Problem 18. Find the maximum total from top to bottom of the triangle below:

75
95 64
17 47 82
18 35 87 10
20 04 82 47 65
19 01 23 75 03 34
88 02 77 73 07 63 67
99 65 04 28 06 16 70 92
41 41 26 56 83 40 80 70 33
41 48 72 33 47 32 37 16 94 29
53 71 44 65 25 43 91 52 97 51 14
70 11 33 28 77 73 17 78 39 68 17 57
91 71 52 38 17 14 91 43 58 50 27 29 48
63 66 04 68 89 53 67 30 73 16 69 87 40 31
04 62 98 27 23 09 70 98 73 93 38 53 60 04 23

NOTE: As there are only 16384 routes, it is possible to solve this problem by trying every route. However, Problem 67, is the same challenge with a triangle containing one-hundred rows; it cannot be solved by brute force, and requires a clever method! ;o)

Knowledge required Dynamic Programming

Solution We will be not solving this using brute force, instead we will be thinking in terms of dynamic programming. This approach can be altogether solve problem-18 and problem-67. We start by taking the triangle input in matrix, where each row of the input is a row in the matrix, all the remaining entries which are not filled by default will be initialized to 0. Then these are the transition states.

$$tri\_sum[i][j] += \begin{cases} tri\_sum[i-1][j] & \text{if j == 0} \\ \max(tri\_sum[i-1][j-1], tri\_sum[i-1][j] & \text{otherwise} \end{cases}$$

Here `tri_sum[i][j]` means the maximum sum that we can get starting from (0,0) to (i, j). The same approach will work for problem-67. The time complexity is O(n^2)

Python Solution

```python
ROWS = COLS = 15

# initial matrix all filled with zeros
tri_sum = [[0 for j in range(COLS)] for i in range(ROWS)]

# read the file and fill the entries in tri_sum
with open('input', 'r') as f:
    r = 0
    for line in f.readlines():
        line = list(map(int, line.split()))

        for c, el in enumerate(line):
            tri_sum[r][c] = el
        r += 1

for i in range(ROWS):
    for j in range(COLS):
        if i == 0 and j == 0: continue
        if j == 0:
            tri_sum[i][j] += tri_sum[i - 1][j]
        else:
            tri_sum[i][j] += max(tri_sum[i - 1][j], tri_sum[i - 1][j - 1])

# as we can end at any position in the last
# row we need to print maximum of all those
# values
print(max(tri_sum[ROWS-1]))
```