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 CNTT

Người hướng dẫn: **Thầy Nguyễn Văn Khoa**

Người thực hiện: **Đặng Thành Nhân – 522H0006**

Lớp **: 22H50201**

Võ Nhật Hào – 522H0090

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 CNTT

Người hướng dẫn: **Thầy Nguyễn Văn Khoa**
Người thực hiện: **Đặng Thành Nhân – 522H0006**
Lớp **: 22H50201**

Võ Nhật Hào – 522H0090
Lớp **: 22H50202**
Khoá **: 26**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Đây là phần tác giả **tự viết** ngắn gọn, thể hiện sự biết ơn của mình đối với những người đã giúp mình hoàn thành Luận văn/Luận án. Tuyệt đối không sao chép theo mẫu những "lời cảm ơn" đã có.

- Lời đầu tiên chúng em cảm ơn ơn thầy Nguyễn Văn Khoa đã đồng hành cùng chúng em. Cảm ơn thầy đã dạy và hướng dẫn cho chúng em hoàn thành bài tiểu luận giữa kì 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 tiểu luận này. Chúng em tin chắc rằng bài tiểu luận này sẽ cho chúng em tiếp cận với những kĩ năng làm việc nhóm để khi đi làm có thể thích nghi với môi trường mới hơn.

- Chúng em chưa được làm nhiều bài tập về tiểu luận nên có thể còn nhiều lỗi và sai sót. Hy vọng thầy/cô chấm điểm và góp ý để chúng em càng phát triển, tốt hơn trên lĩnh vực này.

- Lời cuối cùng, chúng em xin cảm ơn thầy/cô đã đọc và chấm điểm. Chúc cho thầy/cô thật nhiều sức khỏe để có thể truyền tải những kiến thức mới cho chúng em.

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 Thầy 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 22 tháng 4 năm 2023

Tác giả

(ký tên và ghi rõ họ tên)

Nhân

Đặng Thành Nhân

Hào

Võ Nhật Hào

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

Trình bày tóm tắt vấn đề nghiên cứu, các hướng tiếp cận, cách giải quyết vấn đề và một số kết quả đạt được, những phát hiện cơ bản trong vòng 1 -2 trang.

- Bài tiểu luận dựa trên những phương pháp, cách tính trong ma trận và được thực hiện trên ngôn ngữ Python.
- Được thực hiện bởi 2 sinh viên của Đại học Tôn Đức Thắng:
    - Võ Nhật Hào
    - Đặng Thành Nhân
- Bài tiểu luận được tham khảo ở những bài lab đã học.

MỤC LỤC

- Write a detailed description of the solving methods used in each Task 1d, 1e, 1f, 1g, and 1h in the "Programming part". For ex.:
  - Initialize an empty list/numpy array
  - Conduct a for loop to ...
  - Write a function to check a prime number
  - ....

- Create images of all source codes and corresponding outputs in the "Programming part", and insert them into this chapter. The images must be clear, and properly laid out. The images captions and descriptions are also required.
- Hint: To get a clear and high quality image, you could make an image for each sub - task's source code/results.

CHAPTER 1 – METHODOLOGY OF SOLVING TASKS

Task 1:

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

- Tạo một mảng rỗng odd_vector
- Dùng lệnh A.flatten() để chuyển ma trận A thành mảng 1 chiều.
- Tiến hành vòng lặp for để duyệt qua từng phần tử trong mảng A.
    - Kiểm tra phần tử là số chẵn hay lẻ.
        - Nếu là số lẻ lưu phần tử vào mảng odd_vector.
- In ra mảng có phần tử lẻ.

```
print("d)")
#d
odd_vector = []
for i in A.flatten():
  if(i % 2 != 0):
    odd_vector.append(i)
print("odd_vector =", odd_vector)
print()
```

e) Save prime numbers in the matrix A into a new vector, and print the resultant vector to the screen.

- Khai báo hàm boolean is_prime với tham số n.
    - Nếu tham số n bé hơn 1 thì kết luận n không là số nguyên tố

- Tiến hành vòng lặp for để duyệt qua mảng từ 2 đến căn bậc 2 của n+1.
  - Kiểm tra phần tử n có chia hết cho số nào trong khoảng từ 2 đến căn bậc 2 của n+1 không
  - Nếu n không chia hết cho số nào trong khoảng từ 2 đến căn bậc 2 của n+1 thì nó là số nguyên tố và ngược lại.

➢ Tạo một mảng rỗng prime_vectorA.

➢ Dùng lệnh A.flatten() để chuyển ma trận A thành mảng 1 chiều.

➢ Tiến hành vòng lặp for để duyệt x phần tử trong mảng A.

- Sử dụng hàm is_prime để kiểm tra xem x có phải là số nguyên tố hay không.
  - Nếu x là số nguyên tố thì lưu vào mảng prime_vectorA.

➢ In ra mảng prime_vectorA.

```
print("e)")
# e
def is_prime(n):
  if n<=1:
    return False
  for i in range(2, int(np.sqrt(n))+1):
    if n % i == 0:
      return False
  return True

prime_vectorA = []
for x in A.flatten():
  if(is_prime(x)):
    prime_vectorA.append(x)
print("prime_vectorA =", prime_vectorA)
print()
```

f) Given a matrix D = CB, reverse elements in the odd rows of the matrix D, and print the resultant matrix to the screen.

- Tạo ma trận D bằng cách lấy ma trân C nhân với ma trận B.
- In ra ma trận D ban đầu.
- Tiến hành vòng lặp for để duyệt qua i hàng của ma trận D.
- Dùng lệnh D.shape[0] để truy xuất số hàng trong ma trận D.
    - Kiểm tra hàng i là hàng chẵn hay lẻ.
        - Nếu i là hàng lẻ thì đảo ngược vị trí các phần tử trong mảng đó
        - Dùng lệnh [::-1] để đảo ngược vị trí các phần tử trong mảng.
- In ra ma trận D sau khi được đảo ngược các phần tử ở hàng lẻ.

```
print("f)")
#f
D = np.dot(C, B)

print("D = ", D, sep="\n")

for i in range(D.shape[0]):
  if i % 2 != 0 :
    D[i] = D[i][::-1]

print()
print("After reverse the odd rows D = ", D, sep="\n")
print()
```

g) Regarding the matrix A, find the rows which have maximum count of prime numbers, and print the rows to the screen.

- Tạo biến max_count để lưu số lượng số nguyên tố lớn nhất.
- Tạo mảng rỗng max_rows để lưu các phần tử của hàng có nhiều số nguyên tố nhất trong A.
- Tiến hành vòng lặp for để duyệt qua i hàng trong ma trận A.
- Dùng lệnh A.shape[0] để truy xuất số hàng trong ma trận A.
    - Tạo biến count để đếm số nguyên tố có trong hàng i.
    - Tiến hành vòng lặp for để duyệt qua j cột trong hàng i.
    - Dùng lệnh A.shape[1] để truy xuất số cột trong ma trận A.
        - Kiểm tra phần tử ở vị trí hàng i và cột j có là số nguyên tố không.
        - Nếu là số nguyên tố thì count cộng thêm một đơn vị.
    - Nếu giá trị của count lớn hơn giá trị max_count.
        - Cập nhật giá trị biến đếm max_count.
        - Lưu hàng các phần tử trong hàng i đó vào ma trận max_rows.
    - Nếu giá trị count bằng với giá trị max_count.
        - Lưu các phần tử trong hàng i đó vào ma trận max_rows.
        - Dùng lệnh A[i, :] để truy xuất các phần tử (cột) có trong hàng i.
- In ra những hàng có số nguyên tố nhiều nhất trong ma trận A.

```
print("g)")
# g
max_count = 0
max_rows = []

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

    count = 0
    for j in range(A.shape[1]):
        if is_prime(A[i,j]):
            count += 1
    if count > max_count:
        max_count = count
        max_rows = [A[i, :]]
    elif count == max_count:
        max_rows.append(A[i, :])

for row in max_rows:
    print(row)
print()
```

h) Regarding the matrix A, find the rows which have the longest contiguous odd numbers sequence, and print the rows to the screen.

- Tạo biến maxOdd_len để lưu độ dài số lẻ liên tiếp dài nhất.
- Tạo mảng rỗng maxOdd_rows để lưu các phần tử của hàng có dãy số lẻ liên tiếp dài nhất.
- Tiến hành vòng lặp for để duyệt i hàng trong ma trận A.
- Dùng lệnh A.shape[0] để truy xuất số hàng trong ma trận A.
    - Tạo biến currentOdd_len để lưu độ dài dãy số lẻ liên tiếp hiện tại trong hàng i.
    - Tạo biến maxOdd_in_row để lưu độ dài dãy số lẻ liên tiếp dài nhất trong hàng hiện tại trong hàng i.
    - Tiến hành vòng lặp for để duyệt j cột trong hàng i.
    - Dùng lệnh A.shape[1] để truy xuất số cột trong ma trận A.

        - Kiểm tra phần tử ở vị trí hàng i và cột j là số chẵn hay lẻ.
            - Nếu là số lẻ thì giá trị của currentOdd_len tăng thêm một đơn vị.
            - Nếu chuỗi số lẻ liên tiếp trong hàng i lớn hơn chuỗi số lẻ liên tiếp lớn nhất trong hàng i, cập nhật giá trị maxOdd_in_row.
        - Ngược lại, reset giá trị maxOdd_in_row.
    - Nếu độ dài chuỗi số lẻ lớn nhất trong hàng lớn hơn độ dài số lẻ lớn nhất tìm thấy cho đến nay.
        - Cập nhật maxOdd_len
        - Lưu các phần tử trong hàng i đó vào maxOdd_rows.
    - Nếu độ dài số lẻ lớn nhất trong hàng bằng với độ dài số lẻ lớn nhất tìm thấy cho đến nay.
        - Lưu các phần tử trong hàng i đó vào maxOdd_rows.
        - Dùng lệnh A[i, :] để truy xuất các phần tử(cột) có trong hàng i.
- In ra các hàng có dãy số lẻ liên tiếp dài nhất.

```
print("h)")
# h
maxOdd_len = 0
maxOdd_rows = []

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

    for j in range(A.shape[1]):
        if A[i,j] % 2 != 0:
            currentOdd_len += 1
            if currentOdd_len > maxOdd_in_row:
                maxOdd_in_row = currentOdd_len
        else:
            currentOdd_len = 0
    if maxOdd_in_row > maxOdd_len:
        maxOdd_len = maxOdd_in_row
        maxOdd_rows = [A[i, :]]
    elif maxOdd_in_row == maxOdd_len:
        maxOdd_rows.append(A[i,:])

for row in maxOdd_rows:
    print(row)
```

CHAPTER 2 –SOURCE CODES AND OUTPUTS

❖ Task 1

Code:

```
import numpy as np

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

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

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

print("A = ", A, sep="\n")
print("B = ", B, sep="\n")
print("C = ", C, sep="\n")
print()
```

Màn hình chạy:

```
A =
[[  6  13  26  52  65  26   5  98   6  79]
 [ 53  68  82  20  71  39  52  74  98  68]
 [ 83  52  67  36  48 100  47   5  32  27]
 [ 68  12  33  59  13  41  70  22  28  62]
 [ 45  42  15  67  49  94  20  17  63  47]
 [  3  90  84  48  66   7   2  96  23   7]
 [ 23  60  76  74   5  69  99  72  41  97]
 [ 55  43  54  73  23  70  13  67  62  23]
 [  9  37  91  58  53  80  22  34  54  66]
 [ 45  85  77  83  40  53  54  32  70  50]]
B =
[[ 2  2  1  8 13 14  2  8 14 12]
 [11 20  1 12 17  1 18  7  5 11]]
C =
[[11 14]
 [ 6  3]
 [ 8  5]
 [12 16]
 [10 16]
 [20  4]
 [19 18]
 [16 20]
 [20 10]
 [ 6 20]]
```

❖ Task 1a

Code:

```
print("a)")
#a
A_T = A.T
B_T = B.T
C_T = C.T

print(A + A_T + np.dot(C, B) + np.dot(B_T, C_T))
print()
```

Màn hình chạy:

```
a)
[[364 413 205 576 687 281 538 591 389 642]
 [413 280 259 460 582 336 576 618 474 670]
 [205 259 160 221 278 325 266 194 290 281]
 [576 460 221 694 780 481 824 671 614 753]
 [687 582 278 780 902 644 886 780 766 801]
 [281 336 325 481 644 582 467 598 693 448]
 [538 576 266 824 886 467 922 755 639 949]
 [591 618 194 671 780 598 755 670 650 655]
 [389 474 290 614 766 693 639 650 768 670]
 [642 670 281 753 801 448 949 655 670 684]]
```

❖ Task 1b

Code:

```
print("b)")
#b
D = np.zeros((10, 10))
for i in range(10, 19+1):
    j = i - 9
    D += (A/i)**j

print(D)
print()
```

Màn hình chạy:

```
b)
[[1.08779577e+00 6.50886125e+00 1.93823709e+02 5.18930299e+04
  3.97543528e+05 1.93823709e+02 8.08373189e-01 1.91714266e+07
  1.08779577e+00 2.46632021e+06]
 [6.15757328e+04 6.04449717e+05 3.50830858e+06 3.97955983e+01
  9.04280625e+05 4.26693226e+03 5.18930299e+04 1.33262411e+06
  1.91714266e+07 6.04449717e+05]
 [3.93520561e+06 5.18930299e+04 5.26633964e+05 2.20832794e+03
  2.54563046e+04 2.32683564e+07 2.11484498e+04 8.08373189e-01
  8.70628432e+02 2.50894552e+02]
 [6.04449717e+05 5.07215057e+00 1.10492833e+03 1.62968112e+05
  6.50886125e+00 6.49917002e+03 7.92024810e+05 6.76773187e+01
  3.23874665e+02 2.56940287e+05]
 [1.44606647e+04 7.97838227e+03 1.07888516e+01 5.26633964e+05
  3.05495065e+04 1.28666098e+07 3.97955983e+01 1.80702704e+01
  2.97779025e+05 2.11484498e+04]
 [3.93366218e-01 8.49590170e+06 4.40843348e+06 2.54563046e+04
  4.57991148e+05 1.43324217e+00 2.38313319e-01 1.57365074e+07
  8.82218690e+01 1.43324217e+00]
 [8.82218690e+01 1.90090548e+05 1.71225209e+06 1.33262411e+06
  8.08373189e-01 6.92517140e+05 2.11305399e+07 1.03072196e+06
  6.49917002e+03 1.73775929e+07]
 [8.60188419e+04 9.76040682e+03 7.28720782e+04 1.17292656e+06
  8.82218690e+01 7.92024810e+05 6.50886125e+00 5.26633964e+05
  2.56940287e+05 8.82218690e+01]
 [2.40397444e+00 2.76078712e+03 9.43987454e+06 1.39400192e+05
  6.15757328e+04 2.77748331e+06 6.76773187e+01 1.39705727e+03
  7.28720782e+04 4.57991148e+05]
 [1.44606647e+04 4.93242050e+06 1.93651777e+06 3.93520561e+06
  5.27554593e+03 6.15757328e+04 7.28720782e+04 8.70628432e+02
  7.92024810e+05 3.65542522e+04]]
```

❖ Task 1c

Code:

```
print("c)")
#c
odd_row = A[1::2]

print("odd_row = ", odd_row, sep="\n")
print()
```

Màn hình chạy:

```
c)
odd_row =
[[53 68 82 20 71 39 52 74 98 68]
 [68 12 33 59 13 41 70 22 28 62]
 [ 3 90 84 48 66  7  2 96 23  7]
 [55 43 54 73 23 70 13 67 62 23]
 [45 85 77 83 40 53 54 32 70 50]]
```

❖ Task 1d

Code:

```
print("d)")
#d
odd_vector = []
for i in A.flatten():
  if(i % 2 != 0):
    odd_vector.append(i)
print("odd_vector =", odd_vector)
print()
```

Màn hình chạy:

```
d)
odd_vector = [13, 65, 5, 79, 53, 71, 39, 83, 67, 47, 5, 27, 33, 59, 13, 41, 45, 15, 67, 49, 17, 63, 47, 3, 7, 23, 7, 23, 5, 69, 99, 41, 97, 55, 43, 73, 23, 13, 67, 23, 9, 37, 91, 53, 45, 85, 77, 83, 53]
```

❖ Task 1e

Code:

```
print("e)")
# e
def is_prime(n):
  if n<=1:
    return False
  for i in range(2, int(np.sqrt(n))+1):
    if n % i == 0:
      return False
  return True

prime_vectorA = []
for x in A.flatten():
  if(is_prime(x)):
    prime_vectorA.append(x)
print("prime_vectorA =", prime_vectorA)
print()
```

Màn hình chạy:

```
e)
prime_vectorA = [13, 5, 79, 53, 71, 83, 67, 47, 5, 59, 13, 41, 67, 17, 47, 3, 7, 2, 23, 7, 23, 5, 41, 97, 43, 73, 23, 13, 67, 23, 37, 53, 83, 53]
```

❖ Task 1f

Code:

```
print("f)")
#f
D = np.dot(C, B)

print("D = ", D, sep="\n")

for i in range(D.shape[0]):
  if i % 2 != 0 :
    D[i] = D[i][::-1]

print()
print("After reverse the odd rows D = ", D, sep="\n")
print()
```

Màn hình chạy:

```
f)
D =
[[176 302  25 256 381 168 274 186 224 286]
 [ 45  72   9  84 129  87  66  69  99 105]
 [ 71 116  13 124 189 117 106  99 137 151]
 [200 344  28 288 428 184 312 208 248 320]
 [196 340  26 272 402 156 308 192 220 296]
 [ 84 120  24 208 328 284 112 188 300 284]
 [236 398  37 368 553 284 362 278 356 426]
 [252 432  36 368 548 244 392 268 324 412]
 [150 240  30 280 430 290 220 230 330 350]
 [232 412  26 288 418 104 372 188 184 292]]

After reverse the odd rows D =
[[176 302  25 256 381 168 274 186 224 286]
 [105  99  69  66  87 129  84   9  72  45]
 [ 71 116  13 124 189 117 106  99 137 151]
 [320 248 208 312 184 428 288  28 344 200]
 [196 340  26 272 402 156 308 192 220 296]
 [284 300 188 112 284 328 208  24 120  84]
 [236 398  37 368 553 284 362 278 356 426]
 [412 324 268 392 244 548 368  36 432 252]
 [150 240  30 280 430 290 220 230 330 350]
 [292 184 188 372 104 418 288  26 412 232]]
```

❖ Task 1g

Code:

```
print("g)")
# g
max_count = 0
max_rows = []

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

    count = 0
    for j in range(A.shape[1]):
        if is_prime(A[i,j]):
            count += 1
    if count > max_count:
        max_count = count
        max_rows = [A[i, :]]
    elif count == max_count:
        max_rows.append(A[i, :])

for row in max_rows:
    print(row)
print()
```

Màn hình chạy:

```
h)
[68 12 33 59 13 41 70 22 28 62]
[45 85 77 83 40 53 54 32 70 50]
```

❖ Task 1h

Code:

```
print("h)")
# h
maxOdd_len = 0
maxOdd_rows = []

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

    for j in range(A.shape[1]):
        if A[i,j] % 2 != 0:
            currentOdd_len += 1
            if currentOdd_len > maxOdd_in_row:
                maxOdd_in_row = currentOdd_len
        else:
            currentOdd_len = 0
    if maxOdd_in_row > maxOdd_len:
        maxOdd_len = maxOdd_in_row
        maxOdd_rows = [A[i, :]]
    elif maxOdd_in_row == maxOdd_len:
        maxOdd_rows.append(A[i,:])

for row in maxOdd_rows:
    print(row)
```

Màn hình chạy:

```
g)
[55 43 54 73 23 70 13 67 62 23]
```