Data Structure and Programming, Spring 2024
programming assignment #2

Monotonic Routing

Chiao-Yun, Chin
b10901154@ntu.edu.tw

Shao-Ming, Chen
r11942095@ntu.edu.tw

Pei-Yuan, Wu
peiyuanwu@ntu.edu.tw

DUE DATE: 04/18/2024

1 Problem Description

Given an $m \times n$ grid graph, the cost of each graph edge (u, v) is $d(u, v)$, a source grid is S, and a target grid is T (see Figure 1(a)). Let $D(S, g)$ denote the minimum cost of a non-detour path from S to a grid g. Please write a program that finds a monotonic path from the source grid to the target grid, i.e., compute $D(S, T)$, and reports the total edge cost, the number of grids, and the grid coordinates of the path.

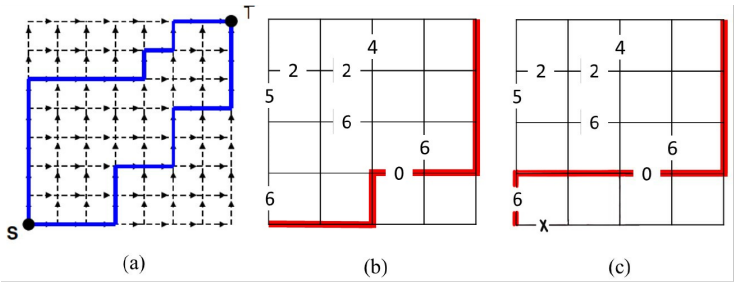

Figure 1: monotonic routing and two example

Notice:

1. Please use python3 to run your program.
2. Monotonic means that you can only go up or right, and you **can't** go back (left or below).

2 Input/Output Specification

2.1 Input format

The file format for the monotonic routing is illustrated, with comments in italics (these will not be in actual input files). The 1st line gives the problem size in terms of the indices of the left-bottom and the right-top grids (the indices of the left-bottom grid will always be (0,0)). Each edge connecting two grids has either default cost or non-default cost. The default cost is given in the 2nd line. The 3rd line gives the number of edges that have non-default costs. Start from the 4th line, the cost difference compared to the default cost of each non-default edge is separately given. For example, if the default cost is 3, then 2 3 2 4 1 means the cost from (2, 3) to (2, 4) is 1 + 3 = 4 . Additionally, if the cost difference is denoted by 'x', it signifies that the edge is not traversable, i.e, cost = ∞. For instance, 2 3 2 4 x indicates that there is no viable path from (2, 3) to (2, 4). Finally, the coordinates of the source and the target grids are listed in the last two lines.
The input file format is as follows:

BoundaryIndex # # # # *//the indices of the left-bottom and the right-top grids*

DefaultCost # *//the default cost*

NumNonDefaultCost # *//the number of non-default edges*

x1 y1 x2 y2 # *// the cost difference of the non-default edge between (x1,y1) and (x2,y2)*

...

//repeat for the number of non-default edges

Source xS yS *//the coordinate of the source grid*

Target xT yT *//the coordinate of the target grid*

2.2 output format

The resulting monotonic path needs to be described in the output file. The 1st line gives the total edge cost in the resulting path, and the 2nd line gives the number of grids on the path. After that, the consecutive grids in the path from source to target have to be listed in order.
The output file format is as follows:

RoutingCost [total edge cost]

RoutingPath [# of grids, k]

[x_1] [y_1]

[x_2] [y_2]

...

[x_k] [y_k]

//repeat for the number of grids in the path

Notice:

1. x1 and y1 must be the same as xS and yS, and xk and yk must be the same as xT and yT in the input file, respectively.
2. There is no need to consider cases where there is no viable path from S to T.

Here is an input/output example of Figures 1(b)/(c):

Sample Input	Sample Output
BoundaryIndex 0 0 4 4	RoutingCost 21
DefaultCost 3	RoutingPath 9
NumNonDefaultCost 8	0 0
3 1 3 2 3	1 0
2 3 2 4 1	2 0
2 1 3 1 -3	2 1
1 3 2 3 -1	3 1
1 2 2 2 x	4 1
0 3 1 3 -1	4 2
0 2 0 3 2	4 3
0 0 0 1 3	4 4
Source 0 0	
Target 4 4	

3 Command-line parameter

In order to test your program, please add the following command-line parameters to your program (e.g., python mono.py –input 5x5.in –output 5x5.out).

4 Submission

Please put mono.py and readme.txt into a directory named studentID and compress the directory into studentID.zip. Finally, upload studentID.zip to NTU Cool. The homework is due on 4/18, at 23:59.

5 Grading policy

This programming assignment will be graded based on (1) the correctness (2) solution quality, and (3) running time. Please check these items before your submission. We provide you three test data: 5x5.in, 50x50.in, 500x500.in, and we will score your program by two private data. If the output format, the number of grids on the path and the consecutive grids in the path from source to target are right, you will get 10% each private data. If the time of executing your algorithm is within the time limit, you will get 20% each private data. The remaining score depends on your solution quality. We will sort all the routing cost and give score from the least routing cost to the largest routing cost.

Notice: If the time of executing your algorithm exceeds the time limit, we'll stop the execution, and you will get 0% on this item. (Brute-Force method will definitely exceed the time limit.)

6 Bonus problem: Hilbert Curve (20pts)

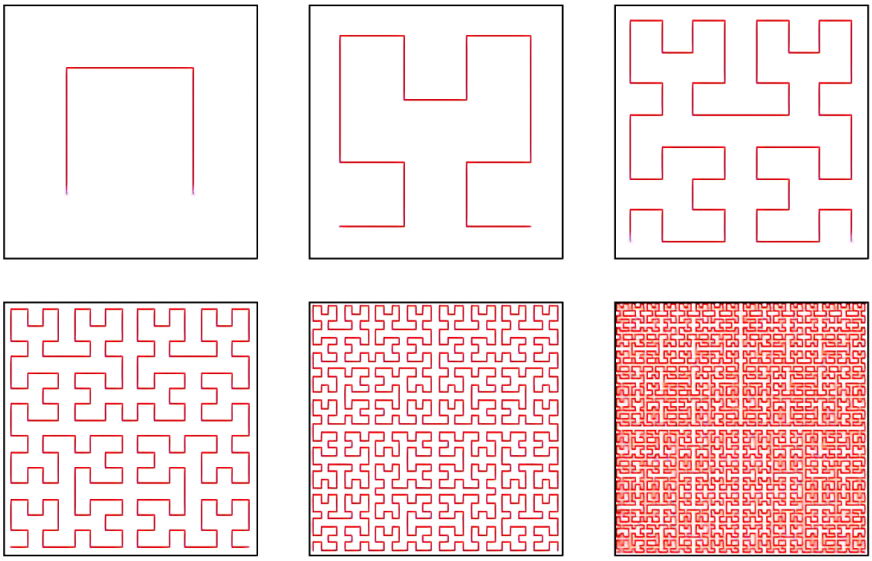

Figure 2: Hilbert curves from order 1 to 6

6.1 Problem Description

This problem is divided into two subproblems. In the first subproblem, you are asked to create a Python program to generate and visualize the Hilbert curve of different orders. The Hilbert curve is a space-filling curve that fills a 2D space with a single continuous line. You will write a function to generate the Hilbert curve and then use this function to create images of Hilbert curves of orders 1 to 4.

In the second subproblem, you are asked to create a program that will convert a given grayscale image into a Hilbert curve representation based on its intensity. The idea is to use low-order curves to represent light regions and high-order curves to represent dark regions in the image. This approach can help emphasize the image's key features and create a visually appealing representation.

For those who are not familiar with the Hilbert curve, we recommend watching the following YouTube video to gain a better understanding:

https://www.youtube.com/watch?v=nXvSyKcnJy0

Requirement 1:

1. Write a function hilbert curve(order) that takes an integer order as input and returns a Hilbert curve image corresponds to that order.

2. Save the above function in a file named hilbert.py.

3. When the file hilbert.py is executed, it should generate and save four images in the working directory, named h 1.png, h 2.png, h 3.png, and h 4.png, corresponding to the Hilbert curves of orders 1 to 4, respectively

Requirement 2:

1. Write a program hilbert art.py that takes a greyscale image as input and returns an image containing the Hilbert curve representation based on its intensity.

2. We provide Figure 3 as an example. We used two different orders for the light and dark areas, but feel free to use more orders to make your image look even cooler.

Note 1: You should not use any pre-built functions or external libraries that can generate the Hilbert curve directly. Note 2: If you can't manage to save the output images, you can show your work by recording a screen capture.

6.2 Submission

Please place the following files into a directory named bonus: hilbert art.py, hilbert.py, a greyscale image that you wish to convert, a README file that explains how to run your code, and a screen recording if necessary. Then, place the bonus directory into the studentID directory.

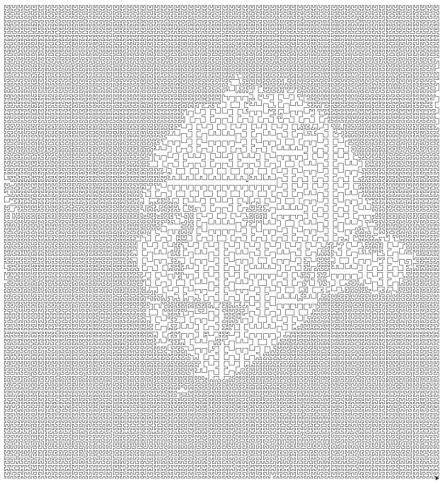

Figure 3: Hilbert curve space filling