advent_of_code_day 15

December 28, 2021

1 Advent of Code

[ ]:
```
# set up the environment
import numpy as np
from collections import Counter
import networkx as nx
```

1.1 Day 15: Chiton

You've almost reached the exit of the cave, but the walls are getting closer together. Your submarine can barely still fit, though; the main problem is that the walls of the cave are covered in chitons, and it would be best not to bump any of them.

The cavern is large, but has a very low ceiling, restricting your motion to two dimensions. The shape of the cavern resembles a square; a quick scan of chiton density produces a map of ***risk level*** throughout the cave (your puzzle input). For example:

1163751742
1381373672
2136511328
3694931569
7463417111
1319128137
1359912421
3125421639
1293138521
2311944581

You start in the top left position, your destination is the bottom right position, and you cannot move diagonally. The number at each position is its **risk level**; to determine the total risk of an entire path, add up the risk levels of each position you **enter** (that is, don't count the risk level of your starting position unless you enter it; leaving it adds no risk to your total).

Your goal is to find a path with the **lowest total risk**. In this example, a path with the lowest total risk is highlighted here:

1163751742
1381373672
2136511328
3694931**5**69

7463417**1**11
1319128**13**7
135991242**1**
312542163**9**
129313852**1**
231194458**1**

The total risk of this path is *40* (the starting position is never entered, so its risk is not counted).

What is the lowest total risk of any path from the top left to the bottom right?

1.2 Solution

This is a shortest path problem which is heavily based on graph theory. After a lot of reading(!) it seems that the `NetworkX` library has the required procedures. A graph can be created using the `grid_2_graph generator` which has the same dimensions as the source grid and connects each node to its nearest neighbour. The `DiGraph` type can be used to store the weighted edges for entering a node.

The hardest part will be setting the weighting for the edges; we can iterate through the edges and set the weighting of the edge to that of the destination.

Once the array is constructed `NetworkX` will return the shortest path i.e. the one with the least total weighting.

[ ]:
```
TEST_PATH = "data/chiton_risk_test.dat"
PROBLEM_PATH = "data/chiton_risk.dat"

def get_data(filename: str):
    """Create a 2 dimensional array from the source file.

    The source file consists of lines of digits, each line corresponding to one␣
 ↪row
    of the array and each digit corresponding to a column in that row.

    filename -- the path to the source file.

    returns a numpy array.
    """

    with open(filename) as file:
        risk_array =np.array([list(map(int, line)) for line in file.read().
 ↪splitlines()])

    return risk_array

def create_graph(source_data):
    rows, cols = source_data.shape
    g = nx.grid_2d_graph(rows, cols, create_using=nx.DiGraph)
```

```
    for (xs, ys), (xd, yd), data in g.edges(data=True):
        data["weight"] = source_data[xd][yd]
    return g

# verify the test data
risk_data = get_data(TEST_PATH)
height, width = risk_data.shape
risk_graph = create_graph(risk_data)
path_length = nx.shortest_path_length(risk_graph, source=(0, 0),␣
 ↪target=(height-1, width-1), weight="weight")

assert(path_length == 40)

# OK so get problem solution
risk_data = get_data(PROBLEM_PATH)
height, width = risk_data.shape
risk_graph = create_graph(risk_data)
path_length = nx.shortest_path_length(risk_graph, source=(0, 0),␣
 ↪target=(height-1, width-1), weight="weight")

print(f'Minimum basic problem path length: {path_length}')
```

```
Minimum basic problem path length: 363
```

1.3 Part Two

Now that you know how to find low-risk paths in the cave, you can try to find your way out.

The entire cave is actually **five times larger in both dimensions** than you thought; the area you originally scanned is just one tile in a 5x5 tile area that forms the full map. Your original map tile repeats to the right and downward; each time the tile repeats to the right or downward, all of its risk levels **are 1 higher** than the tile immediately up or left of it. However, risk levels above 9 wrap back around to 1. So, if your original map had some position with a risk level of 8, then that same position on each of the 25 total tiles would be as follows:

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

Each single digit above corresponds to the example position with a value of 8 on the top-left tile. Because the full map is actually five times larger in both dimensions, that position appears a total of 25 times, once in each duplicated tile, with the values shown above.

Here is the full five-times-as-large version of the first example above, with the original map in the top left corner highlighted:

1163751742274862853338597396444961841755517295286
1381373672249248478335135958944624616915573572712 6
2136511328324762243943587335415469844652657195576 3
3694931569471514267158262537826937364893714847591 4
7463417111857452822296856393331796741444281785255 5
1319128137242123924835323413594643452461575456357 2

1359912421246112353235722346434683345754579445686 5
3125421639423653274153476438526458754963756986517 4
1293138521231424963234253517434536462854564757396 5
2311944581342215569245332667135644377824675548935
2274862853338597396444961841755517295286662831639 7
2492484783351359589446246169155735727126684683823 7
3247622439435873354154698446526571955763768216687 4
4715142671582625378269373648937148475914825958612 5
8574528222968563933317967414442817852555392896366 6
2421239248353234135946434524615754563572686567468 3
2461123532357223464346833457545794456865681556797 6
4236532741534764385264587549637569865174867197628 5
2314249632342535174345364628545647573965674 5
3422155692453326671356443778246755488935786614 6
3385973964449618417555172952866628316397773942741 8
3513595894462461691557357271266846838237795794934 8
4358733541546984465265719557637682166874879327798 5
5826253782693736489371484759148259586125936169723 6
9685639333179674144428178525553928963666413917477 7
3532341359464345246157545635726865674683797678579 4
3572234643468334575457944568656815567976792667818 7
5347643852645875496375698651748671976285978218739 6
3425351743453646285456475739656758684176786979528 7
4533266713564437782467554889357866599146897761125 7
4496184175551729528666283163977739427418884153852 9
4624616915573572712668468382377957949348816815145 9
5469844652657195576376821668748793277985981438819 6
6937364893714847591482595861259361697236147271834 7
1796741444281785255539289636664139174777524128588 8
4643452461575456357268656746837976785794818789681 5
4683345754579445686568155679767926678187813778929 8
6458754963756986517486719762859782187396189329841 7
4536462854564757396567586841767869795287891816398
5644377824675548893578665991468977611257918872236 8
5517295286662831639777394274188841538529995264963 1
5735727126684683823779579493488168151459927926256 1
6571955763768216687487932779859814388196192549921 7
7148475914825958612593616972361472718347258382945 8
2817852555392896366641391747775241285888635239699 9
5754563572686567468379767857948187896815929891792 6
5794456865681556797679266781878137789298924889131 9
7569865174867197628597821873961893298417291431952 8
5647573965675868417678697952878971816398918292741 9
6755488935786659914689776112579188722368129983347 9

Equipped with the full map, you can now find a path from the top left corner to the bottom right corner with the lowest total risk:

11637517422274862853338597396444961841755517295286

138137367224924847833513595894462461691557357271 26
213651132832476224394358733541546984465265719557 63
369493156947151426715826253782693736489371484759 14
746341711185745282229685639333179674144428178525 55
131912813724212392483532341359464345246157545635 72
135991242124611235323572234643468334574579445686 5
312542163942365327415347643852645875496375698651 74
129313852123142496323425351743453646285456475739 65
231194458134221556924533266713564437782467554889 35
227486285333859739644496184175551729528666283163 97
249248478335135958944624616915573572712668468382 37
32476224394358733541546984465265719557637682166 874
47**15**14267158262537826937364893714847591482595861 25
857**4**52822296856393331796741444281785255539289636 66
242**1**23924835323413594643452461575456357268656745 83
246**1123532**35722346434683345754579445686568155679 76
42365327**1**534764385264587549637569865174867197628 5
2314249632342535174345364628545647573965675868417 6
342215569245**332**66713564437782467554889357866599 146
338597396444961841755517295286662831639777394274 18
351359589446246169155735727126684683823779579493 48
435873354154698446526571955763768216687487932779 859814388196
582625378269373648937148475914825958612593616972 36
968563933317967414442817852555392896366641391747 77
353234135946434524615754563572686567468379767856 794
357223464346833457545794456865681556797679266781 87
534764385264587549637569865174867197628597821873 96
342535174345364628545647573965675868417678697952 878971816398
564437782467554889357866599146897761125795187223 68
551729528666283163977394274188841538529995264963 1
573572712668468382377957949348816815145992792625 61
657195576376821668748793277985981438819619254992 17
714847591482595861259361697236147271834725838294 58
281785255539289636664139174777524128588863523969 99
575456357268656746837976785794818789681592989179 26
579445686568155679767926678187813778929892488913 19
756986517486719762859782187396189329841729143195 28
564757396567586841767869795287897181639891829274 19

6755488935786659914689776112579188722368129933479

The total risk of this path is **315** (the starting position is still never entered, so its risk is not counted).

Using the full map, **what is the lowest total risk of any path from the top left to the bottom right?**

1.3.1 Thoughts

We have two options here: - create the enlarged risk map from the original and then create the risk graph from that. - create a risk graph that is 5 times larger and set the weighting depending on the iteration of the original graph

The first option looks the easiest at the moment so will work with that and see how it goes!

[ ]:
```
def expand_risk_data(original_risk_data):
    ''' Expand the data matrix from 10x10 to 50x50
    '''

    new_risk_data: np.array = np.copy(original_risk_data)
    updated_risk_data: np.array = np.copy(original_risk_data)
    for x in range(4):
        updated_risk_data = np.where(updated_risk_data == 9, 1,␣
 ↪updated_risk_data+1)
        new_risk_data = np.concatenate((new_risk_data, updated_risk_data),␣
 ↪axis=1)

    updated_risk_data = np.copy(new_risk_data)
    for x in range(4):
        updated_risk_data = np.where(updated_risk_data == 9, 1,␣
 ↪updated_risk_data+1)
        new_risk_data = np.concatenate((new_risk_data, updated_risk_data),␣
 ↪axis=0)

    return new_risk_data

# verify the test data
new_risk_data = expand_risk_data(get_data(TEST_PATH))
height, width = new_risk_data.shape
new_risk_graph = create_graph(new_risk_data)
path_length = nx.shortest_path_length(new_risk_graph, source=(0, 0),␣
 ↪target=(height-1, width-1), weight="weight")
assert(path_length == 315)

# OK so get problem solution
new_risk_data = expand_risk_data(get_data(PROBLEM_PATH))
height, width = new_risk_data.shape
new_risk_graph = create_graph(new_risk_data)
```

```
path_length = nx.shortest_path_length(new_risk_graph, source=(0, 0),␣
 ↪target=(height-1, width-1), weight="weight")

print(f'Minimum expanded problem path length: {path_length}')
```

```
Minimum expanded problem path length: 2835
```

[ ]: