the square is reached by character is the maximum amount of coins between left side and top right plus the amount of coins which contains in this square.

Suppose that any square (i, j), the max amount which character for each square is $opt(i, j)$, the coins which contains in the square is $C(i, j)$.

$$opt(i,j) = \begin{cases} max\{opt(i-1,j), opt(i,j-1)\} + C(i,j) & 1 < i \leq m \ , 1 < j \leq n \\ opt(i-1,j) + C(i,j) & j = 1 \\ opt(i,j-1) + C(i,j) & i = 1 \end{cases}$$

Therefore, to get the maximum number of coins from $(1, 1)$ to (m, n). Set a $m \times n$ two-dimensional array **opt** record the maximum amount coins in each square(i, j), set the $opt(1, 1) = 0$, the coins which contains in the square is $C(i, j)$.

Set the output loop i from 1 to m and set the input loop j from 1 to n.

- if $i = 1$, $opt(i, j) = opt(i, j - 1) + C(i, j)$.
- if $j = 1$, $opt(i, j) = opt(i - 1, j) + C(i, j)$.
- if else, $opt(i, j) = max\{opt(i - 1, j), opt(i, j - 1)\} + C(i, j)$.

When finish the Nested loop, the max number of coins from $(1, 1)$ to (m, n) is $opt(m, n)$.

The time complexity of output loop is $O(m)$ and input loop is $O(n)$, the total complexity is $O(mn)$.

1.2 **[10 marks]** After playing this game many times, you have broken the controller, and you can no longer control your character. They now walk randomly as follows:

- if there is only one possible square to move to, they move to it;
- otherwise, they move right with probability p and down with probability $1 - p$.

Note that this guarantees that the character arrives at (m, n).

Design an algorithm which runs in $O(mn)$ time and determines the *expected* number of coins that your character will accumulate by walking from $(1, 1)$ to (m, n) according to the random process above.

Recall that for a discrete random variable X which attains values $x_1, \ldots, x_n$ with probabilities $p_1, \ldots, p_n$, the *expected value* of X is defined as

$$\mathbb{E}(X) = \sum_{i=1}^{n} p_i x_i.$$

Answer:

According to the topic, now the move is random, the character move to rightwards with probability p and move to downwards with probability $1-p$. It is also means that any square which come from up side with probability $1-p$ and from left side with probability p.

Suppose that any square (i, j), the expected coins amount which character for each square is $\mathbb{E}(i, j)$, the coins which contains in the square is $C(i, j)$.

$$\mathbb{E}(i,j) = \begin{cases} \mathbb{E}(i-1,j) * (1-p) + \mathbb{E}(i,j-1) * p + C(i,j) & 1 < i \leq m \ , 1 < j \leq n \\ \mathbb{E}(i-1,j) + C(i,j) & j = 1 \\ \mathbb{E}(i,j-1) + C(i,j) & i = 1 \end{cases}$$

Therefore, to get the expected number of coins from $(1, 1)$ to (m, n). Set a $m \times n$ two-dimensional array $\mathbb{E}$ record the expected amount coins in each square(i, j), set the $\mathbb{E}(1, 1) = 0$, the coins which contains in the square is $C(i, j)$.

Set the output loop i from 1 to m and set the input loop j from 1 to n.

- if $i = 1$, $\mathbb{E}(i, j) = \mathbb{E}(i, j - 1) + C(i, j)$.
- if $j = 1$, $\mathbb{E}(i, j) = \mathbb{E}(i - 1, j) + C(i, j)$.
- if else, $\mathbb{E}(i, j) = \mathbb{E}(i - 1, j) * (1 - p) + \mathbb{E}(i, j - 1) * p + C(i, j)$.

When finish the Nested loop, the expected number of coins from $(1, 1)$ to (m, n) is $\mathbb{E}(m, n)$.

The time complexity of output loop is $O(m)$ and input loop is $O(n)$, the total complexity is $O(mn)$.