

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH



## Kiến Trúc Máy Tính

---

Xây Dựng và Triển Khai Phép Tích Chập trên Kiến Trúc MIPS

---

GVHD:  
Sinh viên thực hiện:  
MSSV:  
Lớp:

Nguyễn Thành Lộc  
Ngô Trương Phú  
2352915  
CN03

Thành phố Hồ Chí Minh, Tháng 11/2024

---

## NHẬN XÉT CỦA GVHD

---

# Mục lục

<b>1</b>	<b>Giới thiệu:</b>	<b>3</b>
1.1	Tổng quan:	3
1.2	Phép tích chập là gì?	3
1.3	Cách thực hiện phép tích chập	3
1.4	Giải Quyết Vấn Đề Mất Pixel Biên	4
1.5	Yêu cầu	4
<b>2</b>	<b>Phân tích chương trình</b>	<b>5</b>
2.1	Flowchart của bài toán	5
2.2	Khai báo biến dữ liệu	6
2.3	Đọc dữ liệu từ file	7
2.4	Tính toán ma trận kết quả (tích chập)	8
2.5	Xuất kết quả	8
<b>3</b>	<b>Test case và kết quả:</b>	<b>10</b>
3.1	Test case số 1	10
3.2	Test case số 2	10
3.3	Test case số 3	11
3.4	Test case số 4	11
3.5	Test case số 5	11
3.6	Test case số 6	12
3.7	Test case số 7	12
3.8	Test case số 8	12
<b>4</b>	<b>Lời cảm ơn</b>	<b>13</b>

# 1 Giới thiệu:

## 1.1 Tổng quan:

- Trong thời đại ngày nay, trí tuệ nhân tạo (AI) đang phát triển với tốc độ vượt bậc. Sự cải tiến nhanh chóng về kỹ thuật và dữ liệu đã dẫn đến sự gia tăng đáng kể trong khối lượng tính toán. Các nhánh con của AI, như học máy (machine learning) và học sâu (deep learning), đã đạt đến mức mà hàng tỷ phép tính phải được thực hiện để tính toán hàng triệu tham số trong một mô hình.
- Mạng nơ-ron tích chập (Convolutional Neural Network - CNN) là một trong những loại mạng được sử dụng phổ biến nhất trong học sâu. CNN được ứng dụng rộng rãi trong nhiều lĩnh vực, đặc biệt là xử lý hình ảnh và video. Tuy nhiên, dù một mô hình CNN có tiên tiến đến đâu, nó vẫn bắt nguồn từ các phép toán ma trận, nổi bật là phép tích chập (convolution operation).

## 1.2 Phép tích chập là gì?

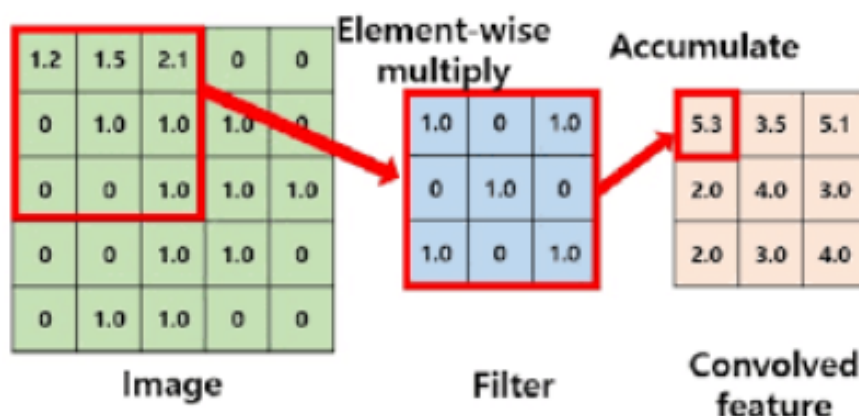
- Phép tích chập là quá trình trượt một bộ lọc (filter), còn được gọi là kernel, qua ma trận đầu vào. Bộ lọc này là một ma trận nhỏ chứa các trọng số được áp dụng cho một vùng cục bộ của ma trận đầu vào, tạo ra một giá trị duy nhất trong bản đồ đặc trưng (feature map) đầu ra. Quá trình này được lặp lại trên toàn bộ ma trận, cho phép mạng học các đặc trưng khác nhau từ ma trận đầu vào.
- Trong phân tích hình ảnh, các đặc trưng này có thể là đường biên, kết cấu (textures), hoặc các mẫu hình (patterns). Về mặt toán học, phép tích chập thực hiện phép tích vô hướng (dot product) giữa kernel và vùng receptive field của ảnh đầu vào.

## 1.3 Cách thực hiện phép tích chập

Quy trình của phép tích chập bao gồm:

- Lấy một ma trận nhỏ, gọi là *kernel* hoặc *filter*, và trượt nó qua ma trận đầu vào, chẳng hạn một hình ảnh.
- Tại mỗi vị trí, tính **tích vô hướng** giữa *kernel* và vùng chồng lấp của ma trận đầu vào.
- Lưu kết quả vào ma trận đầu ra.
- Lặp lại quá trình này cho tất cả các vị trí của *kernel* trên ma trận đầu vào, giúp trích xuất các đặc trưng như đường biên hoặc kết cấu.

## Ví dụ hình ảnh



Hình 1: Minh họa phép tích chập với ma trận đầu vào 5x5 và kernel 3x3.

## 1.4 Giải Quyết Vấn Đề Mất Pixel Biên

Một vấn đề thường gặp khi áp dụng các lớp tích chập là việc mất pixel ở phần rìa của hình ảnh. Để giải quyết vấn đề này, một giải pháp đơn giản là **thêm các pixel đệm** (*padding*) xung quanh biên của ma trận đầu vào, từ đó tăng kích thước hiệu quả của hình ảnh. Thông thường, giá trị của các pixel đệm được đặt bằng 0.

### Ví dụ hình ảnh

0	0	0	0	0
0	0	1	2	0
0	3	4	5	0
0	6	7	8	0
0	0	0	0	0

Hình 2: Minh họa zero padding.

## 1.5 Yêu cầu

Trong bài tập này, nhiệm vụ của bạn là thực hiện các phép toán tích chập (convolution) sử dụng mã lệnh MIPS. Các yêu cầu được liệt kê trong các phần dưới đây.

### Đầu vào

Chương trình cần nhận dữ liệu đầu vào từ một tệp tin ngoài. Tệp tin này có tên là `input_matrix.txt`. Nội dung của tệp tin này là các số được phân tách bằng khoảng trắng. Cả ma trận hình ảnh và ma trận kernel đều là ma trận vuông, với kích thước lần lượt là  $m \times m$  và  $n \times n$ . Ngoài ra, tệp tin `input_matrix.txt` sẽ chứa 3 dòng, trong đó dòng đầu tiên bao gồm 4 giá trị như sau:

- $N$ : Kích thước của ma trận hình ảnh ( $3 \leq N \leq 7$ ).
- $M$ : Kích thước của ma trận kernel ( $2 \leq M \leq 4$ ).
- $p$ : Giá trị padding ( $0 \leq p \leq 4$ ).
- $s$ : Giá trị stride ( $1 \leq s \leq 3$ ).

Do đó, nội dung của ma trận hình ảnh và ma trận kernel sẽ được chứa trong dòng thứ hai và thứ ba, tương ứng.

Để đơn giản, stride sẽ có giá trị giống nhau cho cả hai chiều, trong khi padding sẽ được áp dụng đối xứng. Tất cả các giá trị trong ma trận đều là số thực, trong khi stride và padding là các số nguyên.

### Đầu ra

Chương trình sẽ xuất kết quả vào một tệp tin văn bản có tên là `output_matrix.txt`, chứa ma trận kết quả của phép tích chập, với các giá trị được phân tách bằng khoảng trắng.

### Các biến được xác định

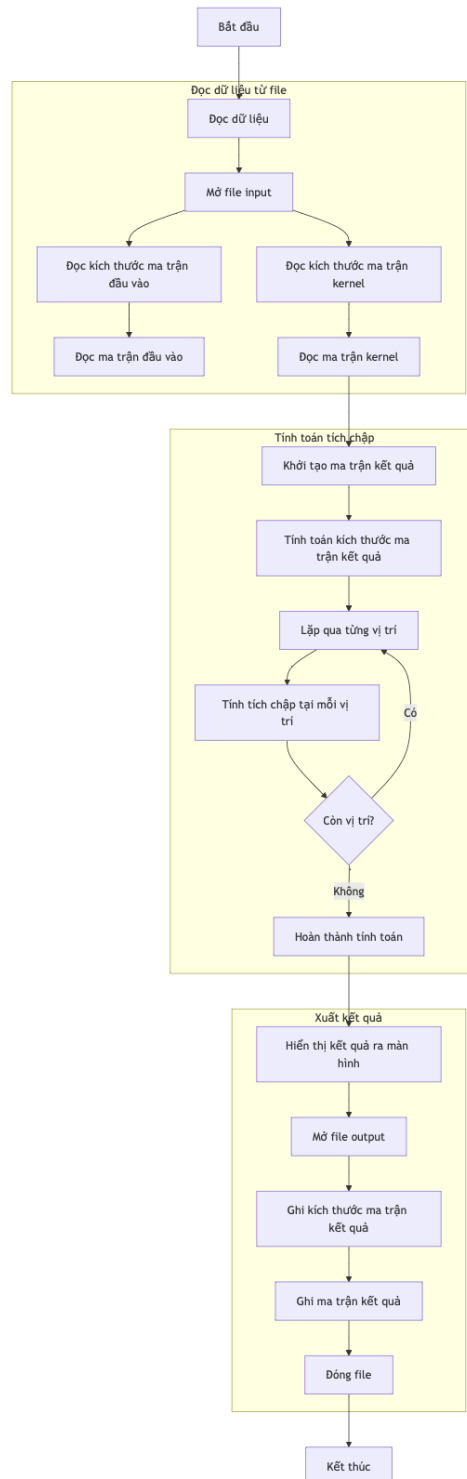
Một số biến cần được định nghĩa như sau:

- `image` (word): Lưu trữ ma trận hình ảnh.
- `kernel` (word): Lưu trữ ma trận kernel.
- `out` (word): Lưu trữ ma trận kết quả tích chập.

## 2 Phân tích chương trình

Chương trình thực hiện phép tích chập được chia thành ba phần lớn: **Đọc dữ liệu từ file**, **Tính toán ma trận kết quả (tích chập)**, và **Xuất kết quả ra màn hình và file**. Trong mỗi phần, các hàm nhỏ đảm nhiệm từng nhiệm vụ cụ thể, giúp tổ chức chương trình rõ ràng và dễ bảo trì.

### 2.1 Flowchart của bài toán



Hình 3: Hình ảnh flowchart của bài toán tính

## 2.2 Khai báo biến dữ liệu

### 2.2.0.1 Biến và dữ liệu đầu vào

- `input_file_name`: Đường dẫn đến file đầu vào chứa dữ liệu ma trận ảnh và kernel.
- `output_file_name`: Đường dẫn đến file để ghi kết quả tính toán ma trận đầu ra.
- `buffer`: Bộ đệm dùng để lưu trữ nội dung của file đầu vào, có kích thước 1024 byte.
- `newline`: Ký tự xuống dòng, sử dụng trong việc hiển thị kết quả.
- `space`: Ký tự dấu cách, sử dụng trong việc hiển thị kết quả.
- `output`: Chuỗi ký tự "Kết quả tích chập là:", được in ra màn hình khi hiển thị kết quả.
- `N, M, p, s`: Các tham số đầu vào để điều khiển quá trình tính toán ma trận:
  - `N`: Kích thước ma trận ảnh gốc.
  - `M`: Kích thước kernel (ma trận nhân).
  - `p`: Kích thước padding (biên mở rộng).
  - `s`: Giá trị stride (bước nhảy).
- `float_zero`: Biến chứa giá trị số thực 0.0.
- `float_ten`: Biến chứa giá trị số thực 10.0.
- `cham`: Ký tự dấu chấm dùng để in sau phần thập phân của số thực.
- `tru`: Ký tự dấu trừ, sử dụng trong việc hiển thị các số âm.
- `errorrg`: Thông báo lỗi nếu kích thước output không hợp lệ.
- `write_error_msg`: Thông báo lỗi khi ghi kết quả ra file.
- `zero_char`: Ký tự "0", sử dụng để xử lý giá trị số bằng 0.
- `digit_array`: Mảng lưu tạm các chữ số.

### 2.2.0.2 Lưu trữ ma trận

- `matrananh`: Ma trận chứa ảnh gốc với kích thước tối đa là 7x7 (49 phần tử). Tương đương với biến image trong yêu cầu.
- `matranpad`: Ma trận chứa ảnh sau khi padding với kích thước tối đa là 15x15 (225 phần tử).
- `matranketqua`: Ma trận chứa kết quả tích chập với kích thước tối đa là 15x15 (225 phần tử). Tương đương với biến out trong yêu cầu.
- `matrankernel`: Ma trận kernel với kích thước tối đa là 5x5 (25 phần tử). Tương đương với biến kernel trong yêu cầu.

### 2.2.0.3 Kích thước ma trận

- `P_size`: Biến lưu kích thước của ma trận sau khi áp dụng padding, được tính theo công thức:

$$P\_size = N + 2 \times p$$

- `O_size`: Biến lưu kích thước của ma trận đầu ra sau khi thực hiện phép tích chập, được tính theo công thức:

$$O\_size = \frac{P\_size - M}{s} + 1$$

Trong đó:

- `N` là kích thước ma trận ảnh ban đầu.
- `M` là kích thước kernel.
- `p` là kích thước padding.
- `s` là giá trị stride (bước nhảy).

## 2.3 Đọc dữ liệu từ file

### 2.3.0.1 Phần đọc file (readInput) Để mở file, sử dụng mã system call 13 trong MIPS:

- `$v0 = 13`: Chỉ định mã system call để mở file.
- `$a0`: Chứa địa chỉ của tên file cần mở.
- `$a1 = 0`: Chỉ định mode mở file là đọc (read).
- `$a2 = 0`: Chỉ định quyền truy cập là read-only.

Sau khi thực hiện system call, file descriptor sẽ được lưu trong thanh ghi `$v0`.  
Để đọc dữ liệu từ file, sử dụng mã system call 14:

- `$v0 = 14`: Mã system call đọc file.
- `$a0`: Chứa file descriptor.
- `$a1`: Địa chỉ buffer để lưu dữ liệu đọc vào.
- `$a2 = 1024`: Chỉ định số byte tối đa cần đọc (1024 bytes).

### 2.3.0.2 Đọc số nguyên (readInt) Để đọc số nguyên, ta thực hiện các bước sau:

- Khởi tạo kết quả bằng 0.
- Đọc từng ký tự từ buffer và kiểm tra xem có phải là một ký tự số (ASCII từ 48 đến 57).
- Nếu ký tự là số, chuyển đổi từ ASCII sang giá trị số thực tế và cập nhật kết quả:

$$\text{result} = \text{result} * 10 + \text{digit}$$

- Nếu ký tự không phải là số, bỏ qua các ký tự đặc biệt như khoảng trắng, tab, newline.

### 2.3.0.3 Đọc số thực (readFloat) Hàm này phức tạp hơn, xử lý cả phần nguyên và phần thập phân:

- Khởi tạo các biến cho phần nguyên (`$v0`), phần thập phân (`$t7`), số chữ số thập phân (`$t8`), và dấu (`$t9`).
- Kiểm tra xem có dấu âm hay không (ASCII của dấu "-" là 45). Nếu có, đánh dấu số âm và bỏ qua dấu âm.
- Đọc phần nguyên của số cho đến khi gặp dấu chấm.
- Sau khi xử lý phần nguyên, đọc phần thập phân sau dấu chấm. Lặp lại quá trình cho đến khi hết ký tự.
- Chuyển phần nguyên thành số thực, kết hợp với phần thập phân đã tính toán.

Kết quả phần nguyên sẽ được chuyển vào thanh ghi số thực (`$f0`), và phần thập phân được chuyển thành số thực với phép chia cho  $10^n$ .

### 2.3.0.4 Đọc ma trận Để đọc ma trận, ta cần làm như sau:

- Đọc ma trận hình ảnh: Lấy kích thước ma trận từ thanh ghi `$s1`, sau đó đọc  $N * N$  phần tử số thực và lưu vào mảng ma trận hình ảnh (`matrananh`).
- Đọc ma trận kernel: Tương tự như ma trận hình ảnh, nhưng kích thước  $M * M$ .

### 2.3.0.5 Hàm skip\_the\_line

- Hàm `skip_the_line` giúp bỏ qua các ký tự không cần thiết cho đến khi gặp newline (10 trong ASCII) hoặc EOF (end of file). Hàm này giúp di chuyển con trỏ đến dòng tiếp theo trong file.

### 2.3.0.6 Xử lý

- **Xử lý số thực:**
  - Sử dụng thanh ghi số thực (`$f0`, `$f1`, `$f2`) để lưu trữ giá trị số thực.
  - Chuyển đổi giữa số nguyên và số thực bằng các lệnh `mtc1` và `cvt.s.w`.
  - Đảm bảo độ chính xác single-precision (32-bit).



## 2.4 Tính toán ma trận kết quả (tích chập)

### Mục tiêu

- Thực hiện phép tích chập giữa ma trận ảnh đã được thêm padding và kernel.
- Tính toán và lưu kết quả vào ma trận đầu ra với kích thước  $O\_size \times O\_size$ , trong đó  $O\_size$  được tính dựa trên stride.

### Các bước thực hiện

#### 2.4.0.1 Tính kích thước ma trận

- Sử dụng hàm `calculate_output_size` để tính toán:

$$P\_size = N + 2 \times p$$
$$O\_size = \frac{P\_size - M}{s} + 1$$

- Nếu kết quả không hợp lệ (phần dư không bằng 0 khi chia stride), chương trình hiển thị lỗi và thoát.

#### 2.4.0.2 Thêm padding vào ma trận

- Hàm `paddingMatrix` thực hiện thêm padding cho ma trận ảnh:
  - Tạo một ma trận mới kích thước  $P\_size \times P\_size$ , trong đó các phần tử ngoài vùng ảnh gốc được gán giá trị 0.
  - Sao chép ma trận ảnh gốc vào trung tâm ma trận mới, đảm bảo giá trị padding được áp dụng đúng.

#### 2.4.0.3 Thực hiện tích chập

- Hàm `convolution` thực hiện các bước:
  - Khởi tạo:**
    - Tải thông số  $P\_size$ ,  $M$ ,  $s$ , và  $O\_size$ .
    - Khởi tạo các bộ đếm để duyệt qua các ô trong ma trận đầu ra.
  - Duyệt từng ô trong ma trận đầu ra:**
    - Với mỗi ô  $(i, j)$ , xác định vùng tương ứng trong ma trận đã padding.
    - Tính toán vùng này bằng cách duyệt qua kernel, nhân từng phần tử của kernel với phần tử tương ứng trong vùng, sau đó cộng các kết quả.
  - Lưu kết quả:**
    - Kết quả được lưu vào ô  $(i, j)$  của ma trận đầu ra.

### Kết quả

Sau bước này, ma trận đầu ra với kích thước  $O\_size \times O\_size$  được lưu trữ trong `matranketqua`.

## 2.5 Xuất kết quả

### Mục tiêu

- Hiển thị ma trận đầu ra trên màn hình với định dạng số thực, giữ 4 chữ số thập phân.
- Ghi ma trận đầu ra vào file `output_matrix.txt`.

### Các bước thực hiện

#### 2.5.0.1 Hiển thị kết quả

- Hàm `printResult` thực hiện:
  - Hiển thị tiêu đề "Ket qua tích chap la:".
  - Duyệt qua từng phần tử trong ma trận đầu ra:
    - Tách phần nguyên và phần thập phân.
    - In từng giá trị với định dạng số thực, giữ 4 chữ số thập phân.
  - Xuống dòng sau khi in xong mỗi hàng.

---

#### 2.5.0.2 Ghi kết quả ra file

- Hàm `write_output` thực hiện:
  1. Mở file `output_matrix.txt` ở chế độ ghi.
  2. Duyệt qua từng phần tử trong ma trận đầu ra:
    - Tách phần nguyên và phần thập phân.
    - Ghi từng giá trị vào file với định dạng số thực.
  3. Đóng file sau khi ghi xong.

#### Kết quả

- Ma trận đầu ra được hiển thị trên màn hình.
- Ma trận đầu ra được ghi vào file `output_matrix.txt` với định dạng chính xác.

### 3 Test case và kết quả:

Dữ liệu đầu vào bao gồm các phần sau:

- Hàng đầu tiên lần lượt là  $N, M, p, s$
- Hàng thứ hai là ma trận ảnh
- Hàng thứ ba là ma trận kernel

#### 3.1 Test case số 1

Input:

```
3 4 0 2
-3.0 -4 4.5 6 7.8 12 5 -0.5 12.0
1 1.2 -1.3 4.5 -5.0 3 3.5 6 -8.9 12 23.2 12 13 -14 -15 16.0
```

Output:

```
-- program is finished running --
Error
-- program is finished running --
```

Hình 4: Hình ảnh kết quả test case số 1

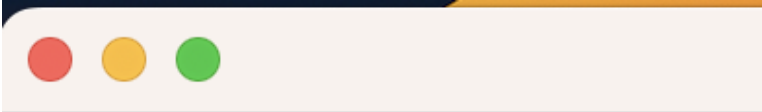
**Giải thích kết quả:** Ở Test case số 1, khi thực thi ta phát hiện đầu vào có  $M \geq N + 2 \cdot p$ , có nghĩa là không thể thực hiện phép tích chập vì phép tích chập yêu cầu kernel phải di chuyển qua ma trận đầu vào để tính toán giá trị cho mỗi vị trí của ma trận đầu ra. Nếu kích thước của kernel lớn hơn hoặc bằng kích thước của ma trận đầu vào sau khi padding, kernel sẽ không thể "di chuyển" qua ma trận đầu vào theo cách thông thường vì không còn không gian để thực hiện phép tính cho các điểm khác nhau. Vì thế sẽ in ra "error" và không có ma trận kết quả ở file output.

#### 3.2 Test case số 2

Input:

```
3 4 1 2
-3.0 -4 4.5 6 7.8 12 5 -0.5 12.0
1 1.2 -1.3 4.5 -5.0 3 3.5 6 -8.9 12 23.2 12 13 -14 -15 16.0
```

Output:



```
530.4600
```

Hình 5: Hình ảnh kết quả test case số 2

**Giải thích kết quả:**

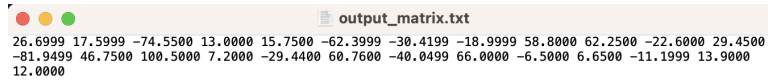
Ở Test case số 2, phép tích chập thực thi bình thường.

### 3.3 Test case số 3

Input:

```
3 3 2 1
-3.0 -4 4.5 6 7.8 12 5 -0.5 12.0
1 1.2 -1.3 4.5 -5.0 3 3.5 6 -8.9
```

Output:



```
26.6999 17.5999 -74.5500 13.0000 15.7500 -62.3999 -30.4199 -18.9999 58.8000 62.2500 -22.6000 29.4500
-81.9499 46.7500 100.5000 7.2000 -29.4400 60.7600 -40.0499 66.0000 -6.5000 6.6500 -11.1999 13.9000
12.0000
```

Hình 6: Hình ảnh kết quả test case số 3

**Giải thích kết quả:**

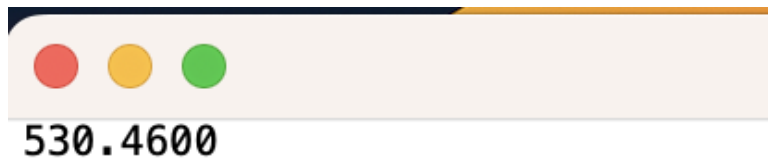
Ở Test case số 3, phép tích chập thực thi bình thường.

### 3.4 Test case số 4

Input:

```
3 4 1 2
-3.0 -4 4.5 6 7.8 12 5 -0.5 12.0
1 1.2 -1.3 4.5 -5.0 3 3.5 6 -8.9 12 23.2 12 13 -14 -15 16.0
```

Output:



```
530.4600
```

Hình 7: Hình ảnh kết quả test case số 4

**Giải thích kết quả:**

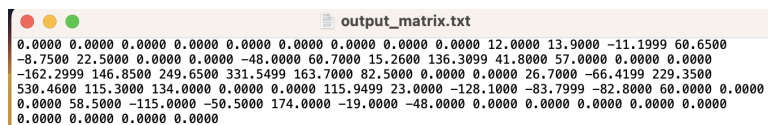
Ở Test case số 4, phép tích chập thực thi bình thường.

### 3.5 Test case số 5

Input:

```
4 3 3 1
1 1.2 -1.3 4.5 -5.0 3 3.5 6 -8.9 12 23.2 12 13 -14 -15 16.0
-3.0 -4 4.5 6 7.8 12 5 -0.5 12.0
```

Output:



```
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 12.0000 13.9000 -11.1999 60.6500
-8.7500 22.5000 0.0000 0.0000 -48.0000 60.7000 15.2600 136.3099 41.8000 57.0000 0.0000 0.0000
-162.2999 146.8500 249.6500 331.5499 163.7000 82.5000 0.0000 0.0000 26.7000 -66.4199 229.3500
530.4600 115.3000 134.0000 0.0000 0.0000 115.9499 23.0000 -128.1000 -83.7999 -82.8000 60.0000 0.0000
0.0000 58.5000 -115.0000 -50.5000 174.0000 -19.0000 -48.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000
```

Hình 8: Hình ảnh kết quả test case số 5

**Giải thích kết quả:**

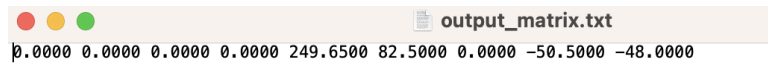
Ở Test case số 5, phép tích chập thực thi bình thường.

### 3.6 Test case số 6

Input:

```
4 3 3 3
1 1.2 -1.3 4.5 -5.0 3 3.5 6 -8.9 12 23.2 12 13 -14 -15 16.0
-3.0 -4 4.5 6 7.8 12 5 -0.5 12.0
```

Output:



```
0.0000 0.0000 0.0000 0.0000 249.6500 82.5000 0.0000 -50.5000 -48.0000
```

Hình 9: Hình ảnh kết quả test case số 6

**Giải thích kết quả:**

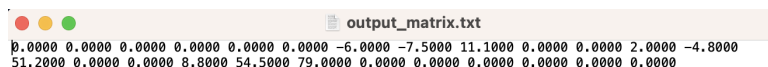
Ở Test case số 6, phép tích chập thực thi bình thường.

### 3.7 Test case số 7

Input:

```
5 2 3 2
-1 1 -2 3 -0.4 1 2.0 3 4 1 1.0 1.2 1.3 1.6 10 2.3 4.5 -5 -6.0 2 3 4 5 6 7.0
-3.0 -4 4.5 6
```

Output:



```
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 -6.0000 -7.5000 11.1000 0.0000 0.0000 2.0000 -4.8000
51.2000 0.0000 0.0000 8.8000 54.5000 79.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
```

Hình 10: Hình ảnh kết quả test case số 7

**Giải thích kết quả:**

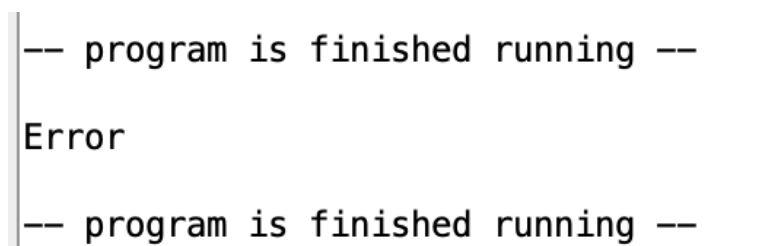
Ở Test case số 7, phép tích chập thực thi bình thường.

### 3.8 Test case số 8

Input:

```
2 4 1 2
-3.0 -4 4.5 6
1 1.2 -1.3 4.5 -5.0 3 3.5 6 -8.9 12 23.2 12 13 -14 -15 16.0
```

Output:



```
-- program is finished running --
Error
-- program is finished running --
```

Hình 11: Hình ảnh kết quả test case số 8

**Giải thích kết quả:**

Ở Test case số 8, vì đề bài yêu cầu ( $3 \leq N \leq 7$ ) và đầu vào là  $N = 2$  nên không thoả yêu cầu đề bài. Nên sẽ in ra "error" và không in ra gì ở file output.

---

## 4 Lời cảm ơn

Em xin chân thành cảm ơn Thầy Nguyễn Thành Lộc đã hỗ trợ và hướng dẫn em trong quá trình học về tính toán tích chập (Convolution Operation). Nhờ sự chỉ bảo của Thầy, em đã có cơ hội hiểu sâu hơn về vấn đề này và phát triển kỹ năng giải quyết vấn đề. Bên cạnh đó, sự hỗ trợ của Thầy cũng giúp em nâng cao nhiều kỹ năng trong việc tìm hướng làm và kiến thức, từ đó ứng dụng chúng trong tương lai.