Write a program to recursively find a path from point A to point B. Obstacles are marked with 0.

The output is a 2D matrix with 1’s in cells of the path

Example:

Map named as Example.npy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 |

Sample input:

Enter Map Name: example.npy

Enter Start Coordinates: 5, 3

Enter End Coordinates: 2, 1

Output:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |

Sample input:

Enter Map Name: example.npy

Enter Start Coordinates: 5, 3

Enter End Coordinates: 1, 1

Output:

Path does not exist.

Optional: Write a program to recursively find the shortest path.

Convolution is a process of multiplying linearly one matrix to a part of the bigger matrix and taking the sum of the results. It can be thought of as placing the window on top of the Matrix such that cells match and then multiplying them.

Matrix.npy:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 |

Window.npy:

|  |  |
| --- | --- |
| 1 | 2 |
| 3 | 4 |

Convolution result:

|  |  |  |  |
| --- | --- | --- | --- |
| 1x1 + 2x2 + 3x6 + 4x7 | 1x2 + 2x3 + 3x7 + 4x8 | 1x3 + 2x4 + 3x8 + 4x9 | 1x4 + 2x5 + 3x9 + 4x10 |
| 1x6 + 2x7 + 3x11 + 4x12 | 1x7 + 2x8 + 3x12 + 4x13 | 1x8 + 2x9 + 3x13 + 4x14 | 1x9 + 2x10 + 3x14 + 4x15 |
| 1x11 + 2x12 + 3x16 + 4x17 | 1x12 + 2x13 + 3x17 + 4x18 | 1x13 + 2x14 + 3x18 + 4x19 | 1x14 + 2x15 + 3x19 + 4x20 |
| 1x16 + 2x17 + 3x21 + 4x22 | 1x17 + 2x18 + 3x22 + 4x23 | 1x18 + 2x19 + 3x23 + 4x24 | 1x19 + 2x20 + 3x24 + 4x25 |

|  |  |  |  |
| --- | --- | --- | --- |
| 51 | 61 | 71 | 81 |
| 101 | 111 | 121 | 131 |
| 151 | 161 | 171 | 181 |
| 201 | 211 | 221 | 231 |

Pooling is taking a unique window of size N x N and taking only the maximum number in that window and creating a resulting matrix.

Sample output:

For a pool size of 2

Pool Result:

|  |  |
| --- | --- |
| 111 | 131 |
| 211 | 231 |

np.amax() can be used to find the maximum element in a ndarray

Write a program to reduce the resolution for a BGR image and also convert it to grayscale.

Color weights being R \* 0.2989, G \* 0.5870, B \* 0.1140.

Image.npy is a (m x n x 3) image stored as a npy file.

HD images ares 1920 x 1080 x 3.

Sample Input:

Enter File Name: Image.npy

Enter Resolution: 300, 300

Output is a saved npy file called answer

To read a npy file:

image = np.load(“Image.npy”)

To save to a npy file:

np.save(“Resized\_Image”, resized\_image)

np.save(“Grayscale”, resized\_image)

Optional: Resize an image to a size larger than the input resolution.

Write a program to create a JSON (A nested dictionary where the innermost field’s value may be a array or list) from the given data.

Hierarchy of fields is as follows :

Stage > Semester > Class > Topic > Grade > Password = Birth Place

Only Passwords that match the following criteria are put into the JSON:

1. At least 1 letter between [a-z]

2. At least 1 number between [0-9]

1. At least 1 letter between [A-Z]

3. At least 1 character from [$#@]

4. Minimum length of transaction password: 6

5. Maximum length of transaction password: 12

Data for the question is given in Data.csv

Output:

The JSON

Example:

{“Name”: {“First”: “Arun”, “Last”: “V K”}}

Optional: Create a json from a csv.

Hint: csv – Comma Seperated Values

A Hotel requires a software for managing room bookings. Write a program with appropriate and efficient data structure to book rooms 30 days in advance. The long term plan is such that the same data structure is expanded to allow bookings 3-6 months in advance.

Since this was an interview question for Goldman Sachs, we are going to assume that the hotel has only one room so that the data structure slightly varies and the answer cannot be googled.

Write a program to create an efficient structure for booking the room and check availability of the room and booking the room if it is available for that time slot.

In a few lines at the top, comment the object (E.g. ndarray) you are using as the “appropriate efficient data structure” used and why.

Input Format:

Integer N representing the number of bookings

Start\_Date, End\_Data, Check\_In\_Time, Check\_Out\_Time

Sample input:

3

2, 4, 11.15, 12.30

3, 6, 15.30, 21.30

4, 5, 13.30, 10.45

Output:

Yes

Already Booked

Yes

Optional: Write a program with a appropriate and efficient data structure to book rooms m days in advance for n rooms collecting relevant details for the same. Checking availability should have a time complexity of O(logn) and new entries should have a time complexity of O(n)