Practice Arena

Practice problems aimed to improve your coding skills.

Convoluted Convolutions

LAB-PRAC-09_PTR-MAT

Convoluted Convolutions [20 marks]

The process of convolution is immensely useful in several areas of computer science, including image processing, robotics, machine learning etc. However, the process is actually very simple and requires a few elementary 2D array operations to be performed. You see, in computer science, images are represented as a 2D array of *pixels*. Each pixel (for sake of simplicity) can be thought of as simply an integer. If you have an image with 20 rows and 30 columns, you will have a total of 20 x 30 = 600 pixels and this image can be represented as a 2D array with 20 rows and 30 columns. Equivalently, when someone says that the size of image is 2048 x 1536 pixels, what they mean is that the matrix of pixels is of the size 2048 x 1536.

The process of convolution is simply that of adding each element of the image to its local neighbors, weighted by a certain thing called a *kernel*. Don't be scared by that statement. We'll explain this to you. As we said, every image can be thought of as just a 2D matrix of pixels. Each pixel has an integer value denoting the color stored at that pixel.

Now lets talk about kernels. A kernel is simply a square matrix with side k, where k is a strictly positive odd integer. Each value in this matrix may be a float number. Applying a kernel to any image means applying the kernel to every pixel of the image. Example: Consider the following 4 x 4 matrix given below. Let's apply a 3 x 3 kernel to this matrix. We choose a pixel for our demonstration (the one colored red). We align the center of the kernel matrix with the required pixel (the grey 3 x 3 region).

Image:

A11	A12	A13	A14
A21	A22	A23	A24
A31	A32	A33	A34
A41	A42	A43	A44

Kernel:

K11	K12	K13
K21	K22	K23
K31	K32	K33

Applying kernel to pixel 3,3

A11	A12	A13	A14
A21	A22	A23	A24

A31	A32	A33	A34
A41	A42	A43	A44

Applying the filter to pixel (3, 3) updates the value of that pixel according to following equation. A’ denotes the pixel value in new image.

$$A'_{3,3} = \frac{\left(\begin{array}{l} A_{2,2} * K_{1,1} + A_{2,3} * K_{1,2} + A_{2,4} * K_{1,3} + A_{3,2} * K_{2,1} + \\ A_{3,3} * K_{2,2} + A_{3,4} * K_{2,3} + A_{4,2} * K_{3,1} + A_{4,3} * K_{3,2} + A_{4,4} * K_{3,3} \end{array}\right)}{K_{1,1} + K_{1,2} + K_{1,3} + K_{2,1} + K_{2,2} + K_{2,3} + K_{3,1} + K_{3,2} + K_{1,3}}$$

The above expression may generate a non-integer value which is typecast to int before using it. The above is carried out for all pixels. At the edges, the kernel wraps around and uses element close to opposite edge

Example:

A11	A12	A13	A14
A21	A22	A23	A24
A31	A32	A33	A34
A41	A42	A43	A44

You’ll receive a image with m x n pixels and a k x k kernel. You’ve to apply the kernel to the image and print the new image.

Input format
The first line contains 3 integers: m, n, and k.
Next m lines contain n integers each, each giving a row of the image.
Next k lines contain k real values each, each giving a row of the kernel. Note that kernel values are to be interpreted as a float value.

Output format
The output should contain m lines, with n integers on each line. These integers should denote the pixel values of new image.

Caution:

1. Take care of trailing white-spaces.
2. Rest ensured that the summation in the denominator of update equation is never zero.
3. Remember that although the convolution process may generate a floating point number, make sure to cast it into an int before giving your output.
4. In your output, make sure there are no trailing spaces or trailing new lines. Output one row in one line, with a single space between two entries of a row.

Code to manipulate matrices

```
int m, n;
scanf("%d %d", &m, &n);
int num[m][n], i, j;
for(i = 0; i < m; i++)
        for(j = 0; j < n; j++)
                scanf("%d", &num[i][j]);
                printf("%d", num[i][j]);
```

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are several lines in your output. Printing each line correctly, in the correct order, carries equal weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

Start Solving! (/editor/practice/6193)