TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TIỂU LUẬN GIỮA KÌ MÔN ĐẠI SỐ TUYẾN TÍNH**

BÁO CÁO THỰC HÀNH

Người hướng dẫn: NGUYỄN VĂN KHOA

Người thực hiện: **PHẠM ĐẶNG THANH TRUNG**

MSSV **: 522H0148**

Lớp **: 22H50302**

Khoá  **: 26**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

LỜI CẢM ƠN

Đầu tiên, trước khi vào bài báo cáo, em xin được gửi lời chân thành cảm ơn đến với trường đại học Tôn Đức Thắng vì đã đầu tư cơ sở vật chất hạ tầng thiết bị hỗ trợ tốt giúp cho em có được điều kiện học tốt nhất ở môn học Đại Số Tuyến Tính. Đặc biệt, em xin gửi lời cảm ơn sâu sắc đến giảng viên thực hành bộ môn-Thầy Nguyễn Văn Khoa đã dạy và truyền đại nhiều kiến thức lí thuyết quý báo đến em trong thời gian học kì qua. Nhờ thầy, em đã dần hiểu hơn về ngôn ngữ lập trình python, trước đó em rất lười khi làm bài tập liên quan đến python, vì nó khá là khó với em, nhưng nhờ thầy chỉ dẫn chi tiết từng bài mà em đã có thể hiểu bài nhanh hơn, nhiều hơn. Dần yêu mến đối với ngôn ngữ lập trình này.

Môn học này không những đòi hỏi người học những kiến thức về toán học mà còn phải có khả năng vận dụng hàm chức năng, phương pháp giải quyết bài toán, tìm được cách nhanh và tối ưu nhất. Thông qua bài báo cáo này,em mong rằng mình có thể cải thiện khả năng tư duy logic, có thêm được kinh nghiệm cho bài báo cáo sau này.

Em là 1 sinh viên năm nhất, chỉ mới bước chân vào thế giới đại học đồng thời đây cũng là bài luận đầu tiên trong cuộc đời em. Kiến thức cũng còn hạn chế,tuy em sẽ cố gắng hết sức,nhưng bài làm chắc chắc sẽ có sai sót, mắc lỗi lầm. Kính mong thầy cô xem xét và góp ý để bài luận em được hoàn thiện hơn.

Lời cuối, em kính chúc thầy cô nhiều sức khoẻ, luôn thành công trên con đường sự nghiệp giảng dạy của mình.

**BÁO CÁO ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Em xin cam đoan đây là bài báo cáo của riêng em và được sự hướng dẫn của Thầy Nguyễn Văn Khoa. Các nội dung nghiên cứu, kết quả trong bài 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 em giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong bài báo cáo 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 em xin hoàn toàn chịu trách nhiệm về nội dung bài báo cáo 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 23 tháng 4 năm 2023*

*Tác giả*

*Trung*

*Phạm Đặng Thanh Trung*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Bài báo cáo được chia làm 2 chương chính là chương I và chương II.

Ở chương I, đây chính là phần giải thích và trình bày phương pháp làm bài. Đề bài bao gồm 7 câu, mỗi câu với mỗi nhiệm vụ khác nhau. Chúng tập chung vào việc xử lí, các bài toán thao tác ma trận, vecto. Các nhiệm vụ trong bài yêu cầu sinh viên phải nắm vững các kiến thức được học qua các bài lab các tuần, bài báo cáo giúp chúng ta ôn lại cách khai báo thư viện numpy tạo list,ma trận, các phép toán với ma trận như cộng trừ hai ma trận, phép nhân chia ma trận với một số, cách dùng vòng lặp để xử lí bài toán, cách thêm dòng/cột dô 1 ma trận mới, ma trận chuyển vị,… Thông qua đó, chúng ta còn tìm ra được nhiều phương pháp giải logic hiệu quả khác nhau, đó là những kinh nghiệm cần thiết cho các bai báo cáo thực hành sau này.

Ở chương II, đây là phần trình bày bài code và kết quả của bài bằng ảnh chụp.

MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc133512589)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN 3](#_Toc133512590)

[TÓM TẮT 4](#_Toc133512591)

[MỤC LỤC 5](#_Toc133512592)

[CHƯƠNG I: PHƯƠNG PHÁP GIẢI BÀI TẬP THỰC HÀNH 6](#_Toc133512593)

[1.1 A) Tính 6](#_Toc133512594)

[1.2 B) Tính 7](#_Toc133512595)

[1.3 C) Lưu tất cả hàng lẻ của ma trận A vào 1 ma trận mới 8](#_Toc133512596)

[1.4 D) Lưu tất cả số nguyên lẻ trong ma trận A vào 1 vecto mới 9](#_Toc133512597)

[1.5 E) Tìm tất cả số nguyên tố trong ma trận A rồi lưu vào vecto mới 10](#_Toc133512598)

[1.6 F) Cho D=BC, đảo ngược phần tử trong hàng lẻ của ma trận D 12](#_Toc133512599)

[1.7 G) Tìm hàng có số lượng số nguyên tố nhiều nhất của ma trận A 13](#_Toc133512600)

[1.8 H) Tìm hàng của ma trận A có số lượng số lẻ liền kề nhiều nhất 14](#_Toc133512601)

[CHƯƠNG II: BÀI CODE VÀ KẾT QUẢ 17](#_Toc133512602)

[2.1 CÂU A 17](#_Toc133512603)

[2.2 CÂU B 18](#_Toc133512604)

[2.3 CÂU C 19](#_Toc133512605)

[2.4 CÂU D 19](#_Toc133512606)

[2.5 CÂU E 20](#_Toc133512607)

[2.6 CÂU F 21](#_Toc133512608)

[2.7 CÂU G 22](#_Toc133512609)

[2.8 CÂU H 23](#_Toc133512610)

CHƯƠNG I: PHƯƠNG PHÁP GIẢI BÀI TẬP THỰC HÀNH

1. A) Tính
2. Vào bài code, đầu tiên phải khai báo thư viện numpy, để có thể dùng nhiều hàm chức năng. Theo đề bài đầu tiên chúng ta phải tạo ra 3 ma trận khác nhau gồm A,B,C. Để tạo chúng với các giá trị là ngẫu nhiên thì chúng ta dùng hàm np.random,randint( low, high, size, dtype**),** hàm giúp tạo ma trận với các phần tử ngẫu nhiên từ low đến high với kích thước bằng size.
3. Dùng các phép tính toán ma trận:

* Trong bài để nhân ma trận chúng ta dùng hàm “np.matmul(C,B)” với C,B là 2 ma trận khác nhau, với điều kiện số lượng cột ma trận C phải bằng số lượng hàng trong ma trận B. Đối với cộng trừ ma trận chúng ta chỉ cần dúng các dấu tính toán toán học cơ bản như “+”, “-”.

1. Ma trận chuyển vị:

* Cách kiểu diễn phép ma trận chuyển vị chúng ta dùng “.T”, ví dụ ma trận chuuyển vị của A là “A.T”, của B là “B.T”, của C là “C.T”

1. Tính toán và lưu kết quả vào biến:

* Với các phương pháp trên chúng ta đã dề dàng biểu diễn và tính ra kết quả, để tính tổng chúng ta sẽ tạo 1 biến lưu kết quả là “ca”:

ca=A+A.T+np.matmul(C,B)+np.matmul(B.T,C.T)

1. In kết quả ra màn hình:

- Để in kết quả ra màn hình ta dùng hàm print():

print(ca)

1. B) Tính
2. Xác định quy luật, công thức chung:

* Đầu tiên chúng ta xác định quy luật: xem xét thấy mẫu và số mũ mỗi phần tử trong dãy tăng dần,. mỗi lần chúng tăng 1 đơn vị số. Qua dó chúng ta rút ra được công thức chung là A/((9+n) tất cả mũ n.

1. Dùng vòng lặp for:

* Sau khi tìm ra quy luật, với phép tính dãy các quy luật tăng dần, chúng ta sẽ dùng vòng lặp “for biến\_lặp in chuỗi\_lặp”. Đối với bài này ta dùng chuỗi\_lặp là hàm range(). Hàm range là hàm được tính hợp sẵn trong python do đó không cần khai báo thêm thư viện. cú pháp hàm range() là range(stop) hoặc range(start,stop,step) với start là số nguyên bắt đầu, stop là số nguyên kết thúc, chỗi sẽ kết thúc với tham số này, step là bước nhảy khoảng cách trong chuỗi, nếu không khai báo vào thì mặc định sẽ là 1. Trong bài ta có thể nhận thấy nó bắt đầu từ 1 đến 10 vì thế ta sẽ tiến hành lặp từ 1 đến 10 cụ thể như sau:

For k in range(1,11):

* Nhờ hàm range() chúng ta có thể thực hiện được phép tính dãy có quy luật, tính tống chúng ta sẽ khai báo biến cb=0, nếu không cho nó bằng 0 đầu tiên trước khi vào vòng lặp thì kết quả có thể bị sai sót. Chúng ta lập công thức tính như sau:

cb = cb + (A/(9+k))\*\*k

1. In kết quả ra màn hình:

* Ra được kết quả của cb (tức là tổng của dãy) chúng ta tiến hành dùng print() để in kết quả ra màn hình theo yêu cần bài toán:

print(cb)

1. C) Lưu tất cả hàng lẻ của ma trận A vào 1 ma trận mới
2. Tạo list trống để lưu kết quả:

* Đầu tiên chúng ta sẽ tạo list cmatrix trống để lưu kết quả vào:

cmatrix=[]

1. Kết hợp vòng lặp for với điều kiện if else để tìm ra kết quả:

* Để tìm và lấy các hàng lẻ của A ra, chúng ta phải tạo nên 1 vòng lặp để chạy đến hết các hàng của ma trận A bằng cách dùng hàm “range()”, cụ thể bắt đầu hàng của 1 ma trận sẽ là 0, kết thúc sẽ là kích thước hàng của nó, để lấy kích thước ma trận, với tư cách là 1 người lập trình chúng ta không được ghi trực tiếp số đếm vào mà phải dùng “shape”,nó có tác dụng lấy số phần tử của mỗi chiều, với shape[0] là hàng, shape[1] là cột. Vì theo đề ta sẽ tìm hàng nên sẽ dùng nó vào ma trận như sau:

A.shape[0]

* Kết hợp hàm “range()” ta sẽ ra được:

for i in range(0,A.shape[0]):

* Tiếp theo đó để tìm hàng lẻ ta sẽ dùng phép if với điều kiện cụ thể để lọc ra:

if(i%2==0):

* Để lưu 1 hàng vào 1 ma trận mới chúng ta sẽ dùng hàm “np.append” cụ thể:

cmatrix.append(A[i,:])

* Với cách ghi như thế này, chúng ta có thể lưu toàn bộ giá trị các cột của hàng i vào cmatrix:

for i in range(0,A.shape[0]):

if(i%2==0):

cmatrix.append(A[i,:])

* Sau đó chúng ta sẽ chuyển nó về dạng ma trận ta dùng “np.array()”:

vc=np.array(cmatrix)

1. In kết quả ra màn hình:

* Sau khi ra được kết quả, ta sẽ dùng hàm “print(vc)” để in kết quả ra màn hình theo yêu cầu đề bài.

1. D) Lưu tất cả số nguyên lẻ trong ma trận A vào 1 vecto mới
2. Tạo list trống để lưu kết quả

* Đầu tiên ta sẽ tạo ra list trống vd:

vd=[]

1. Kết hợp vòng lặp for với điều kiện if else và 1 số thao tác trong ma trận:

* Để có thể kiểm tra từng phần tử trong ma trận A, ta sẽ cho chạy giá trị từng hàng và cột vì thế ta sẽ dùng 2 lần “for in range()”, với for thứ nhất là hàng, for thứ 2 là cột:

for i in range(0,A.shape[0]):

Đây là hàm để chạy từng phần tử của hàng

for j in range(0,A,shape[1]):

Đây là hàm để chạy từng phần tử của cột

* Khi chạy từng phần tử ta sẽ thêm vào điều kiện if để kiểm tra số lẻ:

if(A[i][j]%2!=0):

(với i là số hàng, j là số cột)

* Nếu điều kiện đúng ta sẽ thêm nó vào “vd” bằng hàm append:

vd.append(A[i][j])

* Sau khi tìm được tất cả số lẻ lưu vào “vd” thì ta sẽ dùng hàm “np.array()” để chuyển nó vè dạng vecto:

newvecto=np.array(vd)

1. In kết quả ra màn hình:

* Chúng ta dùng “print()” để in kết quả ra màn hình theo yêu cầu đề bài:

print(newvecto)

1. E) Tìm tất cả số nguyên tố trong ma trận A rồi lưu vào vecto mới
2. Tạo list trống để lưu kết quả:

* Đầu tiên ta sẽ tạo nên 1 list trống là “ve”:

Ve=[]

1. Tạo hàm để kiểm tra số nguyên tố:

* Chúng ta sẽ tạo nên 1 hàm “check\_prime\_number” để kiểm tra số nguyên tố:

def check\_prime\_number(num):

* Chúng ta đều biết số nguyên tố là số tự nhiên lớn hơn 1 và không phải là tích của 2 số nhỏ hơn nó. Do đó để viết hàm “check\_prime\_number” ta sẽ xét nó 2 điều kiện tương ứng 2 lần “if”, lần “if” thứ nhất t sẽ xét nếu nó bé hơn hoặc bằng 1 thì sẽ sẽ trả về sai, nó không phải là số nguyên tố:

if(num<=1):

return false

* Còn “if” thứ hai ta sẽ kiểm tra xem nó có ước hay không, để kiểm tra ước t sẽ dùng “for” để kiểm tra các số bé hơn nó, để nhanh gọn ta chỉ chạy từ 2 đến một nửa lần nguyên của nó:

for i in range(2,int(num\*\*0.5+1)):

if(num%2==0):

return flase

* Nếu con số “num” ta truyên vào dều vượt qua được 2 lần kiểm tra thì chắc chắc nó chính là số nguyên tố vì thế ta sẽ trả về true:

return true

1. Dùng vòng lặp for 2 lần kết hợp if else để kiểm tra từng phần tử của ma trận:

* Để có thể kiểm tra từng phần tử trong ma trận A, ta sẽ cho chạy giá trị từng hàng và cột vì thế ta sẽ dùng 2 lần “for in range()”, với for thứ nhất là hàng, for thứ 2 là cột:

for i in range(0,A.shape[0]):

Đây là hàm để chạy từng phần tử của hàng

for j in range(0,A,shape[1]):

Đây là hàm để chạy từng phần tử của cột

* Khi chạy từng phần tử ta sẽ thêm vào điều kiện if để kiểm tra phần tử đó có phải số nguyên tố hay không:

if(check\_prime\_number(A[i][j])):

* Nếu điều kiện đúng ta sẽ lưu A[i][j] vào “ve” bằng hàm append:

ve.append(A[i][j])

* Sau khi hoàn thành lưu kết quả, chúng ta tiến hành chuyển nó về dạng vecto bằng hàm “np.array”:

ve=ve.np.array(ve)

1. In kết quả ra màn hình:

* Sau đó ta tiến hành dùng “print()” để in kết quả ra màn hình theo yêu cầu đề bài:

print(ve)

1. F) Cho D=BC, đảo ngược phần tử trong hàng lẻ của ma trận D
2. Tạo list trống để lưu kết quả:

* Đầu tiên ta sẽ tạo 1 list trống “newD” bằng cách:

newD=[]

1. Dùng hàm matmul để nhân ma trận:

* Để tìm D, ta tiến hành hành nhân 2 ma trận C và B bằng hàm “np.matmul()”:

D=np.matmul(C,B)

1. Kết hợp vòng lặp for với 1 số thao tác trong ma trận để tìm kết quả:

* Để có thể kiểm tra từng phần tử trong hàng của ma trận D, vì thế ta sẽ dùng “for i in range(D.shape[0])”, để kiểm tra hàng lẻ ta dùng “if” với điều kiện “i%2==0”. Nếu điều kiện đúng ta sẽ dùng hàm “np.flipud()” để đảo ngược phần tử của hàng rồi chuyển nó vào newD:

newD=newD+ [np.flipud(D[i,:])]

* Nếu điều kiện sai ta sẽ lưu hàng bình thường vào newF:

newD=newD+[D[i,:]]

* Sau khi có được newD ta sẽ tiến hành định dạng lại nó với kích thước 10x10 cho phù hợp bằng hàm “np.reshape()”:

newD=np.reshape(newD,(10,10))

1. In kết quả ra màn hình:

* Sau khi có được kết quả newD ta in nó ra màn hình bằng hàm “print()”:

print(newD)

1. G) Tìm hàng có số lượng số nguyên tố nhiều nhất của ma trận A
2. Tạo ra list trống để lưu kết quả:

* Đầu tiên chúng ta phải tạo ra 1 list trống là tempcountrow:

temcountrow=[]

1. Dùng vòng lặp for kết hợp điều kiện if else và hàm kiểm tra số nguyên tố đã làm để tìm ra và đếm số số nguyên tố trong ma trận:

* Chúng ta sẽ đi đếm só lượng số nguyên tố nhiều nhất mỗi hàng, do đó ta sẽ dùng biến đếm “count” , khi bước vào vòng chạy hàng:

for i in range(A.shape[0])

* Thì đầu tiên count sẽ về 0, sau đó tiền hành chạy từng cột của hàng:

for i in range(A.shape[1])

* Trong đây, chúng ta sẽ dùng if với điều kiện là kiểm tra số nguyên tố:

if(check\_prime\_number(A[i][j])

* Nếu điều kiện đúng chúng ta tiến hành tăng biến đếm là count lêm 1, cứ thế đếm hết cột của hàng rồi ta sẽ được số lượng số nguyên tố của hàng trong “count”, rồi ta tiến hành lưu “count” vào “tempcountrow” bằng hàm “append()”:

tempcountrow.append(count)

* Cứ thế ta tiến lặp lại từng hàng, mỗi khi lặp lại count sẽ về 0, giá trị được làm mới. Qua đó toàn độ số lượng số nguyên tố mỗi hàng đều được lưu vào “tempcountrow” số phần tử tương ứng với số hàng.

1. Tiếp tục dùng vòng lặp for và if else để in kết quả ra màn hình:

* Chúng ta tiến hành chạy từng phần tử của “tempcountrow” bằng:

for i in range(0,len(tempcountrow))

* Trong đây chúng ta thêm if với điều kiện “tempcountrơ[i]==max” với max=np.max(tempcountrow), hàm “np.max” có tác dụng trả về giá trị lớn nhất trong tempcountrow. Nếu điều kiện đúng chúng ta tiến hành in nó ra mình hình bằng hàm “print(A[i,:])”:

max=np.nax(tempcountrow)

for i in range(0,len(tempcountrow)):

if(tempcountrow[i]==max):

print(A[i,:])

1. H) Tìm hàng của ma trận A có số lượng số lẻ liền kề nhiều nhất
2. Tạo list trống để lưu kết quả:

* Đầu tiên chúng ta tiến hành tạ 1 list trống là “count\_element”:

count\_element=[]

1. Dùng vòng lặp for kết hợp điều kiện if else để xây dựng nên thuật toán:

* Bài toán khá phức tạp, nó yêu cầu kết hợp cả đếm số lẻ lẫn phải liền kề nhau, vì thế ta sẽ tạo 2 biến đếm 1 biến đếm số lẻ liên tiếp tạm thời là “count” và 1 biến lưu số lượng số lẻ liền kề lớn nhất của 1 hàng đó là “maxodd\_row”. Chúng ta đều khai báo chúng bằng 0:

count=0

maxodd\_row=0

* Những biến đếm này mỗi hàng đều dùng lại, nên ta sẽ khai báo chúng sau khi ta tiến hành chạy hàng:

for i in range(A.shape[0]):

count=0

maxodd\_row=0

* Sau dó ta tiến hành chạy từng cột, để kiểm tra số lẻ ta dùng điều kiện “if” với điều kiện “A[i][j]%2!=0”, nếu điều kiện đúng ta sẽ tăng count lên 1 dơn vị: count=count+1, còn nếu điều kiện sai ta sẽ xét tiếp “if” với điều kiện “maxodd\_max<=count” đúng thì “maxodd\_max=count”, điều kiện này dùng để so sánh số số lẻ liền kề vừa tìm được với với số số lẻ liền kề lớn nhất, nếu nó bẻ hơn ta coi count vừa tìm được là lớn nhất. Cứ thế khi chạy hết các cột ta sẽ tiến hành 1 lần “if” tiếp với điều kiện “maxodd\_max<=count”, ta phải kiểm tra lại lần nữa là vì, ở lần đếm số liền kề cuối cùng chưa chắc đã được kiểm tra.
* Sau khi kiểm tra xong ta sẽ lưu kết quả vào “count\_element” bằng hàm “append()”.

1. Dùng vòng lặp for kết hợp điều kiện if else để in kết quả ra màn hình:

* Sau khi hoàn tất tìm số số lẻ liền kề nhiều nhất mỗi hàng lưu vào “count\_element” ta sẽ chạy từng phần tử trong nó so sánh để tìm ra hàng có số lượng lớn nhất rồi in ra màn hình bằng hàm “print()” cụ thể:

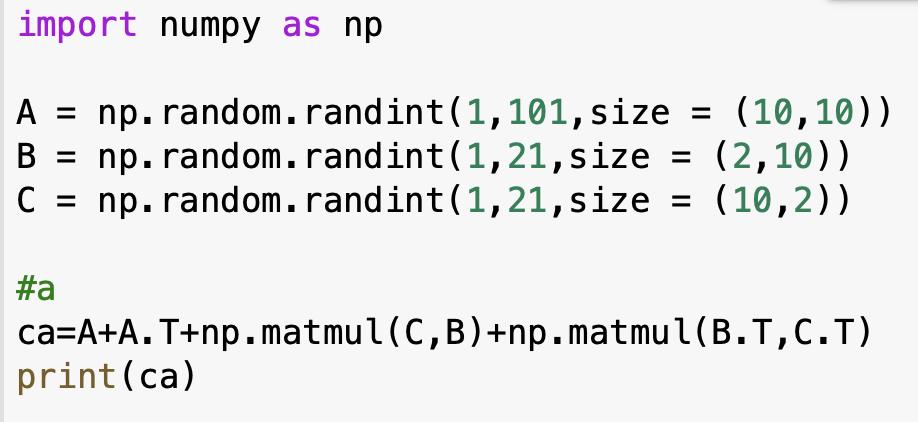
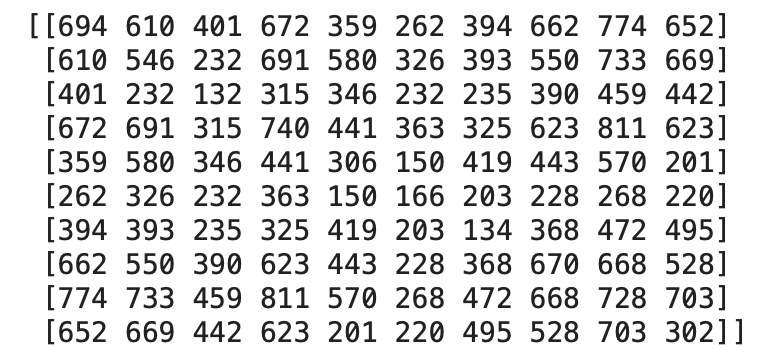
for i in range(0,len(count\_element)):

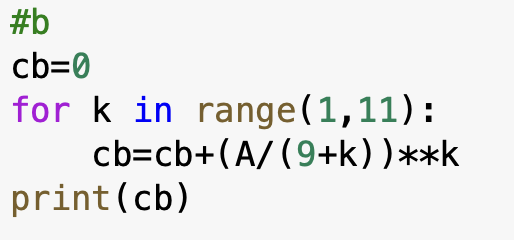
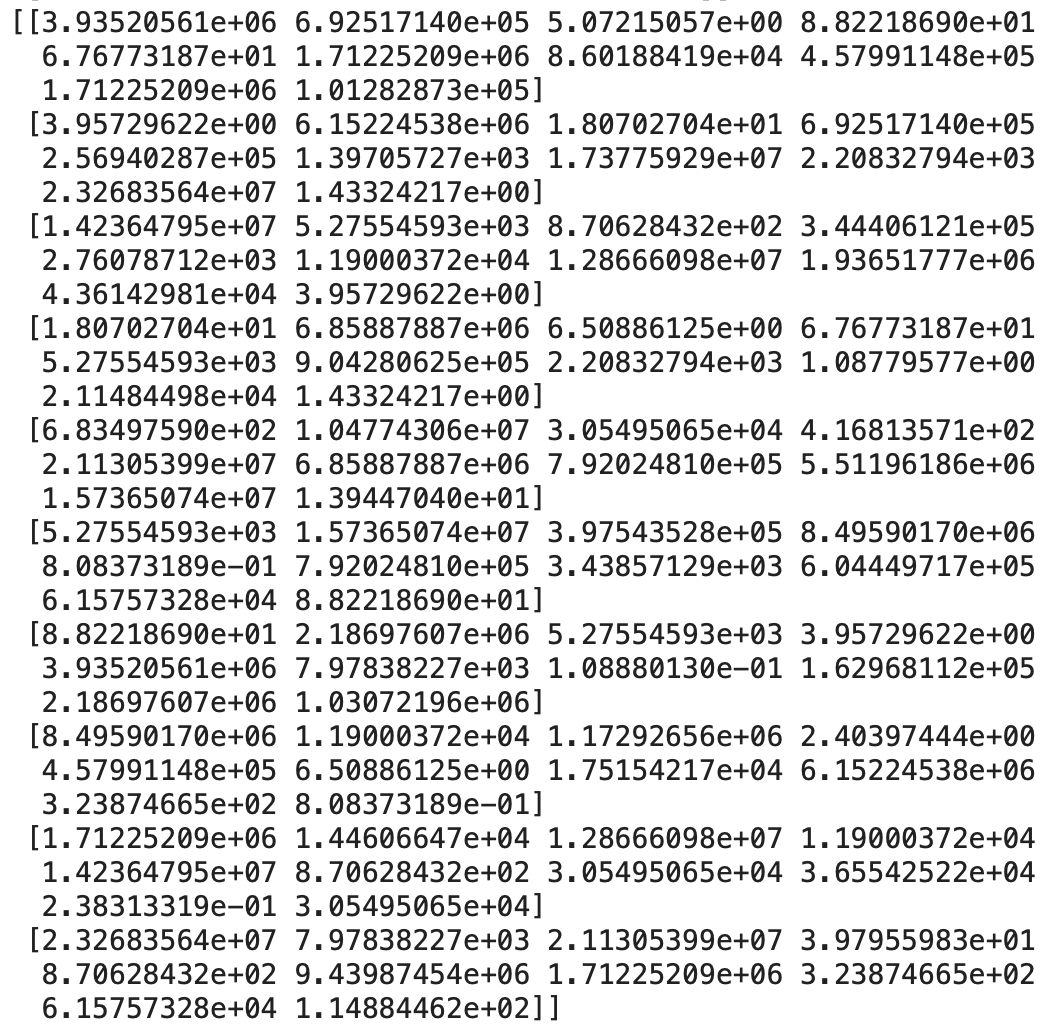
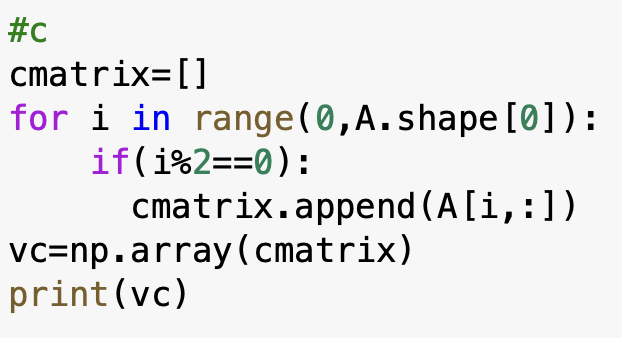
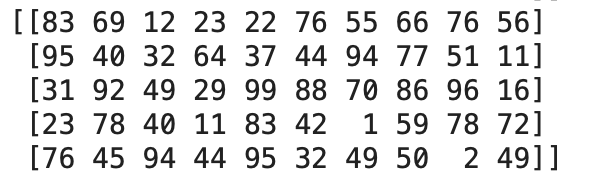
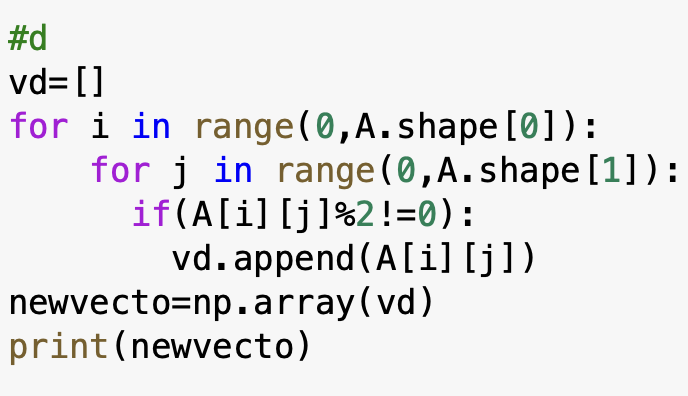
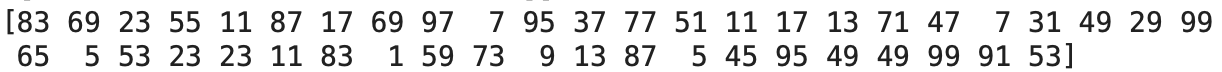
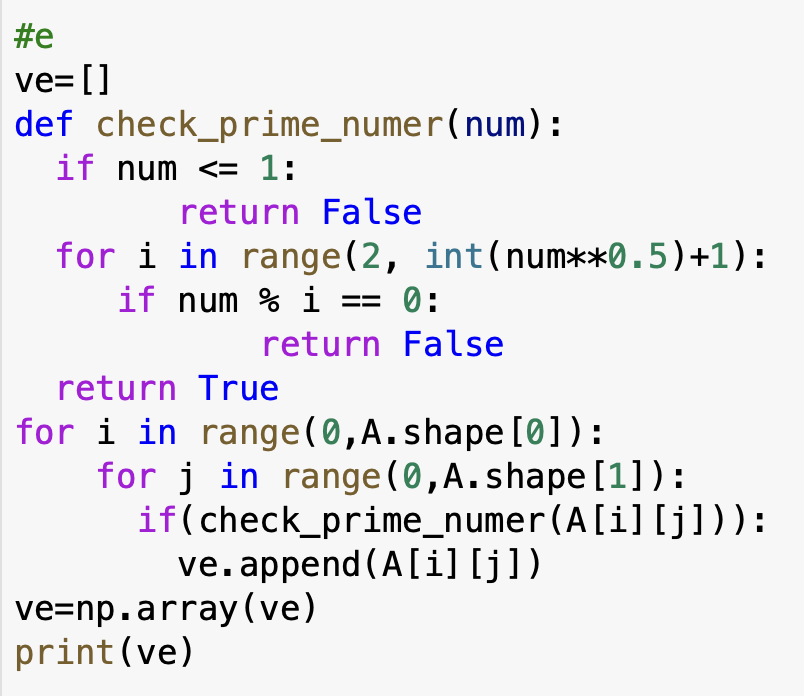
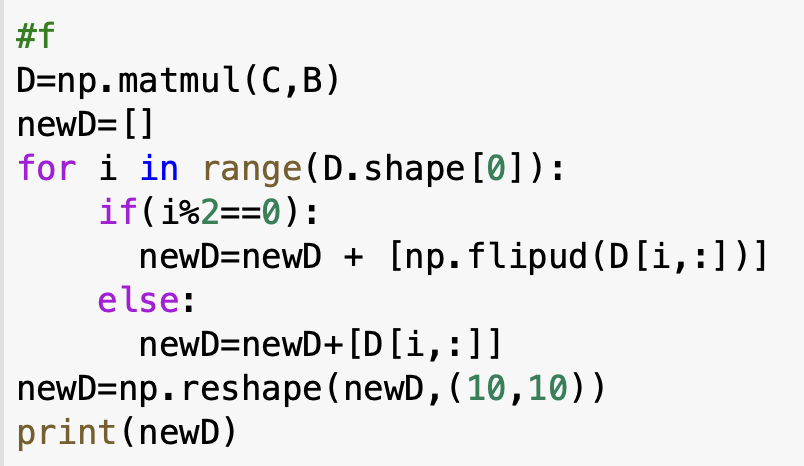
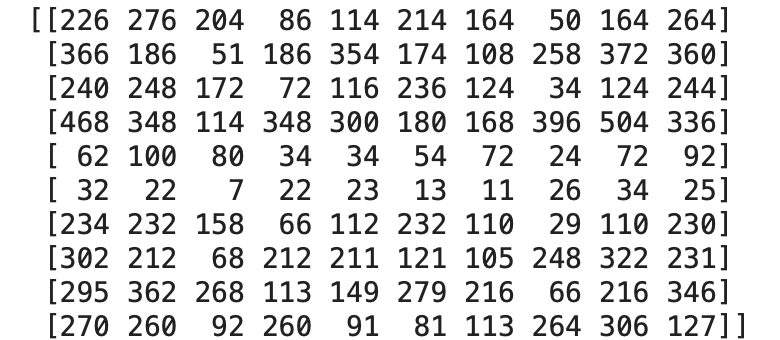
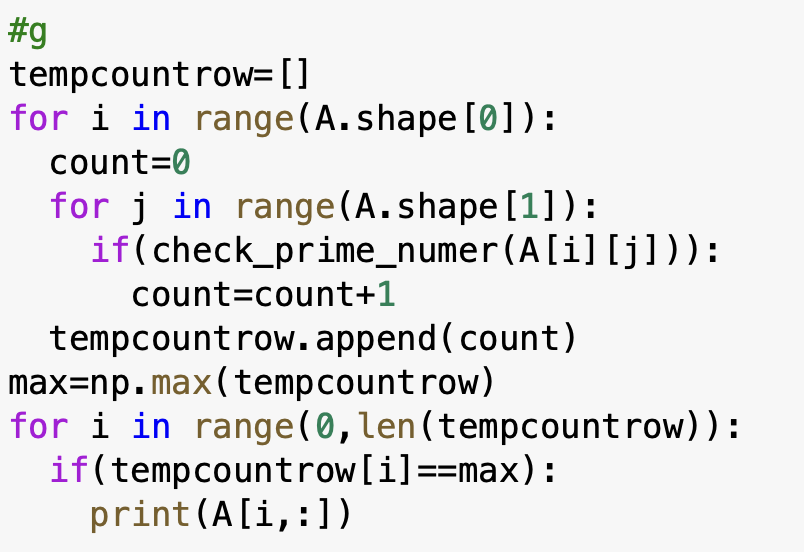
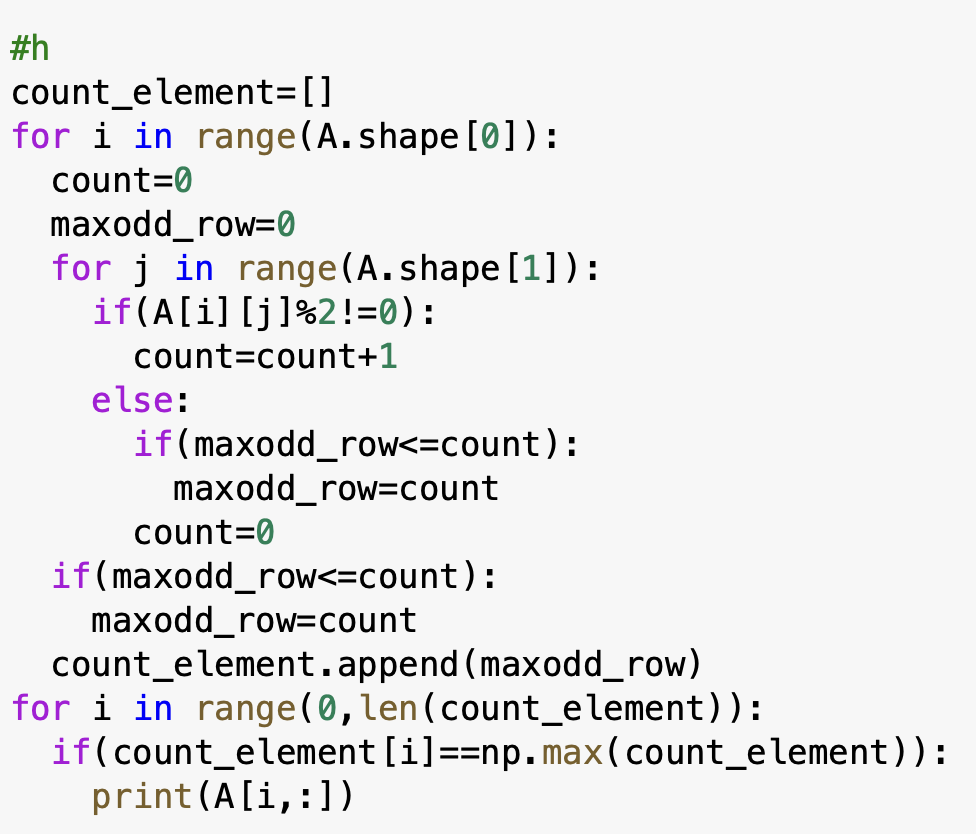
if(counr\_elêmnt[i]==np.max(count\_element)):

print(A[i,:])

CHƯƠNG II: BÀI CODE VÀ KẾT QUẢ

* 1. CÂU A

1. Bài code: 
2. Kết quả bài code:

1. CÂU B
2. Bài code: 
3. Kết quả bài code: 
4. CÂU C
5. Bài code: 
6. Kết quả bài code: 
7. CÂU D
8. Bài code: 
9. Kết quả bài code: 
10. CÂU E
11. Bài code: 
12. Kết quả bài code: 
    1. CÂU F
13. Bài code: 
14. Kết quả bài code: 
15. CÂU G
16. Bài code: 
17. Kết quả bài code: 
18. CÂU H
19. Bài code: 
20. Kết quả bài code: 