

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM  
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG  
KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TIỂU LUẬN GIỮA KÌ MÔN ĐẠI SỐ TUYẾN TÍNH  
CHO CÔNG NGHỆ THÔNG TIN**

# **ĐẠI SỐ TUYẾN TÍNH CHO CÔNG NGHỆ THÔNG TIN**

*Người hướng dẫn:* **GV NGUYỄN VĂN KHOA**

*Người thực hiện:* **TRẦN GIA HÀO – 522H0080**

**NGUYỄN HUỲNH ANH KHOA – 522H0046**

**Lớp : 22H50202**

**Khoá : 26**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM  
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG  
KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TIỂU LUẬN GIỮA KÌ MÔN ĐẠI SỐ TUYẾN TÍNH  
CHO CÔNG NGHỆ THÔNG TIN**

# **ĐẠI SỐ TUYẾN TÍNH CHO CÔNG NGHỆ THÔNG TIN**

*Người hướng dẫn:* **GV NGUYỄN VĂN KHOA**

*Người thực hiện:* **TRẦN GIA HÀO – 522H0080**

**NGUYỄN HUỲNH ANH KHOA – 522H0046**

**Lớp : 22H50202**

**Khoá : 26**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

## LỜI CẢM ƠN

Lời đầu tiên, chúng em xin cảm ơn giảng viên Nguyễn Văn Khoa đã đồng hành và giúp đỡ chúng em trong suốt quá trình hoàn thành bài tập tiểu luận giữa kỳ cho môn Thực Hành Đại Số Tuyến Tính Cho Công Nghệ Thông Tin. Cảm ơn khoa Công Nghệ Thông Tin đã giao cho chúng em bài tập tiểu luận này để giúp chúng em có thể tiếp cận với những kỹ năng làm việc nhóm cũng như thích nghi với môi trường mới hơn. Tuy chúng em chưa có nhiều kỹ năng và kinh nghiệm trong suốt quá trình làm bài nên có thể có sai sót thì mong giáo viên chấm bài có thể góp ý để chúng em có thể phát triển hơn về sau này. Lời cuối cùng, chúng em xin cảm ơn các thầy/cô đã đọc và chấm điểm. Chúng em xin chúc giáo viên chấm bài cũng như các giảng viên thật nhiều sức khỏe để có thể truyền tải cho chúng em nhiều kiến thức thực tế và bổ ích hơn. Chúng em xin cảm ơn.

## **ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của GV Nguyễn Văn Khoa Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 24 tháng 4 năm 2023*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

***Hào***

*Trần Gia Hào*

***Khoa***

*Nguyễn Huỳnh Anh Khoa*

## PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

### Phần xác nhận của GV hướng dẫn

---

---

---

---

---

---

---

Tp. Hồ Chí Minh, ngày    tháng    năm  
(ký và ghi họ tên)

### Phần đánh giá của GV chấm bài

---

---

---

---

---

---

---

Tp. Hồ Chí Minh, ngày    tháng    năm  
(ký và ghi họ tên)

## TÓM TẮT

Bài tiểu luận này được thực hiện dựa trên các phương pháp, cách thức được giảng dạy trong quá trình học tập về các bài toán ma trận được thực hiện thông qua ngôn ngữ lập trình Python và có sử dụng thư viện numpy trong Python. Các thông tin, phương pháp giải được tham khảo thông qua các bài giảng trên lớp và các lab trong quá trình học tập. Thông qua đó, chúng ta có thể hiểu được chi tiết và rõ ràng hơn về những phương pháp, cách thức mà các ma trận, các phép tính toán trên ma trận được thực hiện. Hiểu rõ hơn về mục đích các hàm trong thư viện numpy liên quan đến các phép toán về ma trận. Qua bài tiểu luận lần này, chúng tôi đã hiểu rõ hơn về các phép toán của ma trận, các logic và cách xử lý vấn đề được nâng cao thông qua bài tiểu luận lần này. Hiểu sâu hơn về phép toán nhân ma trận `matmul` trong thư viện numpy.

## MỤC LỤC

LỜI CẢM ƠN	3
PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN	5
TÓM TẮT	6
MỤC LỤC	7
CHAPTER 1 – METHODOLOGY AND SOLVING TASKS	8
1.1 Giải thích hàm main	8
1.2. Câu d - Save odd integer numbers in the matrix A into a new vector, and print the resultant vector to the screen.	8
1.3 Câu e - Save prime numbers in the matrix A into a new vector, and print the resultant vector to the screen.	8
1.4 Câu f - Given a matrix, reverse elements in the odd rows of the matrix D, and print the resultant matrix to the screen.	9
1.5 Câu g: Regarding the matrix A, find the rows which have maximum count of prime numbers, and print the rows to the screen.	10
1.6 Câu h: Regarding the matrix A, find the rows which have the longest contiguous odd numbers sequence, and print the rows to the screen.	10
CHAPTER 2 - SOURCE CODES AND OUTPUTS	12
2.1 Hàm main	12
2.2 Ma trận A	13
2.3 Ma trận B	13
2.4 Ma trận C	13
2.5 Câu a	14
2.6 Câu b	15
2.7 Câu c	17
2.8 Câu d	18
2.9 Hàm isPrime()	19
2.10 Câu e	20
2.11 Câu f	21
2.12 Hàm countPrime()	22
2.13 Câu g	23
2.14 Câu h	24

## CHAPTER 1 – METHODOLOGY AND SOLVING TASKS

### 1.1 Giải thích hàm main

- Để cho thuận tiện trong việc quản lý, bảo trì và đảm bảo tính đúng đắn và không phát sinh lỗi trong chương trình, chúng tôi đặt mỗi câu hỏi vào một hàm với cú pháp “cau\_<câu hỏi>”, chúng tôi định nghĩa một hàm main() gọi tất cả các hàm “cau\_<câu hỏi>” để chạy chương trình. Chúng tôi cũng đặt hàm main() vào câu lệnh kiểm tra “if \_\_name\_\_ == '\_\_main\_\_'” để các chương trình có thể chạy như một chương trình độc lập.

### 1.2. Câu d - Save odd integer numbers in the matrix A into a new vector, and print the resultant vector to the screen.

- Tạo vector trống với tên gọi odd để lưu kết quả
- Tạo vòng lặp cho i chạy trong các dòng của ma trận A qua lệnh ‘A.shape[0]’
- Tạo vòng lặp lồng cho j chạy trong các cột của ma trận A qua lệnh ‘A.shape[1]’
- Câu lệnh điều kiện kiểm tra phần tử dòng i cột j có lẻ hay không qua dòng lệnh ‘A[i][j] % 2 != 0’
- Nếu thỏa điều kiện thì lưu phần tử đó vào vecto odd
- In kết quả odd ra màn hình.

### 1.3 Câu e - Save prime numbers in the matrix A into a new vector, and print the resultant vector to the screen.

- Tôi này định nghĩa một hàm "isPrime" để kiểm tra xem một số có phải là số nguyên tố hay không. Sau đó, tôi sử dụng hàm "isPrime" để tìm các số nguyên tố trong ma trận A.
- Hàm "isPrime" nhận đầu vào là một số nguyên dương n. Nếu n là số nguyên tố, hàm sẽ trả về True. Nếu n không phải số nguyên tố, hàm sẽ trả về False. Hàm sử



dùng một vòng lặp để kiểm tra xem  $n$  có chia hết cho bất kỳ số nguyên nào trong khoảng từ 2 đến căn bậc hai của  $n$  hay không. Nếu có,  $n$  không phải là số nguyên tố.

- Sau khi định nghĩa hàm "isPrime", tôi sử dụng hàm "numpy.unique" để tìm tất cả các giá trị duy nhất trong ma trận A và lưu chúng vào một mảng NumPy có tên là A\_unique. Tiếp theo, tôi khởi tạo một danh sách rỗng có tên là "prime\_vector", sẽ được sử dụng để lưu các số nguyên tố tìm thấy trong ma trận A.
- Sau đó, tôi sử dụng một vòng lặp để kiểm tra từng phần tử trong mảng A\_unique. Nếu một phần tử là số nguyên tố, tôi thêm nó vào danh sách "prime\_vector".
- Cuối cùng, tôi in danh sách "prime\_vector" ra màn hình sử dụng lệnh "print".

#### **1.4 Câu f - Given a matrix, reverse elements in the odd rows of the matrix D, and print the resultant matrix to the screen.**

- Tính ma trận D = cách nhân ma trận B và C bằng hàm matmul trong thư viện numpy qua lệnh 'np.matmul(C, B)'
- In ma trận D và xuống dòng
- Tạo vector trống với tên gọi D\_res
- Tạo 1 vòng lặp cho i chạy trong các dòng của ma trận D qua lệnh 'D.shape[0]'
- Câu lệnh điều kiện kiểm tra nếu dòng i có là dòng lẻ hay không qua lệnh 'i % 2 != 0'
- Nếu có thì đảo ngược các phần tử trong dòng đó qua lệnh 'np.flipud(D[i, :])'.  
Trong lệnh 'D[i, :]', i là dòng được thay đổi theo vòng lặp, : là các cột được giữ nguyên. Sau đó lưu kết quả vào D\_res qua lệnh 'D\_res += [np.flipud(D[i, :])]'
- Câu lệnh rẽ nhánh 'else' nếu không thỏa điều kiện thì giữ nguyên và lưu kết quả vào D\_res qua lệnh 'D\_res += [D[i, :]]'
- Thay đổi cấu trúc của ma trận D\_res theo cấu trúc của ma trận D với 'D.shape[0]' cột và 'D.shape[1]' dòng và in kết quả D\_res ra màn hình.

### **1.5 Câu g: Regarding the matrix A, find the rows which have maximum count of prime numbers, and print the rows to the screen.**

- Tôi này định nghĩa một hàm "countPrime" để đếm số lượng số nguyên tố trong một dòng của ma trận. Sau đó, tôi sử dụng hàm "countPrime" để tìm số lượng số nguyên tố trong từng dòng của ma trận A.
- Hàm "countPrime" nhận đầu vào là một mảng NumPy đại diện cho một dòng của ma trận A. Hàm sử dụng một vòng lặp để kiểm tra từng phần tử trong dòng có phải là một số nguyên tố hay không bằng hàm "isPrime" mà tôi đã định nghĩa ở câu "E". Nếu một phần tử là số nguyên tố, hàm tăng biến đếm "count" lên một đơn vị. Cuối cùng, hàm trả về giá trị của biến đếm "count".
- Sau khi định nghĩa hàm "countPrime", tôi khởi tạo một danh sách rỗng có tên là "primeCounts", sẽ được sử dụng để lưu số lượng số nguyên tố trong từng dòng của ma trận A.
- Sau đó, tôi sử dụng một vòng lặp để duyệt qua từng dòng của ma trận A. Đối với mỗi dòng, tôi gọi hàm "countPrime" để đếm số lượng số nguyên tố trong dòng đó, và lưu kết quả vào danh sách "primeCounts".
- Tiếp theo, tôi tìm ra giá trị lớn nhất trong danh sách "primeCounts" bằng cách sử dụng hàm "max". Sau đó, tôi tìm tất cả các chỉ số của danh sách "primeCounts" có giá trị bằng giá trị lớn nhất và lưu chúng vào danh sách "max\_indices".
- Cuối cùng, tôi in ra màn hình tất cả các dòng trong ma trận A có số lượng số nguyên tố lớn nhất, sử dụng vòng lặp và lệnh "print".


### **1.6 Câu h: Regarding the matrix A, find the rows which have the longest contiguous odd numbers sequence, and print the rows to the screen.**

- Tạo một biến tên max\_len khởi tạo bằng 0 để lưu độ dài dãy số lẻ liên tiếp nhiều nhất

- Tạo một mảng tên `max_rows` rỗng để lưu dòng có phần tử lẻ liên tiếp nhiều nhất
- Cho vòng lặp `i` chạy trong các dòng của ma trận `A`
- Tạo một biến tên `cur_len` khởi tạo bằng 0 để lưu độ dài dãy số lẻ liên tiếp nhiều nhất tại thời điểm `i`
- Tạo một mảng tên `cur_seq` rỗng để lưu dòng có phần tử lẻ liên tiếp nhiều nhất tại thời điểm `i`
- Tạo một vòng lặp `j` chạy trong các cột của ma trận `A`
- Kiểm tra phần tử ma trận `A` dòng `i` cột `j` có lẻ hay không
- Nếu thỏa điều kiện thì lưu phần tử đó vào mảng `cur_seq` và `cur_len` tăng lên 1 đơn vị
- Nếu không thỏa thì các giá trị `cur_seq` và `cur_len` giữ nguyên giá trị
- Kiểm tra nếu `cur_len` có lớn hơn `max_len` không
- Nếu thỏa thì gán `max_len = cur_len` và gán `max_rows = dòng có số lẻ liên tiếp nhiều nhất`
- Nếu `cur_len = max_len` thì thêm chỉ số dòng vào cuối mảng `max_rows`
- Tạo vòng lặp và in kết quả dòng chứa số lẻ liên tiếp nhiều nhất ra màn hình.

## CHAPTER 2 - SOURCE CODES AND OUTPUTS

### 2.1 Hàm main



```
1  def main():
2      A_matrix = create_matrix_A()
3      print("Ma tran A:\n", A_matrix)
4      B_matrix = create_matrix_B()
5      print("Ma tran B:\n", B_matrix)
6      C_matrix = create_matrix_C()
7      print("Ma tran C:\n", C_matrix)
8      cau_A(A_matrix, B_matrix, C_matrix)
9      cau_B(A_matrix)
10     cau_C(A_matrix)
11     cau_D(A_matrix)
12     cau_E(A_matrix)
13     cau_F(C_matrix, B_matrix)
14     cau_G(A_matrix)
15     cau_H(A_matrix)
16
17
18  if __name__ == '__main__':
19      main()
```

## 2.2 Ma trận A

Ma trận A:

```
[[ 46  39  77  97  93  50  71  22  83  26]
 [ 36  66  44  13  94  39  97  93  80  70]
 [ 15  55   5  58  54  82  69   6  77   1]
 [100  95  56   4 100  62  42  49  30  98]
 [  2  61  32  43   2  88  91  92 100  94]
 [ 49  65  99  13  32  13  99  21 100  50]
 [ 25  68  56  60  57  90  20  31  60  26]
 [ 66   6  70  23  61   1  56  75  58   2]
 [ 23  80  44  19  35  33  47  57  82   8]
 [ 95  20  10  63  21  44  98  26  24  67]]
```

## 2.3 Ma trận B

Ma trận B:

```
[[20 14  8 16 13 16  4  9  7  7]
 [ 8  4 18  4  1  2 19 12 11 20]]
```

## 2.4 Ma trận C

Ma trận C:

```
[[ 1  8]
 [ 2  6]
 [ 6  7]
 [ 7 20]
 [ 5  9]
 [19  4]
 [17  2]
 [14 19]
 [15  1]
 [13  7]]
```

## 2.5 Câu a



```

1  def cau_A(A, B, C):
2      A1 = np.add(A, A.T)
3      A2 = np.add(np.matmul(C, B), np.matmul(B.T, C.T))
4      A_result = np.add(A1, A2)
5      print("\nKet qua cau A:")
6      print(A_result)

```

Ket qua cau A:

```

[[260 209 420 545 288 543 608 625 509 604]
 [209 236 335 342 293 430 533 461 454 434]
 [420 335 358 654 373 515 454 668 378 423]
 [545 342 654 392 370 547 790 675 562 846]
 [288 293 373 370 152 469 562 507 465 506]
 [543 430 515 547 469 650 617 503 552 529]
 [608 533 454 790 562 617 252 681 327 468]
 [625 461 668 675 507 503 681 858 569 707]
 [509 454 378 562 465 552 327 569 396 325]
 [604 434 423 846 506 529 468 707 325 596]]

```

## 2.6 Câu b



```
1 def cau_B(A):
2     A_res = np.zeros((A.shape[0], A.shape[1]))
3     for i in range(10, 20):
4         A_res += np.linalg.matrix_power((A/float(i)), i - 9)
5     print("\nKet qua cau B:")
6     print(A_res)
```

Ket qua cau B:

```
[[2.37022157e+13 3.25488448e+13 2.80229033e+13 2.17819343e+13
 3.02923866e+13 2.88846401e+13 3.82367898e+13 2.76792563e+13
 4.06401202e+13 2.37132113e+13]
[2.35548856e+13 3.23465301e+13 2.78487208e+13 2.16465402e+13
 3.01040961e+13 2.87050995e+13 3.79991139e+13 2.75072113e+13
 4.03875117e+13 2.35658112e+13]
[1.67523204e+13 2.30049714e+13 1.98061127e+13 1.53951031e+13
 2.14101458e+13 2.04151747e+13 2.70251161e+13 1.95632340e+13
 2.87237530e+13 1.67600907e+13]
[2.50048639e+13 3.43376923e+13 2.95630083e+13 2.29790473e+13
 3.19572194e+13 3.04721112e+13 4.03382310e+13 2.92004702e+13
 4.28736466e+13 2.50164641e+13]
[2.20525525e+13 3.02834648e+13 2.60725248e+13 2.02659307e+13
 2.81840561e+13 2.68742913e+13 3.55755350e+13 2.57528004e+13
 3.78115976e+13 2.20627829e+13]
[2.00652263e+13 2.75543896e+13 2.37229310e+13 1.84395993e+13
 2.56441704e+13 2.44524344e+13 3.23695400e+13 2.34320164e+13
 3.44040979e+13 2.00745338e+13]
[1.92801969e+13 2.64763534e+13 2.27947954e+13 1.77181809e+13
 2.46408728e+13 2.34957664e+13 3.11031240e+13 2.25152654e+13
 3.30580795e+13 1.92891401e+13]
[1.54567781e+13 2.12258827e+13 1.82744079e+13 1.42045152e+13
 1.97543897e+13 1.88363632e+13 2.49351205e+13 1.80503106e+13
 2.65023975e+13 1.54639469e+13]
[1.60400295e+13 2.20268302e+13 1.89639827e+13 1.47405191e+13
 2.04998125e+13 1.95471461e+13 2.58760381e+13 1.87314337e+13
 2.75024570e+13 1.60474680e+13]
[1.87692603e+13 2.57747025e+13 2.21907084e+13 1.72486348e+13
 2.39878642e+13 2.28731069e+13 3.02788611e+13 2.19185766e+13
 3.21819974e+13 1.87779707e+13]]
```



## 2.7 Câu c



```
1  def cau_C(A):  
2      A_odd = A[1::2]  
3      print("\nKet qua cau C:")  
4      print("Odd vector:")  
5      print(A_odd)
```

Ket qua cau C:

Odd vector:

```
[[ 36  66  44  13  94  39  97  93  80  70]  
 [100  95  56   4 100  62  42  49  30  98]  
 [ 49  65  99  13  32  13  99  21 100  50]  
 [ 66   6  70  23  61   1  56  75  58   2]  
 [ 95  20  10  63  21  44  98  26  24  67]]
```

## 2.8 Câu d



```
1  def cau_D(A):
2      odd = []
3      for i in range(A.shape[0]):
4          for j in range(A.shape[1]):
5              if A[i][j] % 2 != 0:
6                  odd.append(A[i][j])
7      print("\nKet qua cau D:")
8      print(odd)
```

Ket qua cau D:


[39, 77, 97, 93, 71, 83, 13, 39, 97, 93, 15, 55, 5, 69, 77, 1, 95, 49, 61, 43, 91, 49, 65, 99, 13, 13, 99, 21, 25, 57, 31, 23, 61, 1, 75, 23, 19, 35, 33, 47, 57, 95, 63, 21, 67]

## 2.9 Hàm isPrime()



```
1 def isPrime(n):  
2     if n < 2:  
3         return False  
4     for i in range(2, int(np.sqrt(n)) + 1):  
5         if n % i == 0:  
6             return False  
7     return True
```

## 2.10 Câu e



```
1  def cau_E(A):
2      prime_matrix = []
3
4      for i in range(A.shape[0]):
5          for j in range(A.shape[1]):
6              if isPrime(A[i, j]):
7                  prime_matrix.append(A[i, j])
8
9      prime_vector = np.array(prime_matrix)
10     print("\nKet qua cau E:")
11     print("Prime vector:")
12     print(prime_vector)
```

```
Ket qua cau E:
Prime vector:
[97 71 83 13 97  5  2 61 43  2 13 13 31 23 61  2 23 19 47 67]
```

## 2.11 Câu f



```

1  def cau_F(C, B):
2      D = np.matmul(C, B)
3      D_res = []
4
5      for i in range(D.shape[0]):
6          if i % 2 != 0:
7              D_res += [np.flipud(D[i, :])]
8          else:
9              D_res += [D[i, :]]
10     print("\nKet qua cau F:")
11     print(np.reshape(D_res, (D.shape[0], D.shape[1])))

```

Ket qua cau F:

```

[[ 84  46 152  48  21  32 156 105  95 167]
 [134  80  90 122  44  32  56 124  52  88]
 [176 112 174 124  85 110 157 138 119 182]
 [449 269 303 408 152 111 192 416 178 300]
 [172 106 202 116  74  98 191 153 134 215]
 [213 177 219 152 312 251 320 224 282 412]
 [356 246 172 280 223 276 106 177 141 159]
 [478 307 354 417 262 201 300 454 272 432]
 [308 214 138 244 196 242  79 147 116 125]
 [231 168 201 185 222 176 236 230 210 316]]

```

## 2.12 Hàm countPrime()



```
1  def countPrime(row):  
2      count = 0  
3      for num in row:  
4          if isPrime(num):  
5              count += 1  
6      return count
```

## 2.13 Câu g



```

1  def cau_G(A):
2      primeCounts = []
3
4      for row in A:
5          count = countPrime(row)
6          primeCounts.append(count)
7
8      max_count = max(primeCounts)
9      max_indices = []
10     for i in range(len(primeCounts)):
11         count = primeCounts[i]
12         if count == max_count:
13             max_indices.append(i)
14     print("\nKet qua cau G:")
15     print("Rows have maximum count of prime numbers:")
16
17     for i in max_indices:
18         print(A[i])

```

```

Ket qua cau G:
Rows have maximum count of prime numbers:
[ 2  61  32  43   2  88  91  92 100  94]

```

## 2.14 Câu h



```
1  def cau_H(A):
2      max_len = 0
3      max_rows = []
4      for i in range(A.shape[0]):
5          cur_len = 0
6          cur_seq = []
7          for j in range(A.shape[1]):
8              if A[i][j] % 2 == 1:
9                  cur_seq.append(A[i][j])
10                 cur_len += 1
11             else:
12                 cur_seq = []
13                 cur_len = 0
14                 if cur_len > max_len:
15                     max_len = cur_len
16                     max_rows = [i]
17                 elif cur_len == max_len:
18                     max_rows.append(i)
19     print("\nKet qua cau H:")
20     for i in max_rows:
21         print(A[i])
```



Ket qua cau H:

[23 80 44 19 35 33 47 57 82 8]