TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN/ĐỒ ÁN GIỮA KÌ II MÔN**

**ĐẠI SỐ TUYẾN TÍNH**

**BÀI TIỂU LUẬN MÔN THỰC HÀNH**

**ĐẠI SỐ TUYẾN TÍNH**

*Người hướng dẫn*: **Th.S PHẠM QUỐC DUY**

*Người thực hiện*: **HUỲNH HOÀNG TIẾN ĐẠT – 52200023**

Lớp**: 22050201**

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 TẬP LỚN/ĐỒ ÁN GIỮA KÌ II MÔN**

**ĐẠI SỐ TUYẾN TÍNH**

**TIỂU LUẬN THỰC HÀNH**

**ĐẠI SỐ TUYẾN TÍNH**

Người hướng dẫn: **Th.S PHẠM QUỐC DUY**

Người thực hiện: **HUỲNH HOÀNG TIẾN ĐẠT – 52200023**

Lớp**: 22050201**

Khoá**: 26**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

LỜI CẢM ƠN

Để hoàn thành đề tài, bài tiểu luận Python Giữa kì II môn Đại số tuyến tính thì lời đầu tiên, em xin gửi lời cảm ơn chân thành của mình đến Ban giám hiệu trường Đại học Tôn Đức Thắng, quý thầy cô giáo giảng viên của khoa Công nghệ thông tin vì đã tạo điều kiện tốt nhất về cơ sở vật chất và hệ thống thư viện hiện đại, đa dạng đầy đủ các loại sách giúp em có thể trau đồi kiến thức, tìm kiếm thông tin, tư liệu để hoàn thành bài Tiểu luận này. Đây thực sự là một cơ hội tuyệt vời giúp cho nghề nghiệp của em trong tương lai rộng mở hơn khi được tiếp xúc với sự hiện đại, nhiều kiến thức.

Đặt biệt, em xin gửi lời cảm ơn của mình đến thầy Phạm Quốc Duy – Giảng viên thực hành môn Đại số tuyến tính, người đã luôn đồng hành, giúp đỡ em trong việc hoàn thành bài Tiểu luận. Từ những kiến thức thầy đã giảng dạy trên những giờ học để em có thể áp dụng những kiến thức đã được vào bài Tiểu luận lần này. Một lần nữa em xin chân thành cảm ơn thầy vì sự hỗ trợ của thầy ạ.

Vì kiến thức bản thân còn hạn chế nên trong quá trình giải quyết vấn đề, hoàn thành bài tiểu luận lần này em không tránh khỏi những sai sót, em kính mong nhận được những lời nhận xét, đóng góp ý kiến từ thầy ạ.

Em xin chân thành cảm ơn thầy.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của Th.S Phạm Quốc Duy;. 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 15 tháng 04 năm 2023*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

A picture containing linedrawing

Description automatically generated

*Huỳnh Hoàng Tiến Đạt*

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

Tiểu luận này gồm có **02 chương**:

**Chương 1: Mô tả vấn đề nghiên cứu và các bước giải quyết vấn đề:**

Chương này trình bày về vấn đề được đặt ra trong bài tiểu luận lần này là phải giải quyết các bài tập về ma trận trong task 1 và được chia ra thành nhiều ý nhỏ trong task 1 với các câu từ a đến h. Đồng thời vấn đề song song cũng là phải đảm bảo tính hiệu quả khi giải quyết và tính đúng đắn của kết quả được đưa ra.

Để giải quyết vấn đề được đặt ra, một vài hướng tiếp cận được đặt ra, bao gồm sử dụng những thư viện được tích hợp sẵn trong Python như Numpy, Random, Math. Nó bao từ Task 1d đến 1h và mô tả các bước giải quyết. Các bài tập đó bao gồm trích xuất các số nguyên lẻ từ ma trận, tìm và lấy các số nguyên tố từ ma trận và lưu vào vector mới, đảo ngược các phần tử lẻ, tìm hàng có số lượng số nguyên tố lớn nhất và tìm hàng có nhiều số lẻ liền kề nhau nhất trong ma trận. Các bước giải được trình bài chi tiết kèm theo các đoạn mã với đầy đủ các hàm, thư viện Numpy, Math,... minh họa cho bài tập.

**Chương 2: Thực nghiệm:**

Trong chương này, em đã tiến hành thực hiện các bài toán được mô tả các bước giải chi tiết ở chương 1 bằng ngôn ngữ lập trỉnh Python kèm theo đó là sử dụng các thư viện như Numpy, Random, Math,... để có thể cho ra được kết quả. Ở chương này, chủ yếu là hình ảnh những đoạn mã và kết quả được xuất ra màn hình.

Tóm lại, trong quá trình nghiên cứu vấn đề được đặt ra để giải quyết được thì phải vận dụng kết hợp nhiều kiến thức đã được học tại lớp, những kiến thức tự học, tự tìm hiểu để có thể hiểu rõ về thư viện numpy, random, math trong Python và các hàm tích hợp sẵn để có thể sử dụng một cách thành thạo.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc133523158)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc133523159)

[TÓM TẮT iv](#_Toc133523160)

[MỤC LỤC 1](#_Toc133523161)

[DANH MỤC THƯ VIỆN VÀ CÁC HÀM, KÍ HIỆU ĐƯỢC SỬ DỤNG 3](#_Toc133523162)

[Chapter 1: Methodology of Solving Tasks 4](#_Toc133523163)

[1.1 Giới thiệu thư viện Nympy 4](#_Toc133523164)

[1.2 Các hàm được sử dụng trong các Task 1d,e,f,g,h 4](#_Toc133523165)

[1.3 Tạo ma trận A, B, C 4](#_Toc133523166)

[1.4 Task 1d: Lưu số nguyên lẻ trong ma trận A vào một vector mới 4](#_Toc133523167)

[1.5 Task 1e: Lưu số nguyên tố trong ma trận A vào một vector mới 5](#_Toc133523168)

[1.6 Task 1f: Đảo ngược các phần tử thuộc hàng lẻ của ma trận D 5](#_Toc133523169)

[1.7 Task 1g: Tìm các hàng có số lượng số nguyên tố nhiều nhất trong ma trận A 5](#_Toc133523170)

[1.8 Task 1h: Tìm các hàng có dãy số lẻ liền kề nhau dài nhất trong ma trận A 5](#_Toc133523171)

[Chapter 2: Source codes and Outputs 7](#_Toc133523172)

[2.1 - Source code từ task 1a đến 1h và giải thích 7](#_Toc133523173)

[*2.1.1 Source code task 1a 7*](#_Toc133523174)

[*2.1.2 Source code task 1b 7*](#_Toc133523175)

[*2.1.3 Source code task1c 7*](#_Toc133523176)

[*2.1.4 Source code task1d 8*](#_Toc133523177)

[*2.1.5 Source code task1e 8*](#_Toc133523178)

[*2.1.6 Source code task1f 8*](#_Toc133523179)

[*2.1.7 Source code task1g 8*](#_Toc133523180)

[*2.1.8 Source code task1h 9*](#_Toc133523181)

[2.2 - Outputs của đoạn chương trình trên 9](#_Toc133523182)

[2.2.1 In ma trận A 9](#_Toc133523183)

[2.2.2 Outputs task1a 9](#_Toc133523184)

[2.2.3 Outputs task1b 10](#_Toc133523185)

[2.2.4 Outputs task1c 10](#_Toc133523186)

[2.2.5 Outputs task1d 10](#_Toc133523187)

[2.2.6 Outputs task1e 11](#_Toc133523188)

[2.2.7 Outputs task1f 11](#_Toc133523189)

[2.2.8 Outputs task1g 11](#_Toc133523190)

[2.2.9 Outputs task1h 11](#_Toc133523191)

[CHƯƠNG 3 – CƠ SỞ LÝ THUYẾT 12](#_Toc133523192)

DANH MỤC THƯ VIỆN VÀ CÁC HÀM, KÍ HIỆU ĐƯỢC SỬ DỤNG

**THƯ VIỆN**

numpy Là một thư viện toán học phổ biến và mạnh mẽ trong python. Cho phép làm việc hiệu quả với ma trận và mảng.

random Là một thư viện tạo số ngẫu nhiên bằng cách sử dụng module random trong python.

math Là một thư viện dành cho các hàm toán học trong python.

**CÁC HÀM, KÝ HIỆU ĐƯỢC SỬ DỤNG**

np.random.randint Là một hàm trong python, sử dụng để tạo ra các số nguyên ngẫu nhiên từ bao gồm (thấp) đến cao.

np.transpose Là một hàm trong python, sử dụng để tìm chuyển vị của ma trận.

np.add Là một hàm trong python, sử dụng để thực hiện phép toán cộng trên hai mảng.

np.zeros Là một hàm trong python, sử dụng để tạo mảng có giá trị là 0 với kích thước được chỉ định.

np.linalg.matrix\_power Là một hàm trong python, sử dụng để tính tổng các phần tử trong một mảng hoặc trục của mảng đó.

np.max Là một hàm trong python, sử dụng để tìm giá trị lớn nhất trong một mảng hoặc trục của nó.

np.where Là một hàm trong python, sử dụng để tìm vị trí của các phần tử trong một mảng nếu thỏa mãn điều kiện cho trước.

@ (np.dot) Là phép nhân được sử dụng để nhân hai ma trận trong python.

**LỜI MỞ ĐẦU**

Hiện nay, trong thời kỳ các nước đang trong quá trình phát triển mạnh mẽ, chuyển dần sang công nghệ và kỹ thuật số, nhu cầu và đời sống của con người ngày càng được cải thiện, con người dần có xu hướng trải nghiệm, tìm kiếm những điều mới mẻ, tận hưởng cuộc sống hiện đại, ứng dụng những công nghệ hiện đại như robot,... thì việc ứng dụng, sử dụng những ngôn ngữ lập trình để tạo ra những trang web, những trò chơi, ứng dụng để phục vụ cho nhu cầu của con người là điều không thể tránh khỏi. Trong số những ngôn ngữ lập trình đó, có lẽ Python là một ngôn ngữ lập trình phổ biến và được sử dụng rộng rãi trong nhiều lĩnh vực. Nó được đánh giá cao về tính linh hoạt và dần trở thành công cụ quan trọng trong việc giải quyết các bài toán liên quan đến khoa học cụ thể như khoa học dữ liệu, trí tuệ nhân tạo AI, xử lý ảnh, kỹ thuật số,... và còn nhiều lĩnh vực khác nữa. Đặc biệt, Python còn là một công cụ hữu ích, mang lại nhiều lợi ích cho người dùng khi nó được sử dụng để tạo ra các web, desktop, mobile và một vài ứng dụng phần mềm khác.

CHƯƠNG 1: LÝ THUYẾT

* 1. Giới thiệu thư viện Python:

- *Python* là một ngôn ngữ linh hoạt. Các thư viện *Python* khác nhau sẽ bao gồm các module dành riêng cho các lĩnh vực cụ thể. Thư viện của ngôn ngữ lập trình *Python* được sử dụng cho mọi thứ từ trực quan hóa dữ liệu, khoa học dữ liệu, xử lý ngôn ngữ và phân tích những dữ liệu phức tạp. Trong *Python* gồm có các thư viện như *Numpy, Matplotlib, TensorFlow, Theano, PyTorch, Scikit-learn,...*

1.1.1 Giới thiệu về thứ viện Numpy

* *Numpy* hay *Numeric Python* là một thư viện cơ bản phục vụ cho tính toán khoa học trong *Python*. Đây là một thư viện trong Python có chức năng cung cấp các đối tượng mảng đa chiều, các đối tượng dẫn xuất khác nhau (ví dụ như mảng hoặc ma trận) và một loạt các hàm cho phép tính toán nhanh trên mảng, bao gồm toán học, logic, sắp xếp, lựa chọn, đại số tuyến tính cơ bản, các phép biến đổi Fourier, ...
* Thư viện Nympy của Python được viết một phần bằng ngôn ngữ Python và hầu hết phần còn lại được viết bằng ngôn ngữ C hoặc C++ chính vì thế nó cũng có các công cụ tích hợp mã C / C++ và Fortan.

Nguồn tham khảo: *<https://numpy.org/doc/stable/user/whatisnumpy.html>*

1.1.2 Giới thiệu về thư viện Random

* Thư viện *Random* trong ngôn ngữ lập trình *Python* là một module cung cấp các trình tạo số ngẫu nhiên cho các bản phân phối khác nhau. Bên trong thư viện *Random* có các hàm sinh số ngẫu nhiên, lựa chọn một cách ngẫu nhiên, hoán vị và rải rác các phần tử.

Nguồn tham khảo: [*https://docs.python.org/3.11/library/random.html*](https://docs.python.org/3.11/library/random.html)

1.1.3 Giới thiệu về thư viện Math

* Thư viện *Math* trong ngôn ngữ lập trình *Python* là một module được định nghĩa là hàm toán học phổ biến nhất, cung cấp quyền truy cập vào các hàm toán học như là hàm lượng giác, hàm số, hàm logarit,... Dùng để xử lí các vấn đề liên quan đến toán học. Ngoài ra nó còn định nghĩa cả hai hằng số toán học là số *Pi (π)* và *Euler (e)*.

Nguồn tham khảo: *<https://docs.python.org/3.11/library/math.html>*

* 1. **Những kiến thức liên quan đến vấn đề nghiên cứu**
     1. **Giới thiệu về ma trận**
* Ma trận theo định nghĩa trong toán học là một mảng hình chữ nhật, hoặc là hình vuông (thường được gọi là ma trận vuông vì số dòng bằng số cột) – mà mỗi ma trận tuân theo những quy tắc định trước. Từng giá trị trong ma trận thì được gọi là ác phần tử hoặc mục.

Ví dụ một ma trận có 2 hàng và 3 cột là:

A picture containing text, clock

Description automatically generated

Ví dụ một ma trận vuông 3 hàng và 3 cột là:

Calendar

Description automatically generated **Nguồn tham khảo:**[***đường dẫn tham khảo***](https://vi.wikipedia.org/wiki/Ma_tr%E1%BA%ADn_(to%C3%A1n_h%E1%BB%8Dc))

* Trong Python không có kiểu xây dựng ma trận, nên sẽ được biểu diễn một ma trận dưới dạng một nested list.
* Nestedlist là dạng danh sách lồng ghép, tức là list này xuất hiện với vai trò là phần tử trong list khác.
* Thường trong Python, để biểu diễn một ma trận người ta sẽ sử dụng thư viện Numpy với các hàm được tích hợp sẵn.

**Ví dụ:**

import numpy as np

A = np.array([[1, 9, -13],

[20, 5, -6]])

print(A)

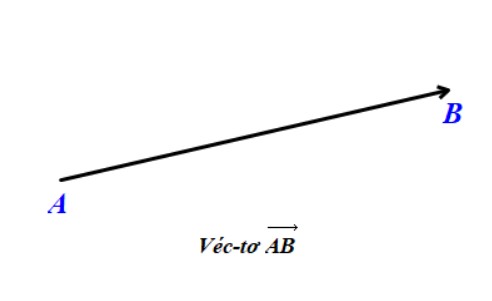
Thì ta sẽ thu được ma trận với 2 hàng và 3 cột là:

[[ 1 9 -13]

[ 20 5 -6]]

**Nguồn tham khảo: *[đường dẫn tham khảo](https://quantrimang.com/hoc/ma-tran-trong-python-160014)***

* + 1. **Giới thiệu về vector**
* Vecto trong toán học được định nghĩa là một đoạn thẳng có hướng chỉ rõ điểm nào là điểm đầu (gốc), điểm nào là điểm cuối (ngọn).



**Nguồn tham khảo: *[đường dẫn tham khảo](https://vi.wikipedia.org/wiki/Vect%C6%A1)***

* Vecto trong Python chỉ có một cột. (Mảng một chiều)

Ví dụ:

import numpy as np

A = np.array([10, 20, 30, 40])

print(A)

Kết quả thu được vector: [10 20 30 40]

* 1. **Các hàm được sử dụng trong các Task 1d, 1e, 1f, 1g, 1h**
* append(): Là một hàm trong python dùng để thêm vào cuối một list.

Ví dụ:

Graphical user interface, text

Description automatically generated

Kết quả:



* + - math.sqrt(): Là một hàm trong thư viện Math của ngôn ngữ lập trình Python, dùng để trả về căn bậc hai của một số.

Ví dụ:

Graphical user interface, text, application

Description automatically generated

Kết quả:



* + - np.max: Là một hàm trong python, sử dụng để tìm giá trị lớn nhất trong một mảng hoặc trục của nó.

Ví dụ:

Text

Description automatically generated

Kết quả:



* + - np.where: Là một hàm trong python, sử dụng để tìm vị trí của các phần tử trong một mảng nếu thỏa mãn điều kiện cho trước.

Ví dụ:

Text, chat or text message

Description automatically generated

Kết quả:

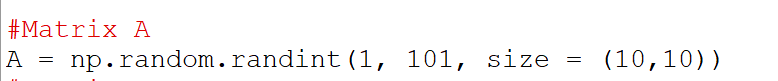


* 1. **Tạo ma trận A, B, C**
     + **Bước 1:** Khai báo thư viện numpy, random để tiến hành tạo dựng ma trận ngẫu nhiên

Text

Description automatically generated

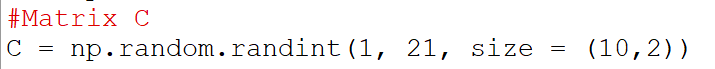
* + - **Bước 2:** Sử dụng hàm “np.random.randint” để tiến hành tạo ma trận A với kích thước 10x10 trong phạm vi từ 1 đến 100.



* + - **Bước 3:** Sử dụng hàm np.random.randint để tiến hành tạo ma trận B với kích thước 2x10 trong phạm vi từ 1 đến 20.



* + - **Bước 4:** Sử dụng hàm np.random.randint để tiến hành tạo ma trận C với kích thước 10x2 trong phạm vi từ 1 đến 20.



* 1. *Task 1d: Lưu số nguyên lẻ trong ma trận A vào một vector mới*
* Bước 1: Khai báo thư viện bằng cách import thư viện numpy, random

Text

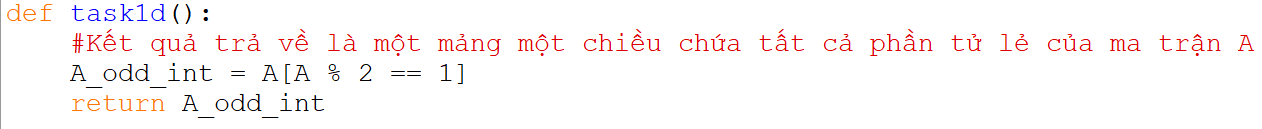
Description automatically generated

* Bước 2: Định nghĩa câu 1d bằng “def task1d”.



* Bước 3:
* Dùng biến A\_odd\_int để trích xuất các số lẻ.
* Biểu thức A % 2 == 1 để lọc ra tất cả những phần tử của ma trận A có giá trị là số lẻ. Biểu thức này sẽ trả về một ma trận có cùng kích thước với ma trận A, tuy nhiên các phần tử trong đó sẽ chỉ mang giá trị True hoặc False, tức là vị trí tương ứng phần tử đó trong ma trận A có phải là phần tử lẻ hay không.
* Hàm sủ dụng mảng boolean trả về từ biểu thức trên để chọn ra các phẩn tử lẻ tương ứng trong ma trận A bằng cách sử dụng [] để index ma trận A.
* A[A%2==1] tạo ra mảng mới chỉ chứa các phần tử A tại các vị trí tương ứng giá trị True trong boolean A%2 == 1.
  + Vậy A\_odd\_int sẽ chứa toàn bộ các số lẻ.
* Bước 4: Tiến hành xuất biến A\_odd\_int ra theo định dạng để được kết quả của task 1d.



Code thực hiện task1d ở trên: 



* 1. *Task 1e: Lưu số nguyên tố trong ma trận A vào một vector mới*
* Bước 1: Khai báo thư viện, import thư viện numpy, random, math

Text

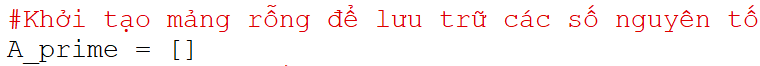
Description automatically generated

* Bước 2: Định nghĩa câu 1e bằng câu lệnh “def task1e”

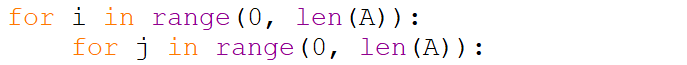
A picture containing logo

Description automatically generated

* Bước 3: Đầu tiên là khởi tạo mảng rỗng để lưu trữ các số nguyên tố “A\_prime” để lưu trữ các số nguyên tố mà tìm thấy trong ma trận A.



* Bước 4:
* Sau đó dùng hai vòng lặp for để tiến hành duyệt qua các phần tử trong ma trận A.
* Vòng lặp đầu tiên duyệt qua từng hàng của ma trận, vòng lặp thứ hai duyệt qua từng cột của mỗi hàng. Lúc này biến i và j được sử dụng để chỉ ra vị trí của phần tử trong ma trận A.



* Bước 5: Trong mỗi vòng lặp for, hàm sử dụng hàm con là “is\_prime” để lần lượt kiểm tra tại vị trí i, j trong ma trận A có phải là số nguyên tố hay không. Hàm sẽ trả về giá trị là 1 nếu tại vị trí i,j là số nguyên tố và sẽ là -1 nếu tại i,j không phải là số nguyên tố.



Code hàm “is\_prime”:

Graphical user interface, text, application, Word

Description automatically generated

* Bước 6: Nếu phần tử tại vị trí i,j trong ma trận A là số nguyên tố, thì hàm sẽ thêm phần tử đó vào mảng A\_prime bằng phương thức append.



* Bước 7: Sau khi duyệt quả tất cả các phần tử trong ma trận A thì hàm sẽ trả về mảng A\_prime dưới dạng mảng bằng cách sử dụng phương thức array của thư viện Numpy tích hợp sẵn trong python.



* Bước 8: Tiến hành xuất A\_prime theo định dạng để được kết quả của task1e.



Code thực hiện task 1e:

A picture containing text

Description automatically generated



* 1. *Task 1f: Đảo ngược các phần tử thuộc hàng lẻ của ma trận D*
* Bước 1: Khai báo thư viện, import thư viện numpy, math, random

Text

Description automatically generated

* Bước 2: Định nghĩa câu 1f bằng câu lệnh “def task1f”



* Bước 3: Bắt đầu bằng cách tính toán tích của hai ma trận B và C bằng cách sử dụng @ hoặc np.dot trong Python. Kết quả của phép nhân ma trận này sẽ được gán vào biến có tên là D.

A picture containing text

Description automatically generated

* Bước 4:
* Hàm duyệt qua các hàng lẻ của ma trận D bằng cách sử dụng vòng lặp for với bước nhảy là 2. Tức là bắt đầu là 0 và tăng lên 2 sau mỗi lần lặp lại. Lúc này, biến i được sử dụng để chỉ ra chỉ số hàng của ma trận D.
* Trong mỗi lần lặp lại, hàm thực hiện phép biến đổi trên hàng i của ma trận D. Để làm được nó, hàm sử dụng toán tử lấy phần của mảng (slicing) của thư viện *Numpy* được tích hợp sẵn trong *Python* để lấy tất cả các phần tử của hàng i theo thứ tự đảo ngược lại. Giá trị của hàng đó được gán bằng kết quả của phép ngược đó.

Text

Description automatically generated

* Bước 5: Sau khi đã thực hiện phép biến đổi trên tất cả các hàng lẻ của ma trận D, hàm trả về ma trận D đã được biến đổi.



* Bước 6: Cuối cùng hiển thị nội dung, xuất ma trận D ra màn hình theo định dạng và thu được kết quả của task1f.



Code thực hiện task1f:

Scatter chart

Description automatically generated



* 1. *Task 1g: Tìm các hàng có số lượng số nguyên tố nhiều nhất trong ma trận A*
* Bước 1: Khai báo thư viện, import thư viện numpy, random, math

Text

Description automatically generated

* Bước 2: Định nghĩa câu 1g bằng câu lệnh “def task1g”



* Bước 3: Khai báo các biến đếm số lượng số nguyên tố trong mỗi hàng của ma trận A.

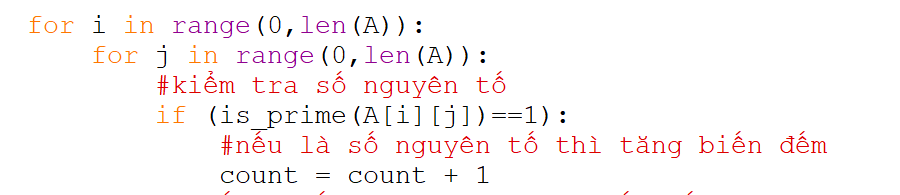
Logo

Description automatically generated with low confidence

* Bước 4: Tiếp tục khởi tạo mảng rộng temp để lưu trữ số lượng số nguyên tố trong từng hàng của ma trận A.



* Bước 5: Tiếp tục sử dùng hai vòng lặp để duyệt qua từng phần tử của ma trận A. Với mỗi phần tử, sử dụng hàm “is\_prime” để kiểm tra xem có phải là số nguyên tố hay không. Nếu đúng thì tăng biến đếm lên 1.



Code hàm kiểm tra số nguyên tố “is\_prime” là:

Graphical user interface, text, application, Word

Description automatically generated

* Bước 6: Sau khi duyệt qua tất cả các phần tử của hàng, thêm giá trị của biến đếm và mảng temp, tiếp tục reset biến đếm về 0 để tính lại số lượng số nguyên tố cho hàng tiếp theo.

Diagram

Description automatically generated with low confidence

* Bước 7: Tiếp tục dùng vòng lặp for để duyệt qua mảng temp để tìm hàng có số lượng số nguyên tố lớn nhất trong mảng temp.

A picture containing diagram

Description automatically generated

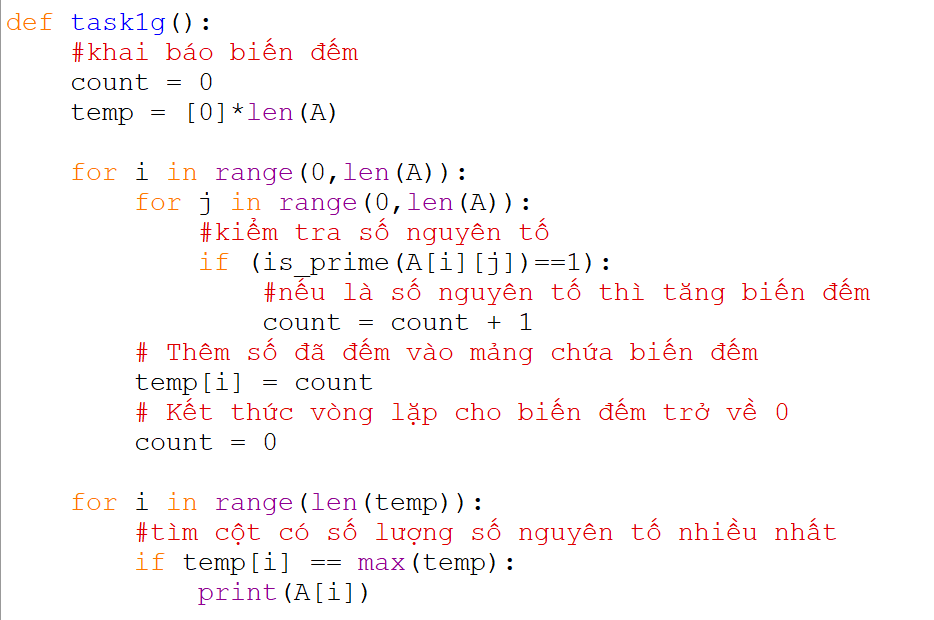
* Bước 8: Sau đó ta in ra hàng của ma trận A tại vị trí bằng cách sử dụng A[i].



* Bước 9: Xuất ra kết quả ra ngoài màn hình theo định dạng và ta thu được kết quả task 1g.



Code thực hiện task1g:





* 1. *Task 1h: Tìm các hàng có dãy số lẻ liền kề nhau dài nhất trong ma trận A*
* Bước 1: Khai báo thư viện, import các thư viện numpy, random, math

Text

Description automatically generated

* Bước 2: Định nghĩa câu 1h bằng câu lệnh “def task1h”



* Bước 3: Tạo một mảng rỗng (biến odd\_length) để lưu giá trị số lượng phần tử liên tiếp lẻ lớn nhất của mỗi hàng của ma trận A.

A picture containing company name

Description automatically generated

* Bước 4: Với mỗi hàng của ma trận A, đoạn code sẽ duyệt qua từng phần tử của hàng đó và đếm số lượng phần tử liên tiếp lẻ của hàng đó. Nếu phần tử hiện tại là số lẻ, biến đếm sẽ là được tăng thêm một đơn vị. Nếu phần tử hiện tại không phải số lẻ, biến đếm sẽ được thiết lập lại thành 0. Trong quá trình duyệt, đoạn mã sẽ cập nhật giá trị lớn nhất của biến đếm (max\_ len) cho mỗi hàng. Cuối cùng, giá trị lớn nhất này sẽ được lưu vào mảng odd\_length.

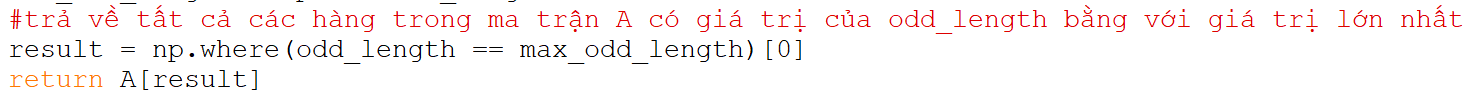
Text

Description automatically generated

* Bước 5: Tìm giá trị lớn nhất trong mảng odd\_length bằng hàm np.max của thư viện numpy được tích hợp sẵn trong python và lưu kết quả vào biến max\_odd\_length.



* Bước 6:
* Tìm tất cả các hàng của ma trận A có giá trị số lượng phần tử liên tiếp lẻ bằng với giá trị lớn nhất (max\_odd\_length) bằng hàm np.where của thư viện numpy được tích hợp sẵn trong python và lưu kết quả vào biến result.
* Hàm np.where sẽ trả về một tuple chứa các mảng tương ứng với vị trí của các phần tử trong mảng vào (odd\_length) thỏa mãn điều kiện. Trong trường hợp này, chúng ta chỉ quan tâm đến mảng đầu tiên trong tuple (np.where(odd\_length == max\_odd\_length)[0]), vì nó chứa các chỉ số hàng của ma trận A mà ta đang tìm kiếm.



* Bước 7: Cuối cùng, hàm trả về các hàng của ma trận A có chỉ số ở result. Tiến hành xuất ra màn hình theo định dạng và thu được kết quả của task1h.



Code thực hiện task1h:

Text

Description automatically generated

**CHƯƠNG 2: THỰC NGHIỆM**

2.1 Câu a

[Tính , in kết quả ra màn hình.](#_Toc387692918)

2.1.1 Mã nguồn task 1a

import numpy as np

import random as rd

#Matrix A

A = np.random.randint(1, 101, size = (10,10))

#Matrix B

B = np.random.randint(1, 21, size = (2,10))

#Matrix C

C = np.random.randint(1, 21, size = (10,2))

def task1a():

#Transpose: (ma trận chuyển vị của ma trận A B C)

a = np.transpose(A)

b = np.transpose(B)

c = np.transpose(C)

#tạo ma trận temp để tính tổng ma trận A ban đầu và ma trận chuyển vị của A

temp = np.add(A, a)

#tạo ma trận temp1 để tính tổng của tích ma trận C và ma trận B

#với tích ma trận chuyển vị B và ma trận chuyển vị C

temp1 = np.add(C @ B, b @ c)

result = np.add(temp,temp1)

return result

print("Matrix A:")

print(A)

print("Matrix B")

print(B)

print("Matrix C")

print(C)

print("---------------------------------")

print("a) Result: \n", task1a())

**2.1.2: Kết quả:**

Matrix A:

[[42 26 63 39 12 81 71 33 90 55]

[39 19 80 86 21 79 4 9 92 19]

[78 19 20 84 92 65 53 68 39 45]

[89 89 27 19 6 66 56 38 59 40]

[82 97 57 60 99 76 78 97 65 4]

[92 7 44 81 65 89 90 37 42 44]

[29 86 88 69 80 44 69 55 71 93]

[66 79 96 55 91 48 36 9 87 5]

[31 8 49 78 73 49 86 40 2 42]

[60 74 13 45 11 72 98 58 69 7]]

Matrix B

[[ 3 14 15 12 9 6 17 19 7 17]

[14 10 2 6 7 16 20 20 18 11]]

Matrix C

[[ 9 9]

[ 6 16]

[ 3 8]

[13 11]

[ 2 1]

[18 1]

[ 7 15]

[14 6]

[17 13]

[ 7 13]]

---------------------------------

a) Result:

[[ 390 523 415 483 258 439 664 576 579 570]

[ 523 526 343 635 322 640 760 778 798 599]

[ 415 343 162 412 264 527 487 603 534 328]

[ 483 635 412 482 290 623 740 764 708 589]

[ 258 322 264 290 248 338 380 414 414 214]

[ 439 640 527 623 338 426 742 627 545 683]

[ 664 760 487 740 380 742 976 882 1025 854]

[ 576 778 603 764 414 627 882 790 916 760]

[ 579 798 534 708 414 545 1025 916 710 826]

[ 570 599 328 589 214 683 854 760 826 538]]

2.2 Câu b

[Tính , và in kết quả ra màn hình](#_Toc387692919).

2.2.1 Mã nguồn task 1b

import numpy as np

import random as rd

#Matrix A

A = np.random.randint(1, 101, size = (10,10))

def task1b():

#Khởi tạo ma trận 10x10 với toàn số 0

result = np.zeros((10,10))

for i in range(1,10):

#dùng hàm tính toán lũy thừa của ma trận

result += np.linalg.matrix\_power((A/(i+10)),i+1)

return result

print("Matrix A: ")

print(A)

print("b) Result: \n", task1b())

**2.2.2: Kết quả**

Matrix A:

[[ 81 78 56 67 81 70 61 53 97 45]

[ 8 70 42 33 72 1 75 96 56 80]

[ 77 98 58 100 41 68 35 94 75 80]

[ 43 37 54 80 46 10 84 32 2 89]

[ 33 64 68 40 48 9 32 55 28 14]

[ 35 81 25 52 97 68 93 26 11 72]

[ 71 53 55 76 45 20 67 49 64 60]

[ 36 24 89 84 36 96 60 43 6 41]

[ 21 58 75 63 2 25 34 82 48 100]

[ 26 40 98 50 60 76 73 49 53 9]]

b) Result:

[[4.20088929e+13 5.73590415e+13 6.06114289e+13 6.35191945e+13

5.04511178e+13 4.20567341e+13 5.98647041e+13 5.63898553e+13

4.20166150e+13 5.71217473e+13]

[3.16161302e+13 4.31687389e+13 4.56165018e+13 4.78049028e+13

3.79697982e+13 3.16521337e+13 4.50545154e+13 4.24393216e+13

3.16219412e+13 4.29901481e+13]

[4.45222999e+13 6.07908536e+13 6.42378318e+13 6.73195689e+13

5.34696263e+13 4.45730005e+13 6.34464299e+13 5.97636807e+13

4.45304851e+13 6.05393604e+13]

[2.94786753e+13 4.02502530e+13 4.25325339e+13 4.45729839e+13

3.54027920e+13 2.95122459e+13 4.20085374e+13 3.95701520e+13

2.94840949e+13 4.00837376e+13]

[2.43072250e+13 3.31891424e+13 3.50710408e+13 3.67535350e+13

2.91920739e+13 2.43349062e+13 3.46389716e+13 3.26283498e+13

2.43116927e+13 3.30518382e+13]

[3.29982329e+13 4.50558650e+13 4.76106364e+13 4.98947031e+13

3.96296489e+13 3.30358140e+13 4.70240772e+13 4.42945654e+13

3.30042993e+13 4.48694704e+13]

[3.47386456e+13 4.74322288e+13 5.01217430e+13 5.25262774e+13

4.17198216e+13 3.47782064e+13 4.95042501e+13 4.66307739e+13

3.47450313e+13 4.72360012e+13]

[3.24269926e+13 4.42758926e+13 4.67864372e+13 4.90309620e+13

3.89436100e+13 3.24639187e+13 4.62100305e+13 4.35277730e+13

3.24329562e+13 4.40927227e+13]

[3.18781608e+13 4.35265164e+13 4.59945662e+13 4.82011034e+13

3.82844866e+13 3.19144607e+13 4.54279206e+13 4.27910547e+13

3.18840214e+13 4.33464446e+13]

[3.33129804e+13 4.54856222e+13 4.80647598e+13 5.03706122e+13

4.00076501e+13 3.33509168e+13 4.74726076e+13 4.47170597e+13

3.33191048e+13 4.52974477e+13]]

2.3 Câu c

Lưu các hàng lẻ của ma trận A vào một ma trận mới và in kết quả ra màn hình.

2.3.1 Mã nguồn task 1c

import numpy as np

import random as rd

#Matrix A

A = np.random.randint(1, 101, size = (10,10))

def task1c():

#Trích xuất các hàng ở vị trí lẻ trong ma trận A

A\_odd = A[::2]

return A\_odd

print("Matrix A: ")

print(A)

print("c) Odd rows of the matrix A into a new matrix is: \n", task1c())

**2.3.2: Kết quả**

Matrix A:

[[ 77 99 95 53 54 73 24 32 55 85]

[ 90 33 10 1 77 11 95 65 76 24]

[ 44 35 32 10 25 56 4 48 48 8]

[ 34 94 31 57 35 63 45 88 53 26]

[ 74 40 51 24 64 18 96 99 50 62]

[ 64 89 52 15 12 76 84 44 100 82]

[ 3 29 7 18 7 62 79 54 82 34]

[ 25 27 90 10 89 83 12 32 22 32]

[ 34 100 87 78 39 97 39 15 61 53]

[ 42 65 68 33 15 25 73 4 15 97]]

c) Odd rows of the matrix A into a new matrix is:

[[ 77 99 95 53 54 73 24 32 55 85]

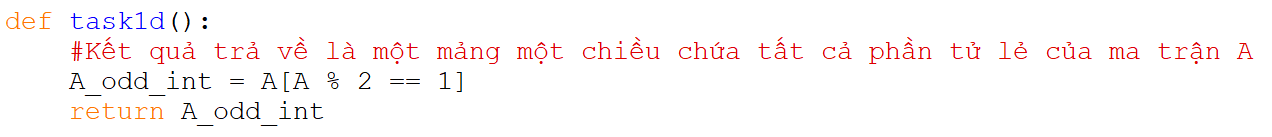
[ 44 35 32 10 25 56 4 48 48 8]

[ 74 40 51 24 64 18 96 99 50 62]

[ 3 29 7 18 7 62 79 54 82 34]

[ 34 100 87 78 39 97 39 15 61 53]]

2.1.4 Source code task1d



2.1.5 Source code task1e

Text

Description automatically generated with low confidence

2.1.6 Source code task1f

Scatter chart

Description automatically generated

2.1.7 Source code task1g

Text

Description automatically generated

2.1.8 Source code task1h

Text

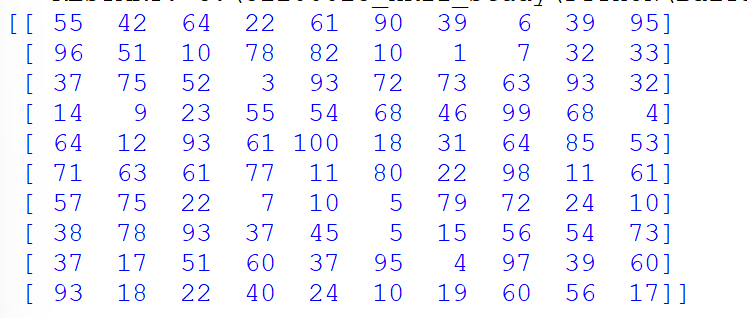
Description automatically generated with medium confidence

Scatter chart

Description automatically generated

2.2 - Outputs của đoạn chương trình trên

2.2.1 In ma trận A



2.2.2 Outputs task1a

Text

Description automatically generated with medium confidence

**2.2.3 Outputs task1b**

Text

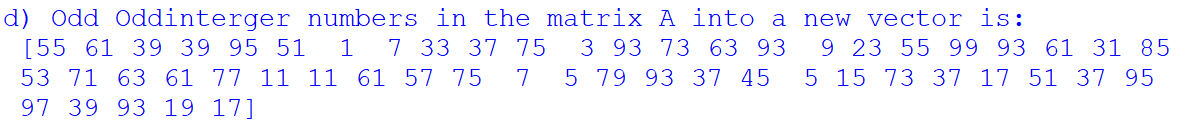
Description automatically generated

2.2.4 Outputs task1c

A screenshot of a computer

Description automatically generated with low confidence

2.2.5 Outputs task1d



**2.2.6 Outputs task1e**

Text

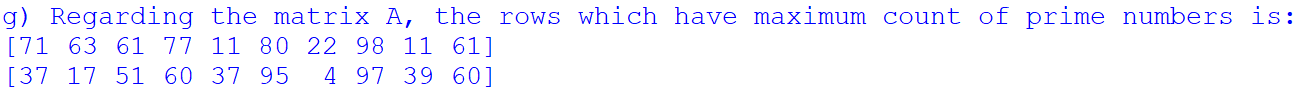
Description automatically generated

2.2.7 Outputs task1f

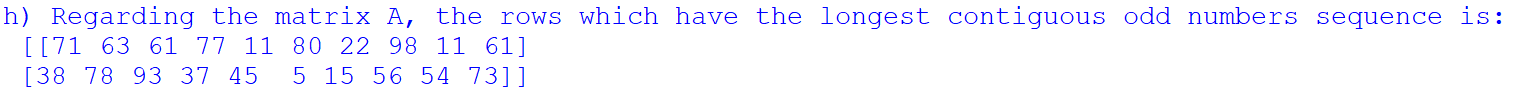
Text

Description automatically generated

2.2.8 Outputs task1g



2.2.9 Outputs task1h



**CHƯƠNG 3 – CƠ SỞ LÝ THUYẾT**

1. *Tiếng Việt:*
2. *Tiếng Anh:*