TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN ĐHQG-HCM

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN 1 – MA TRẬN BẬC THANG & GAUSS

Môn Học: Toán ứng dụng & thống kê

Giảng viên:

Nguyễn Trọng Hiến

Nguyễn Văn Quang Huy

Nguyễn Đình Thúc

Võ Nam Thục Đoan

Sinh viên thực hiện:

Nguyễn Tấn Phát 20127588

1. Nội dung đồ án:

1) Sinh viên viết hàm Gauss_elimination(A), trong đó

Input: A ma trận mở rộng của hệ phương trình.

Output: ma trận có dạng bậc thang có được từ ma trận A.

2) Sinh viên viết hàm back_substitution(A), trong đó

Input: A ma trận có dạng bậc thang từ từ ma trận mở rộng của hệ phương trình Ax=b.

Output: nghiệm của hệ phương trình (trường hợp nghiệm duy nhất/ vô số nghiệm) hoặc thông báo hệ phương trình vô nghiệm.

Sinh viên không được dùng các hàm có sẵn của các thư viện để tìm ma trận bậc thang.

2. Môi trường làm việc:

- Ngôn ngữ lập trình: Python
- Text Editer: Visual Studio Code
- Thư viện hỗ trợ: không

3. Cơ sở và ý tưởng:

a) Xây dựng class và các đối tượng

- Tên class: **SystemLinearEquations** (Hệ phương trình tuyến tính)
- **self.arr_input**: List lưu ma trận A từ input.
- **self.solutions**: List lưu các nghiệm của hệ phương trình tuyến tính.
- **self.numSolutions**: số nghiệm của hệ.
- **self.numEquation**: số phương trình tuyến tính khác 0.
- **self.RowZero**: phương trình tuyến tính bằng 0 (toàn bộ bằng 0)

b) Ý tưởng giải quyết bài toán

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} \Leftrightarrow AX = B$$

- Lấy dữ liệu từ file input.txt rồi đưa vào litst self.arr_input với:
  - Mỗi phần tử là một dòng của hệ phương trình tuyến tính
  - VD: self.arr_input = [[1, 2, 3], [4, 5, 6]]
- Đưa self.arr_input về ma trận bậc thang
  - VD self.arr_input = [[1, 0 , 2],[0, 1, 1]]
  - Phải loại bỏ đi tất các các dòng bằng 0.
- Tính nghiệm rồi đưa vào self.solutions theo công thức:
  - $x_n = a_{n,(n+1)}$
  - $x_i = [a_{i,(n+1)} - \sum_{j=i+1}^{n} a_{i,j} * x_j] \,/\, a_{i,i} (\forall i = n-1, n-2, \dots, 1)$
  - Phải sử dụng hàm đệ quy!
- Tuy nhiên, đây chỉ là lý thuyết và có đến 3 TH nghiệm nên phải xử lí từng trường hợp
  - 1 nghiệm duy nhất – khi:
    - Số ẩn bằng số phương trình tuyến tính khác 0.
    - Các nghiệm là số thực.
  - Vô số nghiệm – khi:
    - Số ẩn > số phương trình tuyến tính khác 0.
    - Các nghiệm được biểu theo các biến thuộc R.
    - Khi code phải sử dụng kiểu chuỗi str.
  - Vô nghiệm – khi:
    - Số ẩn < số phương trình khác 0.
    - Phải thực hiện chia cho 0 (a[i][i] = 0)
    - Đặt tất các các nghiệm bằng None

4. Giải thích các hàm:

a. *getInput(self, filename: str)*

- *Chức năng: Lấy thông tin từ file input.txt rồi chuyển về dạng list phù hơn để gán vào self.arr_input.*
- Input: Số dòng của hệ phương trình tuyến tính, hệ số từng dòng của hệ
- Output: self.arr_input
- VD:
  - Input: 2 \n 1, 2, 3 \n 2, 3, 4
  - Output: self.arr_input =[[1,2,3],[2,3,4]]

b. *echolonMatrix(self)*

- Chức năng: Đưa ma trận của self.arr_input thành ma trận bậc thang.
- Lấy dòng đầu tiên (số 1) làm tiêu chuẩn:
  - Nếu phần tử thứ 1 của dòng đó bằng 0 thì ta đưa dòng đó về cuối list và đẩy dòng khác lên thay thế.
  - Nếu toàn bộ dòng đều có phần tử thứ 1 đều bằng 0 thì bỏ qua dòng này và qua dòng tiếp theo (số 2)

- Dùng vòng for để duyệt các dòng còn lại của list, mục đích là trừ đi k*dòng 1 để tất cả các phần tử thứ 1 các dòng đó đề bằng 0:
  - heso = self.arr_input[j][i] / self.arr_input[i][i]
  - Với i là chỉ số dòng đang làm tiêu chuẩn (số 1)
  - Và j là chỉ số dòng được xét trong vòng for phía trên.
  - self.arr_input[j][k] −= self.arr_input[i][k] * heso
  - Với k là chỉ số từng phần tử của dòng đang được xét trong vòng for.
- Tiếp tục, lấy dòng 2 làm tiêu chuẩn, rồi dòng 3,…
- Đôi khi, sẽ có các dòng bằng 0. Do đó phải thực hiện xóa nó đi khỏi list self.arr_input.

c. *onlySolution(self, i, n, arr_solve, arr_input)*

- Chức năng: Tính toán nghiệm khi trường hợp “một nghiệm duy nhất xảy ra”
- Trong đó:
  - i : Chỉ phần tử nghiệm đang xét và nó sẽ giảm dần về 0 vì đệ quy.

  - n : chỉ số ẩn của hệ phương trình tuyến tính.
  - arr_solve: list các nghiệm của hệ.
  - arr_input: chính là nơi self.arr_input sẽ truyền vào.
- Sử dụng công thức đệ quy phần 3b để tính nghiệm.

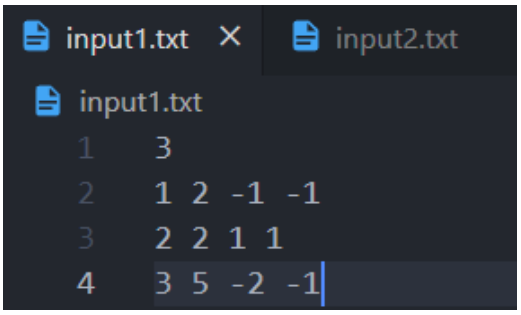
d. manySolution(self, free, i, n, arr_solve, arr_input)

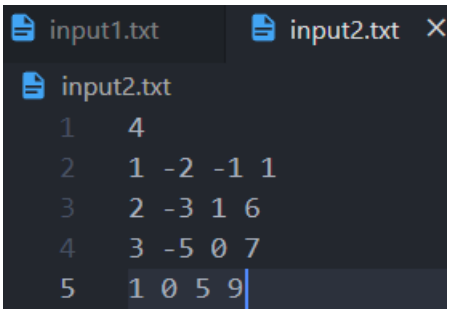
- Chức năng: Tính toán nghiệm khi trường hợp "vô số nghiệm xảy ra"
- Các tham số truyền vào có chức năng như hệt phía trên:
  - free : Đếm số các nghiệm tự do hiện chưa có.
- Cũng sử dụng công thức
- Do nghiệm là các chuỗi str nên phải viết các hàm nhỏ hỗ trợ:
  - subStr(number): trừ number
  - dev(a, b): chia a cho b (a/b)
  - mul(a, b): nhân a với b (a*b)

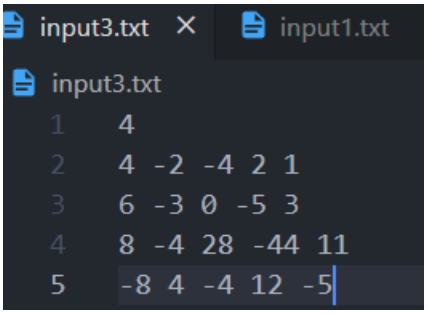
e. Algorithm(self)

- Chức năng: Giải hệ phương trình tuyến tính bằng Gauss.
- Đưa về ma trận bậc thang.
- Kiểm tra xem kết quả rơi vào TH nào:
  - 1 nghiệm duy nhất
  - Vô số nghiệm
  - Vô nghiệm
- Trả về nghiệm phù hợp

5. Kết quả chạy thử:







```python
def main():
    sle = SystemsLinearEquations()

    filename = "input1.txt"
    sle.getInut(filename)
    result = sle.Algorithm()
    for i in range(len(result)):
        print('x[' + str(i + 1) + '] = ', result[i])
    sle.clear()

    print("\n||||||||||||||||||||||||||||\n")

    filename = "input2.txt"
    sle.getInut(filename)
    result = sle.Algorithm()
    for i in range(len(result)):
        print('x[' + str(i + 1) + '] = ', result[i])
    sle.clear()

    print("\n||||||||||||||||||||||||||||\n")

    filename = "input3.txt"
    sle.getInut(filename)
    result = sle.Algorithm()
    for i in range(len(result)):
        print('x[' + str(i + 1) + '] = ', result[i])
    sle.clear()

    print("\n||||||||||||||||||||||||||||\n")
```

```
[1.0, 2.0, -1.0, -1.0]
[2.0, 2.0, 1.0, 1.0]
[3.0, 5.0, -2.0, -1.0]
-----------
[1.0, 2.0, -1.0, -1.0]
[0.0, -2.0, 3.0, 3.0]
[0.0, 0.0, -0.5, 0.5]
One Solution
x[1] =  4.0
x[2] =  -3.0
x[3] =  -1.0

||||||||||||||||||||||||||||

[1.0, -2.0, -1.0, 1.0]
[2.0, -3.0, 1.0, 6.0]
[3.0, -5.0, 0.0, 7.0]
[1.0, 0.0, 5.0, 9.0]
-----------
[1.0, -2.0, -1.0, 1.0]
[0.0, 1.0, 3.0, 4.0]
Many Solutions
x[1] =  (1.0 + 1.0 * a1 + 2.0 * (4.0 - 3.0 * a1))
x[2] =  (4.0 - 3.0 * a1)
x[3] =  a1
```

```
[4.0, -2.0, -4.0, 2.0, 1.0]
[6.0, -3.0, 0.0, -5.0, 3.0]
[8.0, -4.0, 28.0, -44.0, 11.0]
[-8.0, 4.0, -4.0, 12.0, -5.0]
-----------
[4.0, -2.0, -4.0, 2.0, 1.0]
[0.0, 0.0, -12.0, 16.0, -3.0]
[0.0, 0.0, 6.0, -8.0, 1.5]
No Solution
x[1] =  ?
x[2] =  ?
x[3] =  ?
x[4] =  ?
```

**********HẾT**********