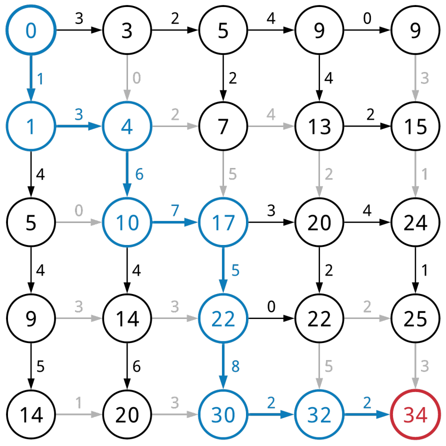
5B Find the Length of a Longest Path in a Manhattan-like Grid

Length of a Longest Path in the Manhattan Tourist Problem
Find the length of a longest path in a rectangular city.

Input: Integers n and m, an $n \times (m+1)$ matrix *Down*, and an $(n+1) \times m$ matrix *Right*.
Output: The length of a longest path from source (0,0) to sink (n,m) in the $n \times m$ rectangular grid whose edges are defined by the matrices *Down* and *Right*.



Formatting

Input: Integers n and m, followed by an $n \times (m+1)$ matrix *Down*, and an $(n+1) \times m$ matrix *Right*. The two matrices are separated by the "-" symbol.
Output: The length of a longest path from source (0,0) to sink (n,m) in the $n \times m$ rectangular grid whose edges are defined by the matrices *Down* and *Right* as an integer.

Constraints

- The values of n and m will be between 1 and 10^2.

Test Cases

Case 1

Description: The sample dataset is not actually run on your code.

Input:

```
4 4
1 0 2 4 3
4 6 5 2 1
4 4 5 2 1
5 6 8 5 3
-
3 2 4 0
3 2 4 2
0 7 3 3
3 3 0 2
1 3 2 2
```

Output:

```
34
```

Case 2

Description: This dataset checks that your code considers the first column of *Down* and the last row of *Right*. If there is some off-by-one error (possibly due to 0/1 indexing confusion) in updating the dynamic programming matrix it could be possible to miss these edges.

Input:

```
2 2
20 0 0
20 0 0
-
0 0
0 0
10 10
```

Output:

```
60
```

Case 3

Description: This dataset checks that your code considers the last column of Down and the first row of Right. If there is some off-by-one error (possibly due to 0/1 indexing confusion) in updating the dynamic programming matrix it could be possible to miss these edges. This dataset is very similar to test case 2.

Input:

```
2 2
0 0 20
0 0 20
-
10 10
0 0
0 0
```

Output:

```
60
```

Case 4

Description: This dataset checks that you are not using some sort of greedy approach to solving this problem. If the movement with the highest weight is chosen at each step the first entry of *Down* should be chosen. This will result in missing the higher total weight that just goes all the way right then down.

Input:

```
2 2
20 0 0
0 0 0
-
0 30
0 0
0 0
```

Output:

```
30
```

Case 5

Description: This dataset checks that your code can correctly parse and use inputs in which n and m are not the same. In all previous test datasets n and m were the same. If your output doesn't match the correct output make sure that your dynamic programming matrix has the dimensions $(n+1)\times(m+1)$. In previous datasets your code could get away with creating a dynamic programming matrix with dimensions $(n+1)\times(n+1)$ or $(m+1)\times(m+1)$, but implementations relying on those dimensions will fail this dataset.

Input:
```
5 3
20 5 0 10
0 5 10 0
10 10 0 15
0 20 20 25
30 10 5 30
-
0 30 15
10 20 10
10 10 20
20 25 30
15 35 40
15 10 25
```

Output:
```
175
```

Case 6

Description: This dataset checks that your code can correctly parse and use inputs in which n and m are not the same. This dataset is similar to test dataset 5.

Input:
```
3 5
0 5 10 0 10 10
15 0 20 20 25 30
10 5 30 15 0 20
-
0 30 15 10 20
10 10 10 20 20
25 30 15 35 40
15 10 25 15 20
```

Output:
```
180
```

Case 7

Description: A larger dataset of the same size as that provided by the randomized autograder. Check input/output folders for this dataset.