Maximum path sum I

Problem 18 (https://projecteuler.net/problem=18)

By starting at the top of the triangle below and moving to adjacent numbers on the row below, the maximum total from top to bottom is 23.

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

That is, 3 + 7 + 4 + 9 = 23.

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)

Solution

In [1]:

```
data="""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"""

mx = [[int(y) for y in x.split()] for x in data.splitlines()]

def maxpath(m,r,c):
    if r == len(m)-1:
        return [max(m[r][c],m[r][min(c+1, len(m[r])-1)])]
    else:
        lp = maxpath(m,r+1,c)
        rp = maxpath(m,r+1,min(c+1, len(m[r+1])-1))
        return [m[r][c]] + (lp if sum(lp)>sum(rp) else rp)

print(maxpath(mx,0,0), sum(maxpath(mx,0,0)), sep='\n')
```

```
[75, 64, 82, 87, 82, 75, 73, 28, 83, 32, 91, 78, 58, 73, 93]
1074
```