Phần I

Câu 1:

$$A=\begin{bmatrix}6 & 5\\ 3 & 2\end{bmatrix};\ B=\begin{bmatrix}4 & 3\\ 2 & 1\end{bmatrix}$$

$$AB=\begin{bmatrix}34 & 23\\ 16 & 11\end{bmatrix}=>tr(AB)=34+11=45$$

$$BA=\begin{bmatrix}33 & 26\\ 15 & 12\end{bmatrix}=>tr(BA)=33+12=45$$

Vậy $tr(AB)=tr(BA)$

Câu 2:

Gọi a_{ij} ; b_{ij} với $1\le i,j\le n$ lần lượt là các tử của 2 ma trận A,B

$$A=\begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix};\ B=\begin{bmatrix}b_{11} & b_{12}\\ b_{21} & b_{22}\end{bmatrix}$$

$$AB=\begin{bmatrix}a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22}\\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22}\end{bmatrix}=>tr(AB)=a_{11}b_{11}+a_{12}b_{21}+a_{21}b_{12}+a_{22}b_{22}$$

$$BA=\begin{bmatrix}a_{11}b_{11}+a_{21}b_{12} & a_{12}b_{11}+a_{22}b_{12}\\ a_{11}b_{21}+a_{21}b_{22} & a_{12}b_{21}+a_{22}b_{22}\end{bmatrix}=>tr(AB)=a_{11}b_{11}+a_{21}b_{12}+a_{12}b_{21}+a_{22}b_{22}$$

Vậy $tr(AB)=tr(BA)$

Câu 3:

Gọi a_{ij} ; b_{ij} với $1 \le i, j \le n$ lần lượt là các tử của 2 ma trận A,B

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}; \; B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{bmatrix}$$

Ta có $[AB]_{ij} = \sum_{h=1}^{n} a_{ih} b_{hj}$; $[BA]_{ij} = \sum_{k=1}^{n} a_{kj} b_{ik}$ và $tr(AB) = \sum_{g=1}^{n} [AB]_{gg}$; $tr(BA) = \sum_{l=1}^{n} [BA]_{ll}$

Suy ra $tr(AB) = \sum_{g=1}^{n} [AB]_{gg} = \sum_{g=1}^{n} \sum_{h=1}^{n} a_{gh} b_{hg}$ và $tr(BA) = \sum_{l=1}^{n} [BA]_{ll} = \sum_{l=1}^{n} \sum_{k=1}^{n} a_{kl} b_{lk}$

Mà ta lại chứng minh được với $n = 2$ thì $tr(AB) = tr(BA)$.(Bài 2)

*Ta sử dụng qui nạp để chứng minh bài toán.

Giả sử $tr(AB) = tr(BA)$ (1) đúng tới $n = t$.

Ta cần chứng minh $tr(AB) = tr(BA)$ đúng với $n = t + 1$

Từ giả thiết qui nạp, ta có $\sum_{g=1}^{t} \sum_{h=1}^{t} a_{gh} b_{hg} = \sum_{l=1}^{t} \sum_{p=1}^{t} a_{pl} b_{lp}$

Ta cộng thêm 2 vế của phương trình với

$$\sum_{k=1}^{t} a_{k(t+1)} b_{(t+1)k} + \sum_{k=1}^{t} a_{(t+1)k} b_{k(t+1)} + a_{(t+1)(t+1)} b_{(t+1)(t+1)}$$

Ta được:

$$VT = \sum_{g=1}^{t}\sum_{h=1}^{t} a_{gh}b_{hg} + \sum_{k=1}^{t} a_{k(t+1)}b_{(t+1)k} + \sum_{k=1}^{t} a_{(t+1)k}b_{k(t+1)} + a_{(t+1)(t+1)}b_{(t+1)(t+1)}$$

$$= \left(\sum_{h=1}^{t} \left(a_{1h}b_{h1} + a_{2h}b_{h2} + ... + a_{th}b_{ht} \right) + \sum_{k=1}^{t+1} a_{k(t+1)}b_{(t+1)k} \right) + \left(\sum_{k=1}^{t} a_{(t+1)k}b_{k(t+1)} + a_{(t+1)(t+1)}b_{(t+1)(t+1)} \right)$$

$$= \sum_{h=1}^{t+1} \left(a_{1h}b_{h1} + a_{2h}b_{h2} + ... + a_{th}b_{ht} \right) + \sum_{k=1}^{t+1} a_{(t+1)k}b_{k(t+1)}$$

$$= \sum_{g=1}^{t}\sum_{h=1}^{t+1} a_{gh}b_{hg} + \sum_{k=1}^{t+1} a_{(t+1)k}b_{k(t+1)}$$

$$= \sum_{g=1}^{t+1}\sum_{h=1}^{t+1} a_{gh}b_{hg}$$

Ta có được điều tương tự ở vế phải của phương trình

$$VP = \sum_{l=1}^{t}\sum_{p=1}^{t} a_{pl}b_{lp} + \sum_{k=1}^{t} a_{k(t+1)}b_{(t+1)k} + \sum_{k=1}^{t} a_{(t+1)k}b_{k(t+1)} + a_{(t+1)(t+1)}b_{(t+1)(t+1)}$$

$$= \left(\sum_{l=1}^{t} \left(a_{1l}b_{l1} + a_{2l}b_{l2} + ... + a_{tl}b_{lt} \right) + \sum_{k=1}^{t+1} a_{k(t+1)}b_{(t+1)k} \right) + \left(\sum_{k=1}^{t} a_{(t+1)k}b_{k(t+1)} + a_{(t+1)(t+1)}b_{(t+1)(t+1)} \right)$$

$$= \sum_{l=1}^{t+1} \left(a_{1l}b_{l1} + a_{2l}b_{l2} + ... + a_{tl}b_{lt} \right) + \sum_{k=1}^{t+1} a_{(t+1)k}b_{k(t+1)}$$

$$= \sum_{p=1}^{t}\sum_{l=1}^{t+1} a_{pl}b_{lp} + \sum_{k=1}^{t+1} a_{(t+1)k}b_{k(t+1)}$$

$$= \sum_{g=1}^{t+1}\sum_{h=1}^{t+1} a_{pl}b_{lp}$$

Vậy với 2 ma trận A,B vuông nxn, ta luôn có $tr\left(AB\right) = tr\left(BA\right)$

Câu 4:

Gọi a_{ij}, b_{ij}, c_{ij} với $1 \le i, j \le n$ lần lượt là các tử của 3 ma trận A,B,C

Ta có được:

$$\begin{cases} [AB]_{ij} = \sum_{h=1}^{n} a_{ih} b_{hj} \\ [BC]_{ij} = \sum_{h=1}^{n} b_{ih} c_{hj} \\ [CA]_{ij} = \sum_{h=1}^{n} c_{ih} a_{hj} \end{cases}$$

Suy ra:

$$\begin{cases} [ABC]_{ij} = \sum_{k=1}^{n} \sum_{h=1}^{n} a_{ih} b_{hk} c_{kj} \\ [BCA]_{ij} = \sum_{k=1}^{n} \sum_{h=1}^{n} b_{ih} c_{hk} a_{kj} \\ [CAB]_{ij} = \sum_{k=1}^{n} \sum_{h=1}^{n} b_{ih} a_{hk} b_{kj} \end{cases}$$

Khi đó, ta được:

$$\begin{cases} tr(ABC) = \sum_{i=1}^{n} [ABC]_{ii} = \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{h=1}^{n} a_{ih} b_{hk} c_{ki} \\ tr(BCA) = \sum_{i=1}^{n} [BCA]_{ii} = \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{h=1}^{n} b_{ih} c_{hk} a_{ki} \\ tr(CAB) = \sum_{i=1}^{n} [CAB]_{ii} = \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{h=1}^{n} c_{ih} a_{hk} b_{ki} \end{cases}$$

Gọi $a_{mp} b_{pq} c_{qm}$ là một số hạng bất kì trong biểu thức $tr(ABC)$. Ta sẽ chứng minh có số hạng giống như vậy trong biểu thức $tr(BCA)$ và $tr(CAB)$.

Trong biểu thức $tr(BCA)$, với $i = p; k = m; h = q$, sẽ có số hạng $b_{pq} c_{qm} a_{mp}$ duy nhất.

Tương tự, Trong biểu thức $tr(CAB)$, với $i=q;k=p;h=m$, sẽ có số hạng $c_{qm}a_{mp}b_{pq}$ duy nhất

Vậy ta đã chứng minh được cả 3 biểu thức $tr(ABC),tr(CAB),tr(BCA)$ có chung tất cả số hạng.

Do đó, $tr(ABC)=tr(CAB)=tr(BCA)$.

Câu 5:

$$A=\begin{bmatrix}1 & -1\\ 5 & 4\end{bmatrix};\ B=\begin{bmatrix}2 & 0\\ 3 & 9\end{bmatrix};C=\begin{bmatrix}8 & 1\\ 2 & 3\end{bmatrix}$$

Ta có:

$$AB=\begin{bmatrix}-1 & -9\\ 22 & 36\end{bmatrix}$$

$$CB=\begin{bmatrix}19 & 9\\ 13 & 27\end{bmatrix}$$

Suy ra:

$$ABC=\begin{bmatrix}-26 & -28\\ 248 & 130\end{bmatrix}=>tr(CBA)=-26+130=104$$

$$CBA=\begin{bmatrix}64 & 17\\ 148 & 95\end{bmatrix}=>tr(CBA)=64+95=159$$

Vậy $tr(ABC)\neq tr(CBA)$

REFERENCE

[1] Prince, E., & Prince, E. (2004). Matrices: Definitions and Fundamental Operations. *Mathematical Techniques in Chrystallography and Materials Science*, 1-19.

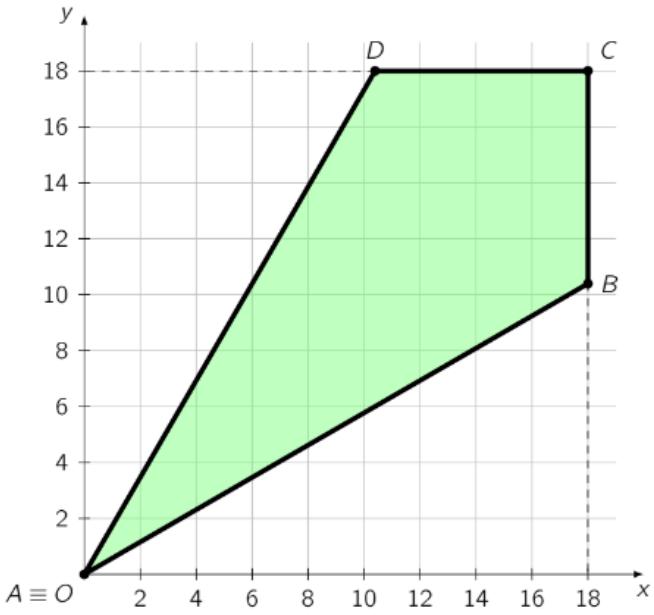
Phần II

Câu 1:

a)

Ta tìm tọa độ các đỉnh của tứ giác ABCD

Ta đã có tọa độ 2 đỉnh là $A(0;0)$ và $C(18;18)$



Hình 1: Hình minh họa cho Câu 1

Dựa vào đồ thị và giả thiết:

$$\begin{cases} AB = AD = 12\sqrt{3} \\ CB = CD = 18 - 6\sqrt{3} \\ A(0;0) \\ C(18;18) \end{cases}$$

Mà ta lại có:

$$\begin{cases} x_B{}^2 + y_B{}^2 = AB^2 \\ y_C - y_B = CB \end{cases} \text{ và } \begin{cases} x_D{}^2 + y_D{}^2 = AD^2 \\ x_C - x_D = CD \end{cases}$$

Ta suy ra được:

$$\begin{cases} x_B = 18 \\ y_B = 6\sqrt{3} \end{cases} \text{ và } \begin{cases} x_D = 6\sqrt{3} \\ y_D = 18 \end{cases}$$

Vậy có tọa độ 2 đỉnh còn lại là $C\left(18;6\sqrt{3}\right)$ và $D\left(6\sqrt{3};18\right)$

Ta tìm phương trình đường thẳng chứa các cạnh của tứ giác ABCD:

Đường thẳng AB: $y=\frac{6\sqrt{3}}{18}\left(x-0\right)+0 \Rightarrow \frac{\sqrt{3}}{3}x-y=0$

Đường thẳng BC: $x=18$

Đường thẳng CD: $y=18$

Đường thẳng DA: $y=\frac{18}{6\sqrt{3}}\left(x-0\right)+0 \Rightarrow \sqrt{3}x-y=0$

Dựa vào đồ thị ta có thể biểu diễn miền nghiệm được bao bởi tứ giác ABCD bằng một hệ bất phương trình như sau :

$$\begin{cases} x\le 18 \\ y\le 18 \\ y-\frac{\sqrt{3}}{3}x\ge 0 \\ y-\sqrt{3}x\le 0 \end{cases}$$

b)

Khoảng cách từ S tới các cạnh lần lượt phải là một số dương(Vì S là một điểm nằm trong tứ giác nên s1,s2 phải là nghiệm của hệ bất phương trình ở câu a)

Ta tính khoảng cách từ đỉnh S tới các cạnh của tứ giác ABCD:

$$d_{[S;AB]}=\frac{\left|s_2-\frac{\sqrt{3}}{3}s_1\right|}{\sqrt{1+\left(\frac{\sqrt{3}}{3}\right)^2}}$$

$$d_{[S;BC]}=\frac{\left|18-s_1\right|}{1}$$

$$d_{[S;CD]}=\frac{\left|18-s_2\right|}{1}$$

$$d_{[S;DA]}=\frac{\left|\sqrt{3}s_1-s_2\right|}{2}$$

Từ điều kiện trên ở câu a như đã lập luận, ta xét dấu và phá trị tuyệt đối của biểu thức, ta được:

$$d_{[S;AB]} = \frac{\left| s_2 - \frac{\sqrt{3}}{3} s_1 \right|}{\sqrt{1 + \left(\frac{\sqrt{3}}{3} \right)^2}} = \frac{\sqrt{3} s_2 - s_1}{2}$$

$$d_{[S;BC]} = \frac{|18 - s_1|}{1} = 18 - s_1$$

$$d_{[S;CD]} = \frac{|18 - s_2|}{1} = 18 - s_2$$

$$d_{[S;DA]} = \frac{\left| \sqrt{3} s_1 - s_2 \right|}{2} = \frac{\sqrt{3} s_1 - s_2}{2}$$

c)

Vì S là một điểm nằm trong tứ giác nên tọa độ của điểm $S(s_1, s_2)$ phải là nghiệm của hệ bất phương trình ở câu (a):

$$\begin{cases} s_1 \le 18 \\ s_2 \le 18 \\ s_2 - \frac{\sqrt{3}}{3} s_1 \ge 0 \\ s_2 - \sqrt{3} s_1 \le 0 \end{cases}$$

$$<=> \begin{cases} s_1 \le 18 \\ s_2 \le 18 \\ \sqrt{3} s_2 - s_1 \ge 0 \\ s_2 - \sqrt{3} s_1 \le 0 \end{cases}$$

$$<=> \begin{cases} s_1 \le 18 \\ s_2 \le 18 \\ \sqrt{3} s_2 - s_1 \ge 0 \\ -s_2 + \sqrt{3} s_1 \ge 0 \end{cases}$$

Ta có r là bán kính của cái đế dự kiến được cắt. Do đó,r không được vượt quá khoảng cách của điểm S tới các cạnh của tứ giác ABCD. Ta lập được hệ bất phương trình như sau:

$$\begin{cases} r \le \dfrac{\sqrt{3}s_2 - s_1}{2} \\ r \le 18 - s_1 \\ r \le 18 - s_2 \\ r \le \dfrac{\sqrt{3}s_1 - s_2}{2} \\ r \ge 0 \end{cases}$$

Vậy để cắt được một cái đế hình tròn từ tứ giác ABCD có tâm là $S(s_1, s_2)$ và bán kính r, 3 giá trị s_1, s_2, r phải thỏa hệ phương trình:

$$\begin{cases} r \le \dfrac{\sqrt{3}s_2 - s_1}{2} \\ r \le 18 - s_1 \\ r \le 18 - s_2 \\ r \le \dfrac{\sqrt{3}s_1 - s_2}{2} \\ r \ge 0 \end{cases}$$

Câu 2:

- Để dễ dàng trong việc giải thích, ta sẽ điền các giá trị của bộ k^2 số vào ma trận vuông A có kích thước k x k có a_{ij} là các phần tử với $1 \le i \le k, 1 \le j \le k$ theo qui tắc $A_{ij} = x_{ij}$.

  Khi đó, ta có thể hiểu các điều kiện của bài toán như sau:

1) $x_{ij} \in \{0;1\}$ với mọi i,j thỏa đề

   Vậy các phần tử trong ma trận chỉ mang giá trị 0 hoặc 1

2) $\sum_{i=1}^{k} x_{ij} = 1$ với mọi j thỏa đề

Tổng các phần tử có cùng giá trị j luôn bằng 1 với mọi j thỏa đề

Vậy ta suy ra được trên mỗi cột của ma trận, chỉ có đúng một phần tử mang giá trị bằng 1

3) $\sum_{j=1}^{k} x_{ij} = 1$ với mọi i thỏa đề

Tổng các phần tử có cùng giá trị i luôn bằng 1 với mọi i thỏa đề

Vậy ta suy ra được trên mỗi hàng của ma trận, chỉ có đúng một phần tử mang giá trị bằng 1

Từ đây, ta có thể hiểu biến số $x_{ij} \in \{0;1\}$ là một giá trị để chỉ bạn A_i có làm công việc j hay không, theo nguyên tắc sau:

- Nếu bạn A_i là người được phân công làm công việc j , $x_{ij} = 1$.
- Nếu bạn A_i không phải là người được phân công làm công việc j , $x_{ij} = 0$.
- Một bạn chỉ được phân công đúng một công việc
- Một công việc chỉ được phân công cho đúng một người

Nhận thấy z là tổng thời gian của các bạn để hoàn thành được nhiệm vụ mình được giao.

Để z đạt giá trị nhỏ nhất thì ta cần tìm phương án tối ưu để phân công nhiệm vụ sao cho tổng thời gian hoàn thành của mọi người ngắn nhất.

REFERENCE

[1] Eisenhart, L. P. (2005). Coordinate geometry. Courier Corporation.

Phần III

Câu 1:

Vì bài toán có hữu hạn phương án tối ưu nên ta sẽ luôn tìm được phương án tối ưu của hệ bất phương trình là đỉnh hình đa giác của miền nghiệm hoặc hệ bất phương trình vô nghiệm

Ta xây dựng thuật toán như sau:

a) Tìm tất cả các đỉnh của đa giác:
   1) Dùng câu lệnh if loại bỏ những đường thẳng song song hoặc trùng nhau
   2) Tìm tọa giao điểm của các cặp đường thẳng giao nhau và lưu lại
   3) Thế từng cặp tọa độ vào hệ bất phương trình, nếu không thỏa hệ bất phương trình thì bỏ
   4) Nếu tất cả các tọa độ điểm đều không thỏa thì hệ phương trình vô nghiệm (INVALID)
   5) Các tọa độ thỏa hệ bất phương trình chính là các đỉnh của đa giác
b) Xác định giá trị cao nhất
   1) Đặt một giá trị của một cặp tọa độ đỉnh khi thế vào hàm mục tiêu là max
   2) Tiếp tục so với các giá trị của các tọa độ đỉnh khác cho tới hết để tìm giá trị lớn nhất
   3) In giá trị max

Khi đó, ta có thể viết lại dưới dạng mã giả như sau:

Hàm LPMax2Var(alpha, beta, constraint)

Khởi tạo danh sách list_vertices trống để lưu tọa độ các giao điểm

Đặt biến đếm số lượng hàm điều kiện là num

// Tìm tọa độ giao điểm của các đỉnh của miền nghiệm

Gọi lần lượt từng cặp đường thẳng và xét cho chúng có song song hay không

Nếu không. Ta dung hàm intersect_two_lines tìm tọa độ giao điểm

Lưu tọa độ này vào list_vertices

// Lọc các điểm không nằm trong miền nghiệm

Khởi tạo danh sách vertices trống để lưu các điểm hợp lệ

Thay mỗi cặp tọa độ (x, y) trong list_vertices vào từng hàm điều kiện

Nếu thỏa hết các hàm điều kiện:

Thêm (x, y) vào danh sách vertices

// Tìm giá trị tối ưu

Tạo optimalVal là giá trị tối ưu

Khởi tạo optimalVal là giá trị của hàm mục tiêu khi thế tọa độ đỉnh đầu tiên trong vertices

Cho từng cặp tọa độ (x, y) trong vertices

Tính giá trị hàm mục tiêu tại (x, y): val = alpha * x + beta * y

So sánh val với optimalVal

Nếu val > optimalVal

Cập nhật optimalVal = val

Tiếp tục cho tới hết tất cả các tọa độ trong vertices

Trả về optimalVal

Kết thúc hàm

// Nhập giá trị theo đúng yêu cầu gồm hệ số trong hàm mục tiêu và các hàm điều kiện

Thực hiện hàm LPMax2Var(alpha, beta, constraint) và in ra giá trị tối ưu optimalVal

Câu 2:

Ta có thể chứng minh được đối với bài toán quy hoạch tuyến tính có giá trị tối ưu hữu hạn, phương án tối ưu sẽ luôn nằm ở đỉnh của đa giác hay miền nghiệm của hệ bất phương trình thông qua việc biểu diễn trực quan hệ bất phương trình và hàm mục tiêu lên cùng một mặt phẳng tọa độ Oxy.

Vì bài toán có hữu hạn phương án tối ưu nên ta sẽ luôn tìm được phương án tối ưu của hệ bất phương trình là đỉnh hình đa giác của miền nghiệm. Khi đó, ta thế tọa độ các giao điểm thỏa hệ bất phương trình vào hàm mục tiêu , ta sẽ tìm được giá trị lớn nhất (Chắc chắn có đối với bài toán quy hoạch tuyến tính có giá trị tối ưu hữu hạn) cũng chính là giá trị tối ưu cần tìm.

Câu 3:

Link code GitHub: Phần III Câu 3

Thuật toán đã được giải thích ở Câu 2

RAW CODE

```
import sys
def intersectTwoLine(a1, b1, c1, a2, b2, c2):
  det = a1 * b2 - a2 * b1
  if det!=0 or ( det==0 and (a1==0 or a2==0 or b1==0 or b2==0)):
    x0 = (b2 * c1 - b1 * c2) / det
    y0 = (a1 * c2 - a2 * c1) / det
    return x0, y0
  else:
      return
def LPMax2Var (alpha , beta , constraint ):
    a=[]
    b=[]
    c=[]
    x=[]
    y=[]
    g=0
    x0=[]
    y0=[]
    # Tìm tọa độ giao điểm của các đỉnh của miền nghiệm
    for i in constraint:
            a.append(i[0])
            b.append(i[1])
            c.append(i[2])
            g+=1
    for l in range(g):
        for j in range(l,g):
            if a[l]*b[j]-a[j]*b[l] != 0:
                o= intersectTwoLine(a[l],b[l],c[l],a[j],b[j],c[j])
                x.append(o[0])
```

```
                y.append(o[1])
    j=0
    for k in range(len(x)):
        for l in range(len(a)):
            if a[l]*x[k]+b[l]*y[k]-c[l]<=0:
                j+=1
        if j== len(a):
            x0.append(x[k])
            y0.append(y[k])
        j=0
    # print(x0,y0)
    optimalVal=alpha*x0[0]+beta*y0[0]
    # print(optimalVal)
    for q in range(len(x0)):
        if alpha*x0[q]+beta*y0[q] > optimalVal:
            optimalVal=alpha*x0[q]+beta*y0[q]
    return optimalVal
print(LPMax2Var (6, 25, [[3, 5, 240] , [0, 1, 12], [-1, 0, 0], [0, -1, 0]])
```

OUTPUT

```
Giá trị tối ưu: 660.0
```

REFERENCE

[1] Eisenhart, L. P. (2005). Coordinate geometry. Courier Corporation.

[2] Sarlette, A. (2009). Geometry and symmetries in coordination control.

[3] Gowrishankar, S., & Veena, A. (2018). Introduction to Python programming. Chapman and Hall/CRC.

Phần IV

Câu 1:

a_{ij} với $1 \le i \le n, 1 \le j \le n$ và $b_1, b_2, ..., b_m$ lần lượt là các hệ số của các ẩn và kết quả (giá trị ở vế phải hay hệ số tự do) sau khi hệ phương trình được đưa về dạng bậc thang.

Gọi l là số phương trình trong hệ phương trình bậc thang có hệ số $k_i < n$

Mặt khác, dựa vào điều kiện của hệ phương trình bậc thang, ta nhận thấy rằng ta chỉ nhận được tối đa n số nguyên $k_i < n$ phân biệt.

Khi đó, với hệ phương trình bậc thang , ta chứng minh được $l \le n$

Với hệ có dạng bậc thang ta xét các trường hợp sau:

TH1: $l < n$

Nếu tồn tại i thỏa $\begin{cases} k_i \ge n \\ b_i \ne 0 \end{cases}$ => Hệ phương trình vô nghiệm

Ngược lại, nếu $\begin{cases} k_i \ge n \\ b_i = 0 \end{cases}$ với mọi i thì hệ phương trình có vô số nghiệm

TH2: $l = n$

Nếu tồn tại i thỏa $k_i \ge n$, ta xét b_i:

- Nếu $b_i \ne 0$ => Hệ phương trình vô nghiệm
- Nếu $b_i = 0$ với mọi i thỏa $k_i \ge n$ thì phương trình có duy nhất 1 nghiệm (Hệ phương trình bậc thang có n ẩn và n phương trình có chứa ẩn có hệ số khác 0)

Nếu $k_i < n$ với mọi i thì phương trình có duy nhất 1 nghiệm (Hệ phương trình bậc thang có n ẩn và n phương trình có chứa ẩn có hệ số khác 0)

Câu 2:

Khi nhân ma trận A với:

- Ma trận $M(i;\lambda)$: Ta được một ma trận mới cùng kích thước có cá phần tử ở dòng i bị nhân gấp λ lần

- Ma trận $E(i;j)$: Ta nhận được một ma trận mới cùng kích thước có các phần tử ở dòng i đổi chỗ với các phần tử ở dòng j theo quy tắc A_{ik} đổi chỗ với A_{jk} với $k \in \{1;2;...;n\}$
- Ma trận $S(i,j;\lambda)$: Ta nhận được một ma trận mới cùng kích thước có các phần tử ở dòng A_{ik} bị biến đổi theo quy tắc $A_{ik} = A_{ik} + \lambda A_{jk}$ với $k \in \{1;2;...;n\}$. Hay có thể hiểu ta cộng các phần tử ở dòng i với λ lần các phần tử ở các cột tương ứng trên dòng j.

Khi nhân ma trận b với:

- Ma trận $M(i;\lambda)$: Ta được một ma trận mới cùng kích thước có cá phần tử ở dòng i bị nhân gấp λ lần
- Ma trận $E(i;j)$: Ta nhận được một ma trận mới cùng kích thước có phần tử ở dòng i đổi chỗ với phần tử ở dòng j hay có thể nói b_{i1} đổi chỗ với b_{j1}
- Ma trận $S(i,j;\lambda)$: Ta nhận được một ma trận mới cùng kích thước có phần tử ở dòng b_{i1} bị biến đổi theo quy tắc $b_{i1} = b_{i1} + \lambda b_{j1}$. Hay có thể hiểu ta cộng phần tử ở dòng i thêm với λ lần các phần tử ở trên dòng j.

Câu 3:

Như đã giải thích ở câu 2, việc nhân các ma trận sơ cấp vào 2 ma trận A và b chỉ tương tự với việc:

- Nhân một số λ $(\lambda \neq 0)$ vào 2 vế của một phương trình: $M(i;\lambda)$
- Đổi chỗ 2 phương trình trong hệ phương trình: $E(i;j)$
- Cộng một một phương trình với λ $(\lambda \neq 0)$ lần phương trình khác: $S(i,j;\lambda)$

Do đó, tập nghiệm của hệ phương trình $Ax = b$ và $TAx = Tb$ luôn trùng nhau với mọi ma trận sơ cấp $T \in R^{mxm}$

Câu 4:

1) Link code GitHub: Phần IV Câu 4
   Code được dán ở trang 7

CODE OUTPUT

```
T^(1):= S(2,1,-2.00)
T^(2):= S(3,1,-0.50)
T^(3):= E(3,2)

A^(4) =
[[ 1.          1.5         0.5        ]
 [ 0.          1.          0.14285714]
 [-0.         -0.          1.         ]]

b^(4) =
[ 5.5
  2.4285714285714284
  3.0 ]

Bộ nghiệm của hệ là: [1.0, 2.0, 3.0]
```

2) Thuật toán:
   - Ta tìm hệ số k của từng phương trình trong hệ phương trình đã cho
   - Ta tạo một list bỏ trống
   - Tạo thêm 2 ma trận A0=A và B0=B
   - Ta thực hiện vòng lặp sau n-1 lần:
     - Tìm phương trình i không nằm trong list bỏ có hệ số k nhỏ nhất và bé hơn n (số nghiệm của hệ phương trình)
     - Dùng phép khử Gauss để khử những hệ số ở cột k của các phương trình còn lại của ma trận A
     - Ở phép khử này, mỗi lần ta khử một phương trình t ta sẽ in ra môt ma trận trung gian S(t;j;lamda) với lamda là số đối của tỉ lệ giữa hệ số ở hàng t cột k với hệ số ở hàng i cột k.
     - Ta điều chỉnh hệ số ở mỗi hàng của ma trận b tương ứng
     - Thêm i vào list bỏ
     - Tính lại hệ số k của từng phương trình trừ những phương trình trong list bỏ
     - Ta cho A0=A,B0=B
     - Tiếp tục vòng lặp

- Ta nhận được một hệ phương trình với các phương trình trong nó có hệ số k đôi một khác nhau ngoại trừ những phương trình không chứa ẩn (hệ số n ẩn trong phương trình đó đều bằng 0)
- Ta tính lại các hệ số k và l của hệ phương trình mới này
- Ta tạo một list bỏ trống
- Ta xem những hàng ta chưa sắp xếp sẽ được gọi là hàng trống hay là chưa nằm trong list bỏ
- Tiếp tục sắp xếp lại hệ phương trình này thông qua vòng lặp:
  - Tìm phương trình i không nằm trong list bỏ có hệ số k nhỏ nhất và bé hơn n (số nghiệm của hệ phương trình)
  - Gọi hàng trống trên cùng ma trận là (hàng j)
  - Nếu i khác j:
  - Khi này ta in ra một ma trận trung gian M(i;j) để thể hiện việc đổi chỗ hàng i với hàng j (hàng trống trên cùng của ma trận)
  - Ta đổi chỗ các phần tử ở hàng i và hàng j ở các cột tương ứng trong cả ma trận A và b
  - Thêm j vào list bỏ
- Ta thu được một hệ phương trình có dạng bậc thang sau vòng lặp

Câu 5:

1) Link code GitHub: Phần IV Câu 5
   Code được dán ở trang 10
   CODE OUTPUT

```
Những bộ nghiệm của hệ là: [1.0, 2.0, 3.0]
```

2) Thuật toán

- Thực chất câu 4 được làm dựa trên câu 5 nên thuật toán của 2 câu gần như tương tự chỉ khác ở câu 5, ta không in ra các ma trận trung gian trong quá trình giải mà chỉ in ra tập nghiệm của hệ phương trình.
  - Ta tìm hệ số k của từng phương trình trong hệ phương trình đã cho
  - Ta tạo một list bỏ trống
  - Tạo thêm 2 ma trận A0=A và B0=B
  - Ta thực hiện vòng lặp sau n-1 lần:
    - Tìm phương trình i không nằm trong list bỏ có hệ số k nhỏ nhất và bé hơn n (số nghiệm của hệ phương trình)

- Dùng phép khử Gauss để khử những hệ số ở cột k của các phương trình còn lại của ma trận A
- Ở phép khử này, mỗi lần ta khử một phương trình t ta sẽ cộng mỗi phần tử ở hàng t với lamda lần phần tử ở hàng j ở cột tương ứng (lamda là số đối của tỉ lệ giữa hệ số ở hàng t cột k với hệ số ở hàng i cột k).
- Ta điều chỉnh hệ số ở mối hằng của ma trận A và b tương ứng
- Thêm i vào list bỏ
- Tính lại hệ số k của từng phương trình trừ những phương trình trong list bỏ
- Ta cho A0=A, B0=B
- Tiếp tục vòng lặp

o Ta nhận được một hệ phương trình với các phương trình trong nó có hệ số k đôi một khác nhau ngoại trừ những phương trình không chứa ẩn (hệ số n ẩn trong phương trình đó đều bằng 0)
o Ta tính lại các hệ số k và l của hệ phương trình mới này
o Ta tạo một list bỏ trống
o Ta xem những hàng ta chưa sắp xếp sẽ được gọi là hàng trống
o Tiếp tục sắp xếp lại hệ phương trình này thông qua vòng lặp:
- Tìm phương trình i không nằm trong list bỏ có hệ số k nhỏ nhất và bé hơn n (số nghiệm của hệ phương trình)
- Gọi hàng trống trên cùng ma trận là (hàng j)
- Nếu i khác j:
- Khi này ta in ra một ma trận trung gian M(i;j) để thể hiện việc đổi chỗ hàng i với hàng j (hàng trống trên cùng của ma trận)
- Ta đổi chỗ các phần tử ở hàng i và hàng j ở các cột tương ứng trong cả ma trận A và b
- Thêm j vào list bỏ

o Ta thu được một hệ phương trình có dạng bậc thang sau vòng lặp
o Ta thu được một hệ phương trình có dạng bậc thang sau vòng lặp
o Ta tính lại các hệ số k và l của hệ phương trình mới này
o Ta tiếp tục biện luận nghiệm và giải như đã giải thích ở câu 1:
- a_{ij} với $1 \le i \le n, 1 \le j \le n$ và $b_1, b_2, ..., b_m$ lần lượt là các hệ số của các ẩn và kết quả(giá trị ở vế phải hay hệ số tự do) sau khi hệ phương trình được đưa về dạng bậc thang.

- Gọi l là số phương trình trong hệ phương trình bậc thang có hệ số $k_i < n$
- Mặt khác, dựa vào điều kiện của hệ phương trình bậc thang, ta nhận thấy rằng ta chỉ nhận được tối đa n số nguyên $k_i < n$ phân biệt.
- Khi đó, với hệ phương trình bậc thang , ta chứng minh được $l \le n$

Với hệ có dạng bậc thang ta xét các trường hợp sau:

TH1: $l < n$

Nếu tồn tại i thỏa $\begin{cases} k_i \ge n \\ b_i \ne 0 \end{cases}$ => Hệ phương trình vô nghiệm

Ngược lại, nếu $\begin{cases} k_i \ge n \\ b_i = 0 \end{cases}$ với mọi i thì hệ phương trình có vô số nghiệm

TH2: $l = n$

Nếu tồn tại i thỏa $k_i \ge n$, ta xét b_i:

- Nếu $b_i \ne 0$ => Hệ phương trình vô nghiệm
- Nếu $b_i = 0$ với mọi i thỏa $k_i \ge n$ thì phương trình có duy nhất 1 nghiệm (Hệ phương trình bậc thang có n ẩn và n phương trình có chứa ẩn có hệ số khác 0)

Nếu $k_i < n$ với mọi i thì phương trình có duy nhất 1 nghiệm (Hệ phương trình bậc thang có n ẩn và n phương trình có chứa ẩn có hệ số khác 0)

TH3: $l > n$

Khi đó ta có được l giá trị $k_i < n$ phân biệt (Vô Lý)

Câu 4:

CODE

```
import numpy as np

def matrix(A,b):
    #Input
    A = [[float(item) for item in row] for row in A]
    for i in range(len(b)):
        b[i]=float(b[i])
    A = np.array(A)
    A0 = np.array(A)
    b = np.array(b)
    b0 = np.array(b)
    m,n=A.shape
    p=len(b)
    dk=[]
    for i in range(m):
        dk.append(0)
    for i in range(m):
         for j in range(n):
            if A[i,j] == 0:
                dk[i]+=1
            else:
                break
    min=dk[0]
    o=0
    bo=[]
    thai=False
    c=0
    count=0
    # Phép khử Gauss
    for w in range(n-1):
        for i in range(m):
            if (i in bo)==False:
                if min> dk[i]:
                    min=dk[i]
                    o=i
        bo.append(o)
        for j in range(m):
            for k in range(n):
                if j in bo:
                    break
                else:
                    thai=True
```

```
                A[j,k]-= (A0[o,k])*( (A0[j,dk[o]])/ (A0[o,dk[o]]) )
                c=-1*(A0[j,dk[o]])/ (A0[o,dk[o]])
        if thai==True:
            b[j]-=b0[o]*( A0[j,dk[o]]/ A0[o,dk[o]] )
            thai=False
            if c!=0:

                count+=1
                print(f"T^({count:.0f}):= S({j+1:.0f},{o+1:.0f},{c:.2f})")
    min=n
    for r in range(m):
        dk[r]=0
    for t in range(m):
        for y in range(n):
            if A[t,y] == 0:
                dk[t]+=1
            else:
                break
    for i in range(m):
        for j in range(n):
            A0[i,j]=A[i,j]
            b0[i]=b[i]
for r in range(m):
        dk[r]=0
for t in range(m):
    for y in range(n):
        if A[t,y] == 0:
            dk[t]+=1
        else:
            break
for i in range(m):
    for j in range(n):
        A[i,j]/=A0[i,dk[i]]
    b[i]/=A0[i,dk[i]]

A0=A
b0=b
i=0
l=0
k=0
j=0
t=0
y=0
r=0
bo=[]
```

```
    min=dk[0]
    o=0
    w=0

    for w in range(m):
        for i in range(m):
            if (i in bo)==False:
                if min> dk[i]:
                    min=dk[i]
                    o=i

            bo.append(o)
    # Sắp xếp hệ phương trình theo đúng dạng hệ phương trình bậc thang
        if o != w:
            count+=1
            print(f"T^({count:.0f}):= E({o+1:.0f},{w+1:.0f})")

            for k in range(n):
                va=A[o,k]
                A[o,k]=A[w,k]
                A[w,k]=va
                va=b[o]
                b[o]=b[w]
                b[w]=va
        min=n
        bo.append(w)
    # In ra he phuong trinh bac thang duoi dang hai ma tran A^(k)
    print()
    print(f"A^({count+1:.0f}) =")
    print(A)
    print()
    print(f"b^({count+1:.0f}) =")
    print("[ ", end="")
    for i in range(len(b)):
        if i == len(b)-1:
            print(" ",b[i],"]")
        elif i == 0:
            print(b[i])
        elif 1==1:
            print(" ",b[i])
    print()
    # Tinh lai he so k va l cua he phuong trinh
    for i in range(m):
        if dk[i]<n:
            l+=1
```

```
    x=[]
    dk=[]
    for r in range(m):
        dk.append(0)
    for t in range(m):
        for y in range(n):
            if A[t,y] == 0:
                dk[t]+=1
            else:
                break
    for i in range(n):
        x.append(0)
# Biện luận nghiệm
    if l<n:
        for i in range(m):
            if dk[i]>=n and b[i]!=0:
                print('He phuong trinh vo nghiem')
            if dk[i]>=n and b[i]==0:
                print('He phuong trinh co vo so nghiem')
    if l<n:

        for i in range(m):
            if dk[i]>=n and b[i]!=0:
                print('He phuong trinh vo nghiem')
            if dk[i]>=n and b[i]==0:
                print('He phuong trinh co vo so nghiem')
    if l==n:
        for i in range(m):
            if dk[i]>=n and b[i]!=0:
                print('He phuong trinh vo nghiem')
                exit()
        for j in range(n-1,-1,-1):
            pos=dk.index(j)
            # print(pos)
            for l in range(n):
                b[pos]-=A[pos,l]*x[l]
            x[j]=round(b[pos],2)
        print('Bộ nghiệm của hệ là:',x)
    if l>n:
        print('Day khong phai he phuong trinh bac thang')
    exit()

print(matrix([[2,3,1],[4,6,1],[1,5,1]],[11,19,14]))
```

OUTPUT

```
T^(1):= S(2,1,-2.00)
T^(2):= S(3,1,-0.50)
T^(3):= E(3,2)

A^(4) =
[[ 1.          1.5         0.5        ]
 [ 0.          1.          0.14285714]
 [-0.         -0.          1.         ]]

b^(4) =
[ 5.5
  2.4285714285714284
  3.0 ]

Bộ nghiệm của hệ là: [1.0, 2.0, 3.0]
```

Câu 5:

CODE

```
import numpy as np

def matrix(A,b):
    A = [[float(item) for item in row] for row in A]
    for i in range(len(b)):
        b[i]=float(b[i])
    A = np.array(A)
    A0 = np.array(A)
    b = np.array(b)
    b0 = np.array(b)
    m,n=A.shape
    p=len(b)
    dk=[]

    for i in range(m):
        dk.append(0)
    for i in range(m):
         for j in range(n):
            if A[i,j] == 0:
                dk[i]+=1
            else:
                break
    min=dk[0]
    o=0
    bo=[]
```

```
    thai=False
    #Đưa hệ phương trình về dạng bậc thang bằng phép khử Gauss
    for w in range(n-1):
        for i in range(m):
            if (i in bo)==False:
                if min> dk[i]:
                    min=dk[i]
                    o=i
        bo.append(o)
        for j in range(m):
            for k in range(n):
                if j in bo:
                    break
                else:

                    thai=True
                    A[j,k]-= (A0[o,k])*( (A0[j,dk[o]])/ (A0[o,dk[o]]) )

            if thai==True:
                b[j]-=b0[o]*( A0[j,dk[o]]/ A0[o,dk[o]] )
                thai=False

        min=n
        for r in range(m):
            dk[r]=0
        for t in range(m):
            for y in range(n):
                if A[t,y] == 0:
                    dk[t]+=1
                else:
                    break
        for i in range(m):
            for j in range(n):
                A0[i,j]=A[i,j]
                b0[i]=b[i]
    for r in range(m):
            dk[r]=0
    for t in range(m):
        for y in range(n):
            if A[t,y] == 0:
                dk[t]+=1
            else:
                break
    for i in range(m):
        for j in range(n):
```

```
            A[i,j]/=A0[i,dk[i]]
        b[i]/=A0[i,dk[i]]
    A0=A
    b0=b
    i=0
    l=0
    k=0
    j=0
    t=0
    y=0
    r=0
    for i in range(m):
        if dk[i]<n:
            l+=1
    x=[]
    for i in range(n):
        x.append(0)
    #Giải hệ phương trình
    if l<n:
        for i in range(m):
            if dk[i]>=n and b[i]!=0:
                print('He phuong trinh vo nghiem')
            if dk[i]>=n and b[i]==0:
                print('He phuong trinh co vo so nghiem')
    if l==n:
        for i in range(m):
            if dk[i]>=n and b[i]!=0:
                print('He phuong trinh vo nghiem')
                exit()
        for j in range(n-1,-1,-1):
            pos=dk.index(j)
            # print(pos)
            for l in range(n):
                b[pos]-=A[pos,l]*x[l]
            x[j]=round(b[pos],2)
        print('Bộ nghiệm của hệ là:',x)
    if l>n:
        print('Day khong phai he phuong trinh bac thang')
    exit()

print(matrix([[2,3,1],[4,6,1],[1,5,1]],[11,19,14]))
```

OUTPUT

```
Bộ nghiệm của hệ là: [1.0, 2.0, 3.0]
```

REFERENCE

[1] Idris, I. (2015). NumPy: Beginner's Guide. Packt Publishing Ltd.

[2] Higham, N. J. (2011). Gaussian elimination. Wiley Interdisciplinary Reviews: Computational Statistics, 3(3), 230-238.

[3] Prince, E., & Prince, E. (2004). Matrices: Definitions and Fundamental Operations. *Mathematical Techniques in Chrystallography and Materials Science*, 1-19.

Phần IV

Câu 1:

a_{ij} với $1 \le i \le n, 1 \le j \le n$ và $b_1, b_2, ..., b_m$ lần lượt là các hệ số của các ẩn và kết quả (giá trị ở vế phải hay hệ số tự do) sau khi hệ phương trình được đưa về dạng bậc thang.

Gọi l là số phương trình trong hệ phương trình bậc thang có hệ số $k_i < n$

Mặt khác, dựa vào điều kiện của hệ phương trình bậc thang, ta nhận thấy rằng ta chỉ nhận được tối đa n số nguyên $k_i < n$ phân biệt.

Khi đó, với hệ phương trình bậc thang , ta chứng minh được $l \le n$

Với hệ có dạng bậc thang ta xét các trường hợp sau:

TH1: $l < n$

Nếu tồn tại i thỏa $\begin{cases} k_i \ge n \\ b_i \ne 0 \end{cases}$ => Hệ phương trình vô nghiệm

Ngược lại, nếu $\begin{cases} k_i \ge n \\ b_i = 0 \end{cases}$ với mọi i thì hệ phương trình có vô số nghiệm

TH2: $l = n$

Nếu tồn tại i thỏa $k_i \geq n$, ta xét b_i:

- Nếu $b_i \neq 0$ => Hệ phương trình vô nghiệm
- Nếu $b_i = 0$ với mọi i thỏa $k_i \geq n$ thì phương trình có duy nhất 1 nghiệm (Hệ phương trình bậc thang có n ẩn và n phương trình có chứa ẩn có hệ số khác 0)

Nếu $k_i < n$ với mọi i thì phương trình có duy nhất 1 nghiệm (Hệ phương trình bậc thang có n ẩn và n phương trình có chứa ẩn có hệ số khác 0)

Câu 2:

Khi nhân ma trận A với:

- Ma trận $M(i;\lambda)$: Ta được một ma trận mới cùng kích thước có cá phần tử ở dòng i bị nhân gấp λ lần
- Ma trận $E(i;j)$: Ta nhận được một ma trận mới cùng kích thước có các phần tử ở dòng i đổi chỗ với các phần tử ở dòng j theo quy tắc A_{ik} đổi chỗ với A_{jk} với $k \in \{1;2;...;n\}$
- Ma trận $S(i,j;\lambda)$: Ta nhận được một ma trận mới cùng kích thước có các phần tử ở dòng A_{ik} bị biến đổi theo quy tắc $A_{ik} = A_{ik} + \lambda A_{jk}$ với $k \in \{1;2;...;n\}$. Hay có thể hiểu ta cộng các phần tử ở dòng i với λ lần các phần tử ở các cột tương ứng trên dòng j.

Khi nhân ma trận b với:

- Ma trận $M(i;\lambda)$: Ta được một ma trận mới cùng kích thước có cá phần tử ở dòng i bị nhân gấp λ lần
- Ma trận $E(i;j)$: Ta nhận được một ma trận mới cùng kích thước có phần tử ở dòng i đổi chỗ với phần tử ở dòng j hay có thể nói b_{i1} đổi chỗ với b_{j1}
- Ma trận $S(i,j;\lambda)$: Ta nhận được một ma trận mới cùng kích thước có phần tử ở dòng b_{i1} bị biến đổi theo quy tắc $b_{i1} = b_{i1} + \lambda b_{j1}$. Hay có thể hiểu ta cộng phần tử ở dòng i thêm với λ lần các phần tử ở trên dòng j.

Câu 3:

Như đã giải thích ở câu 2, việc nhân các ma trận sơ cấp vào 2 ma trận A và b chỉ tương tự với việc:

- Nhân một số λ $(\lambda \neq 0)$ vào 2 vế của một phương trình: $M(i;\lambda)$
- Đổi chỗ 2 phương trình trong hệ phương trình: $E(i;j)$
- Cộng một một phương trình với λ $(\lambda \neq 0)$ lần phương trình khác: $S(i,j;\lambda)$

Do đó, tập nghiệm của hệ phương trình $Ax = b$ và $TAx = Tb$ luôn trùng nhau với mọi ma trận sơ cấp $T \in R^{mxm}$

Câu 4:

3) Link code GitHub: Phần IV Câu 4
Code được dán ở trang 7

CODE OUTPUT

```
T^(1):= S(2,1,-2.00)
T^(2):= S(3,1,-0.50)
T^(3):= E(3,2)

A^(4) =
[[ 1.          1.5         0.5        ]
 [ 0.          1.          0.14285714]
 [-0.         -0.          1.         ]]

b^(4) =
[ 5.5
  2.4285714285714284
  3.0 ]

Bộ nghiệm của hệ là: [1.0, 2.0, 3.0]
```

4) Thuật toán:
   - Ta tìm hệ số k của từng phương trình trong hệ phương trình đã cho
   - Ta tạo một list bỏ trống
   - Tạo thêm 2 ma trận A0=A và B0=B
   - Ta thực hiện vòng lặp sau n-1 lần:

        - Tìm phương trình i không nằm trong list bỏ có hệ số k nhỏ nhất và bé hơn n (số nghiệm của hệ phương trình)
        - Dùng phép khử Gauss để khử những hệ số ở cột k của các phương trình còn lại của ma trận A
        - Ở phép khử này, mỗi lần ta khử một phương trình t ta sẽ in ra môt ma trận trung gian S(t;j;lamda) với lamda là số đối của tỉ lệ giữa hệ số ở hàng t cột k với hệ số ở hàng i cột k.
        - Ta điều chỉnh hệ số ở mỗi hàng của ma trận b tương ứng
        - Thêm i vào list bỏ
        - Tính lại hệ số k của từng phương trình trừ những phương trình trong list bỏ
        - Ta cho A0=A,B0=B
        - Tiếp tục vòng lặp
    - Ta nhận được một hệ phương trình với các phương trình trong nó có hệ số k đôi một khác nhau ngoại trừ những phương trình không chứa ẩn (hệ số n ẩn trong phương trình đó đều bằng 0)
    - Ta tính lại các hệ số k và l của hệ phương trình mới này
    - Ta tạo một list bỏ trống
    - Ta xem những hàng ta chưa sắp xếp sẽ được gọi là hàng trống hay là chưa nằm trong list bỏ
    - Tiếp tục sắp xếp lại hệ phương trình này thông qua vòng lặp:
        - Tìm phương trình i không nằm trong list bỏ có hệ số k nhỏ nhất và bé hơn n (số nghiệm của hệ phương trình)
        - Gọi hàng trống trên cùng ma trận là (hàng j)
        - Nếu i khác j:
        - Khi này ta in ra một ma trận trung gian M(i;j) để thể hiện việc đổi chỗ hàng i với hàng j (hàng trống trên cùng của ma trận)
        - Ta đổi chỗ các phần tử ở hàng i và hàng j ở các cột tương ứng trong cả ma trận A và b
        - Thêm j vào list bỏ
    - Ta thu được một hệ phương trình có dạng bậc thang sau vòng lặp

Câu 5:

3) Link code GitHub: Phần IV Câu 5
   Code được dán ở trang 10
   CODE OUTPUT

```
Những bộ nghiệm của hệ là: [1.0, 2.0, 3.0]
```

4) Thuật toán

- Thực chất câu 4 được làm dựa trên câu 5 nên thuật toán của 2 câu gần như tương tự chỉ khác ở câu 5, ta không in ra các ma trận trung gian trong quá trình giải mà chỉ in ra tập nghiệm của hệ phương trình.
    - Ta tìm hệ số k của từng phương trình trong hệ phương trình đã cho
    - Ta tạo một list bỏ trống
    - Tạo thêm 2 ma trận A0=A và B0=B
    - Ta thực hiện vòng lặp sau n-1 lần:
        - Tìm phương trình i không nằm trong list bỏ có hệ số k nhỏ nhất và bé hơn n (số nghiệm của hệ phương trình)
        - Dùng phép khử Gauss để khử những hệ số ở cột k của các phương trình còn lại của ma trận A
        - Ở phép khử này, mỗi lần ta khử một phương trình t ta sẽ cộng mỗi phần tử ở hàng t với lamda lần phần tử ở hàng j ở cột tương ứng (lamda là số đối của tỉ lệ giữa hệ số ở hàng t cột k với hệ số ở hàng i cột k).
        - Ta điều chỉnh hệ số ở mối hằng của ma trận A và b tương ứng
        - Thêm i vào list bỏ
        - Tính lại hệ số k của từng phương trình trừ những phương trình trong list bỏ
        - Ta cho A0=A, B0=B
        - Tiếp tục vòng lặp
    - Ta nhận được một hệ phương trình với các phương trình trong nó có hệ số k đôi một khác nhau ngoại trừ những phương trình không chứa ẩn (hệ số n ẩn trong phương trình đó đều bằng 0)
    - Ta tính lại các hệ số k và l của hệ phương trình mới này
    - Ta tạo một list bỏ trống
    - Ta xem những hàng ta chưa sắp xếp sẽ được gọi là hàng trống
    - Tiếp tục sắp xếp lại hệ phương trình này thông qua vòng lặp:
        - Tìm phương trình i không nằm trong list bỏ có hệ số k nhỏ nhất và bé hơn n (số nghiệm của hệ phương trình)
        - Gọi hàng trống trên cùng ma trận là (hàng j)
        - Nếu i khác j:
        - Khi này ta in ra một ma trận trung gian M(i;j) để thể hiện việc đổi chỗ hàng i với hàng j (hàng trống trên cùng của ma trận)

- Ta đổi chỗ các phần tử ở hàng i và hàng j ở các cột tương ứng trong cả ma trận A và b
- Thêm j vào list bỏ

- Ta thu được một hệ phương trình có dạng bậc thang sau vòng lặp
- Ta thu được một hệ phương trình có dạng bậc thang sau vòng lặp
- Ta tính lại các hệ số k và l của hệ phương trình mới này
- Ta tiếp tục biện luận nghiệm và giải như đã giải thích ở câu 1:
  - a_{ij} với $1 \le i \le n, 1 \le j \le n$ và $b_1, b_2, ..., b_m$ lần lượt là các hệ số của các ẩn và kết quả(giá trị ở vế phải hay hệ số tự do) sau khi hệ phương trình được đưa về dạng bậc thang.
  - Gọi l là số phương trình trong hệ phương trình bậc thang có hệ số $k_i < n$
  - Mặt khác, dựa vào điều kiện của hệ phương trình bậc thang, ta nhận thấy rằng ta chỉ nhận được tối đa n số nguyên $k_i < n$ phân biệt.
  - Khi đó, với hệ phương trình bậc thang , ta chứng minh được $l \le n$

Với hệ có dạng bậc thang ta xét các trường hợp sau:

TH1: $l < n$

Nếu tồn tại i thỏa $\begin{cases} k_i \ge n \\ b_i \ne 0 \end{cases}$ => Hệ phương trình vô nghiệm

Ngược lại, nếu $\begin{cases} k_i \ge n \\ b_i = 0 \end{cases}$ với mọi i thì hệ phương trình có vô số nghiệm

TH2: $l = n$

Nếu tồn tại i thỏa $k_i \ge n$, ta xét b_i:

- Nếu $b_i \ne 0$ => Hệ phương trình vô nghiệm
- Nếu $b_i = 0$ với mọi i thỏa $k_i \ge n$ thì phương trình có duy nhất 1 nghiệm (Hệ phương trình bậc thang có n ẩn và n phương trình có chứa ẩn có hệ số khác 0)

Nếu $k_i < n$ với mọi i thì phương trình có duy nhất 1 nghiệm (Hệ phương trình bậc thang có n ẩn và n phương trình có chứa ẩn có hệ số khác 0)

TH3: $l > n$

Khi đó ta có được l giá trị $k_i < n$ phân biệt (Vô Lý)

Câu 4:

CODE

```
import numpy as np

def matrix(A,b):
    #Input
    A = [[float(item) for item in row] for row in A]
    for i in range(len(b)):
        b[i]=float(b[i])
    A = np.array(A)
    A0 = np.array(A)
    b = np.array(b)
    b0 = np.array(b)
    m,n=A.shape
    p=len(b)
    dk=[]
    for i in range(m):
        dk.append(0)
    for i in range(m):
         for j in range(n):
            if A[i,j] == 0:
                dk[i]+=1
            else:
                break
    min=dk[0]
    o=0
    bo=[]
    thai=False
    c=0
    count=0
    # Phép khử Gauss
    for w in range(n-1):
        for i in range(m):
```

```
            if (i in bo)==False:
                if min> dk[i]:
                    min=dk[i]
                    o=i
        bo.append(o)
        for j in range(m):
            for k in range(n):
                if j in bo:
                    break
                else:
                    thai=True
                    A[j,k]-= (A0[o,k])*( (A0[j,dk[o]])/ (A0[o,dk[o]]) )
                    c=-1*(A0[j,dk[o]])/ (A0[o,dk[o]])
            if thai==True:
                b[j]-=b0[o]*( A0[j,dk[o]]/ A0[o,dk[o]] )
                thai=False
                if c!=0:

                    count+=1
                    print(f"T^({count:.0f}):= S({j+1:.0f},{o+1:.0f},{c:.2f})")
        min=n
        for r in range(m):
            dk[r]=0
        for t in range(m):
            for y in range(n):
                if A[t,y] == 0:
                    dk[t]+=1
                else:
                    break
        for i in range(m):
            for j in range(n):
                A0[i,j]=A[i,j]
                b0[i]=b[i]
    for r in range(m):
            dk[r]=0
    for t in range(m):
        for y in range(n):
            if A[t,y] == 0:
                dk[t]+=1
            else:
                break
    for i in range(m):
        for j in range(n):
            A[i,j]/=A0[i,dk[i]]
        b[i]/=A0[i,dk[i]]
```

```
A0=A
b0=b
i=0
l=0
k=0
j=0
t=0
y=0
r=0
bo=[]
min=dk[0]
o=0
w=0

for w in range(m):
    for i in range(m):
        if (i in bo)==False:
            if min> dk[i]:
                min=dk[i]
                o=i

        bo.append(o)
# Sắp xếp hệ phương trình theo đúng dạng hệ phương trình bậc thang
    if o != w:
        count+=1
        print(f"T^({count:.0f}):= E({o+1:.0f},{w+1:.0f})")

        for k in range(n):
            va=A[o,k]
            A[o,k]=A[w,k]
            A[w,k]=va
            va=b[o]
            b[o]=b[w]
            b[w]=va
    min=n
    bo.append(w)
# In ra he phuong trinh bac thang duoi dang hai ma tran A^(k)
print()
print(f"A^({count+1:.0f}) =")
print(A)
print()
print(f"b^({count+1:.0f}) =")
print("[ ", end="")
for i in range(len(b)):
```

```
        if i == len(b)-1:
            print(" ",b[i],"]")
        elif i == 0:
            print(b[i])
        elif 1==1:
            print(" ",b[i])
    print()
    # Tinh lai he so k va l cua he phuong trinh
    for i in range(m):
        if dk[i]<n:
            l+=1
    x=[]
    dk=[]
    for r in range(m):
        dk.append(0)
    for t in range(m):
        for y in range(n):
            if A[t,y] == 0:
                dk[t]+=1
            else:
                break
    for i in range(n):
        x.append(0)
# Biện luận nghiệm
    if l<n:
        for i in range(m):
            if dk[i]>=n and b[i]!=0:
                print('He phuong trinh vo nghiem')
            if dk[i]>=n and b[i]==0:
                print('He phuong trinh co vo so nghiem')
    if l<n:

        for i in range(m):
            if dk[i]>=n and b[i]!=0:
                print('He phuong trinh vo nghiem')
            if dk[i]>=n and b[i]==0:
                print('He phuong trinh co vo so nghiem')
    if l==n:
        for i in range(m):
            if dk[i]>=n and b[i]!=0:
                print('He phuong trinh vo nghiem')
                exit()
        for j in range(n-1,-1,-1):
            pos=dk.index(j)
            # print(pos)
```

```
            for l in range(n):
                b[pos]-=A[pos,l]*x[l]
            x[j]=round(b[pos],2)
        print('Bộ nghiệm của hệ là:',x)
    if l>n:
        print('Day khong phai he phuong trinh bac thang')
    exit()

print(matrix([[2,3,1],[4,6,1],[1,5,1]],[11,19,14]))
```

OUTPUT

```
T^(1):= S(2,1,-2.00)
T^(2):= S(3,1,-0.50)
T^(3):= E(3,2)

A^(4) =
[[ 1.          1.5         0.5        ]
 [ 0.          1.          0.14285714]
 [-0.         -0.          1.         ]]

b^(4) =
[ 5.5
  2.4285714285714284
  3.0 ]

Bộ nghiệm của hệ là: [1.0, 2.0, 3.0]
```

Câu 5:

CODE

```
import numpy as np

def matrix(A,b):
    A = [[float(item) for item in row] for row in A]
    for i in range(len(b)):
        b[i]=float(b[i])
    A = np.array(A)
    A0 = np.array(A)
    b = np.array(b)
    b0 = np.array(b)
    m,n=A.shape
    p=len(b)
    dk=[]

    for i in range(m):
        dk.append(0)
```

```
for i in range(m):
     for j in range(n):
        if A[i,j] == 0:
            dk[i]+=1
        else:
            break
min=dk[0]
o=0
bo=[]
thai=False
#Đưa hệ phương trình về dạng bậc thang bằng phép khử Gauss
for w in range(n-1):
    for i in range(m):
        if (i in bo)==False:
            if min> dk[i]:
                min=dk[i]
                o=i
    bo.append(o)
    for j in range(m):
        for k in range(n):
            if j in bo:
                break
            else:

                thai=True
                A[j,k]-= (A0[o,k])*( (A0[j,dk[o]])/ (A0[o,dk[o]]) )

        if thai==True:
            b[j]-=b0[o]*( A0[j,dk[o]]/ A0[o,dk[o]] )
            thai=False

    min=n
    for r in range(m):
        dk[r]=0
    for t in range(m):
        for y in range(n):
            if A[t,y] == 0:
                dk[t]+=1
            else:
                break
    for i in range(m):
        for j in range(n):
            A0[i,j]=A[i,j]
            b0[i]=b[i]
for r in range(m):
```

```
            dk[r]=0
    for t in range(m):
        for y in range(n):
            if A[t,y] == 0:
                dk[t]+=1
            else:
                break
    for i in range(m):
        for j in range(n):
            A[i,j]/=A0[i,dk[i]]
        b[i]/=A0[i,dk[i]]
    A0=A
    b0=b
    i=0
    l=0
    k=0
    j=0
    t=0
    y=0
    r=0
    for i in range(m):
        if dk[i]<n:
            l+=1
    x=[]
    for i in range(n):
        x.append(0)
    #Giải hệ phương trình
    if l<n:
        for i in range(m):
            if dk[i]>=n and b[i]!=0:
                print('He phuong trinh vo nghiem')
            if dk[i]>=n and b[i]==0:
                print('He phuong trinh co vo so nghiem')
    if l==n:
        for i in range(m):
            if dk[i]>=n and b[i]!=0:
                print('He phuong trinh vo nghiem')
                exit()
        for j in range(n-1,-1,-1):
            pos=dk.index(j)
            # print(pos)
            for l in range(n):
                b[pos]-=A[pos,l]*x[l]
            x[j]=round(b[pos],2)
        print('Bộ nghiệm của hệ là:',x)
```

```
    if l>n:
        print('Day khong phai he phuong trinh bac thang')
    exit()

print(matrix([[2,3,1],[4,6,1],[1,5,1]],[11,19,14]))
```

OUTPUT

```
 Bộ nghiệm của hệ là: [1.0, 2.0, 3.0]
```

REFERENCE

[1] Idris, I. (2015). NumPy: Beginner's Guide. Packt Publishing Ltd.

[2] Higham, N. J. (2011). Gaussian elimination. Wiley Interdisciplinary Reviews: Computational Statistics, 3(3), 230-238.

[3] Prince, E., & Prince, E. (2004). Matrices: Definitions and Fundamental Operations. *Mathematical Techniques in Chrystallography and Materials Science*, 1-19.

Phần V

Câu 1:

Như đã chứng minh ở phần 1, ta có các điều kiện rang buộc sau đối với s_1, s_2, r :

$$\begin{cases} r \le \dfrac{\sqrt{3}s_2 - s_1}{2} \\ r \le 18 - s_1 \\ r \le 18 - s_2 \\ r \le \dfrac{\sqrt{3}s_1 - s_2}{2} \\ r \ge 0 \end{cases}$$

Nhận thấy tứ giác ABCD là tứ giác ngoại tiếp đường tròn (vì có tổng 2 cặp cạnh đối bằng nhau)

Do đó, với một đường tròn $(S;r)$ nằm trong tứ giác ABCD, bán kính r đạt giá trị lớn nhất khi và chỉ khi $(S;r)$ là đường tròn nội tiếp tứ giác ABCD.

Khi đó khoảng cách từ điểm $S(s_1;s_2)$ tới các cạnh của tứ giác ABCD đều bằng nhau và bằng bán kính đế r_{max} lớn nhất có thể cắt được. Ta có được hệ sau:

$$\begin{cases} r_{max} = \dfrac{\sqrt{3}s_2 - s_1}{2} \\ 18 - s_2 = 18 - s_1 \\ \dfrac{\sqrt{3}s_2 - s_1}{2} = \dfrac{\sqrt{3}s_1 - s_2}{2} \\ 18 - s_1 = \dfrac{\sqrt{3}s_1 - s_2}{2} \\ r \ge 0 \end{cases}$$

Giải hệ phương trình sau ta được:

$$\begin{cases} r_{max} = 36 - 18\sqrt{3} \\ s_1 = -18 + 18\sqrt{3} \\ s_2 = -18 + 18\sqrt{3} \end{cases}$$

Vậy cái đế hình tròn có diện tích lớn nhất có thể cắt được từ tứ giác ABCD là khoảng $73{,}08\ cm^2$ (hay $\left(2268 - 1296\sqrt{3}\right)\pi\ cm^2$) với bán kính bằng $36 - 18\sqrt{3}\ cm$

Câu 2:

1) Link code Github: Phần V Câu 2

RAW CODE

```
import math
import sys
def intersectTwoLine(a1, b1, c1, a2, b2, c2):
  det = a1 * b2 - a2 * b1
  if det!=0 or ( det==0 and (a1==0 or a2==0 or b1==0 or b2==0)):
    x0 = (b2 * c1 - b1 * c2) / det
    y0 = (a1 * c2 - a2 * c1) / det
    return x0, y0
  else:
      return
def LPMax2Var (alpha , beta , constraint ):
    a=[]
    b=[]
    c=[]
    x=[]
    y=[]
    g=0
    x0=[]
    y0=[]
    # Trường hợp 1:
    if len(constraint)<=1:
        for i in constraint:
                a.append(i[0])
                b.append(i[1])
                c.append(i[2])
                g+=1
        t=float(0)
        g=float(0)
        if a[0]==0:
            g=c[0]/b[0]
        elif b[0]==0:
```

```
            t=c[0]/a[0]
        elif 1==1:
            g=c[0]/b[0]
        if a[0]*(t+a[0])+b[0]*(g+b[0])-c[0]<=0:
            if alpha*t+beta*g<(alpha*(t+a[0])+beta*(g+b[0])):
                print('Co vo han phuong an toi uu')
            else:
                print(alpha*t+beta*g)
        elif a[0]*(t-a[0])+b[0]*(g-b[0])-c[0]<=0:
            if alpha*t+beta*g<(alpha*(t-a[0])+beta*(g-b[0])):
                print('Co vo han phuong an toi uu')
            else:
                print(alpha*t+beta*g)
    #Trường hợp 2:
    if len(constraint)>=2:
        for i in constraint:
                a.append(i[0])
                b.append(i[1])
                c.append(i[2])
                g+=1
        for l in range(g):
            for j in range(l,g):
                if a[l]*b[j]-a[j]*b[l] != 0:
                    o= intersectTwoLine(a[l],b[l],c[l],a[j],b[j],c[j])
                    x0.append(o[0])
                    y0.append(o[1])
        j=0
        for k in range(len(x0)):
            for l in range(len(a)):
                if a[l]*x0[k]+b[l]*y0[k]-c[l]<=0:
                    j+=1
            if j== len(a):
                x.append(x0[k])
                y.append(y0[k])
            j=0

        if x==[]:
            print('INVALID')
            exit()
        sumdegrees=0
        #Tìm tổng số đo các góc ở đỉnh của miền nghiệm
        for q in range(len(x)):
            for j in range(len(constraint)):
                if a[j]*x[q]+b[j]*y[q]-c[j]==0:
                    for l in range(j+1,len(constraint)):
```

```
                        if a[l]*x[q]+b[l]*y[q]-c[l]==0:
                            if ((b[q]-b[j])**2+(a[q]-a[j])**2) != 0:
                                if a[l] * (x[q]-b[j]) + b[l] * (y[q]+a[j]) -
c[l]<=0 and a[j] * (x[q]-b[l]) + b[j] * (y[q]+a[l]) - c[j]<=0 :
                                    cos=(b[q]*b[j]+a[q]*a[j])/(math.sqrt((b[q]-
b[j])**2+(a[q]-a[j])**2))
                                    radian=math.acos(cos)
                                    degree=math.degrees(radian)
                                    sumdegrees+=degree

                                if a[l] * (x[q]+b[j]) + b[l] * (y[q]-a[j]) -
c[l]<=0 and a[j] * (x[q]-b[l]) + b[j] * (y[q]+a[l]) - c[j]<=0 :
                                    cos=(-b[q]*b[j]-a[q]*a[j])/(math.sqrt((b[q]-
b[j])**2+(a[q]-a[j])**2))
                                    radian=math.acos(cos)
                                    degree=math.degrees(radian)
                                    sumdegrees+=degree

                                if a[l] * (x[q]+b[j]) + b[l] * (y[q]-a[j]) -
c[l]<=0 and a[j] * (x[q]+b[l]) + b[j] * (y[q]-a[l]) - c[j]<=0 :
                                    cos=(b[q]*b[j]+a[q]*a[j])/(math.sqrt((b[q]-
b[j])**2+(a[q]-a[j])**2))
                                    radian=math.acos(cos)
                                    degree=math.degrees(radian)
                                    sumdegrees+=degree

                                if a[l] * (x[q]-b[j]) + b[l] * (y[q]+a[j]) -
c[l]<=0 and a[j] * (x[q]+b[l]) + b[j] * (y[q]-a[l]) - c[j]<=0 :
                                    cos=(-b[q]*b[j]-a[q]*a[j])/(math.sqrt((b[q]-
b[j])**2+(a[q]-a[j])**2))
                                    radian=math.acos(cos)
                                    degree=math.degrees(radian)
                                    sumdegrees+=degree
        # print(sumdegrees)
        # Dựa vào số đo để nhận xét miền nghiệm đóng hay mở
        if sumdegrees==((len(x)-2)*180):

            optimalVal=alpha*x[0]+beta*y[0]
            for q in range(len(x)):
                if alpha*x[q]+beta*y[q] > optimalVal:
                    optimalVal=alpha*x[q]+beta*y[q]
                    print('aaa')
            return optimalVal
        else:
            optimalVal=alpha*x[0]+beta*y[0]
```

```
            xmax=x[0]
            ymax=y[0]
            for q in range(len(x)):
                if alpha*x[q]+beta*y[q] > optimalVal:
                    optimalVal=alpha*x[q]+beta*y[q]
                    xmax=x[q]
                    ymax=y[q]
            g=0
            i=0
            q=0
            j=0
            l=0
            thai=True
            dk=[]
            for l in range(len(constraint)):
                dk.append(0)
            l=0
            for l in range(len(constraint)):
                for q in range(len(x)):
                    if a[l]*x[q]+b[l]*y[q]-c[l]==0:
                        dk[l]+=1
            l=0
            q=0
            # print(dk)
            #Xem hàm mục tiêu tiến ra vô cùng hay có giá trị tối ưu
            for l in range(len(constraint)):
                if dk[l]==1:
                    for i in range(len(x)):
                        if a[l]*x[i]+b[l]*y[i]-c[l]==0:
                            if (x[i]!=xmax and y[i]!=ymax and
alpha*x[i]+beta*y[i] != optimalVal):
                                print(optimalVal)
                                exit()
                            else:
                                for q in range(len(constraint)):
                                    if a[q]*(x[i]-b[l])+b[q]*(y[i]+a[l])-c[q]<=0
and alpha*(x[i]-b[l])+beta*(y[i]+a[l]) > optimalVal:
                                        optimalVal='INFINITE'
                                        return optimalVal
                                    elif (a[q]*(x[i]+b[l])+b[q]*(y[i]-a[l])-
c[q])<=0 and (alpha*(x[i]+b[l])+beta*(y[i]-a[l])) > optimalVal:
                                        optimalVal='INFINITE'
                                        return optimalVal
                                    elif 1==1:
                                        return optimalVal
```

```
print(LPMax2Var (6, 25, [[3, 5, 240] , [0, 1, 12], [-1, 0, 0], [0,-1, 0]]))
```

OUTPUT

```
Giá trị tối ưu: 660.0
```

2) Thuật Toán

- Thay tọa độ các giao điểm vào hệ bất phương trình (hàm điều kiện), nếu không có điểm nào thỏa thì in INVALID
- Nếu có đỉnh thỏa thì đếm số đỉnh thỏa rồi lưu vào n
- Nếu có ta tiếp tục tìm góc mở của vùng nghiệm giữa 2 cạnh kề nhau của miền nghiệm
- Nếu tổng các góc này bằng đúng tổng các góc trong một đa giác lồi n cạnh thì ta làm như bài toán ở phần III
- Nếu tổng các góc này khác tổng các góc trong đa giác n đỉnh hoặc n=1. Ta sẽ tìm hai cạnh biên (cạnh chỉ chứa đúng một điểm cực trị và là biên của miền nghiệm) của miền nghiệm. Điểm cực trị duy nhất trên đó ta sẽ gọi là điểm cực trị biên.
- Ta thay giá trị của tất cả các điểm cực trị thỏa vào hàm mục tiêu tìm max
- Nếu max nằm ở điểm không phải điểm cực trị biên thì in ra rồi DỪNG
- Nếu max nằm ở điểm cực trị biên, ta cộng hoặc trừ tọa độ của điểm cực trị biên với vecto chỉ phương của cạnh biên chứa nó, xong thay vào hàm điều kiện.
- Nếu thỏa ta tiếp tục thay vào hàm mục tiêu và so sánh với giá trị khi thay bằng tọa độ của điểm cực trị biên.
- Nếu kết quả là lớn hơn in INFINITE
- Nếu kết quả là bé hơn in giá trị của hàm mục tiêu khi thay tọa độ của điểm cực trị biên.

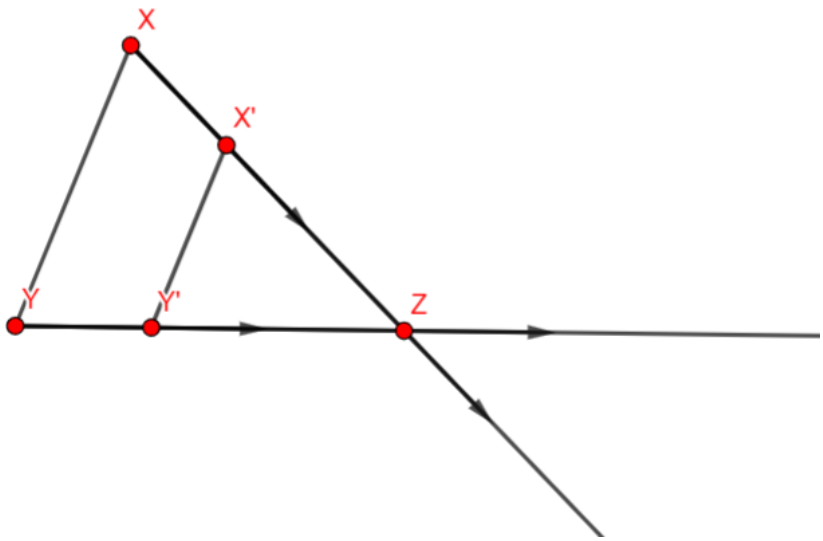
Câu 3:

Ta chứng minh bài toán nhỏ sau:

(1) Cho 2 người ở 2 điểm X,Y phân biệt, đi theo hai phương không song song với mỗi người có một vận tốc xác định, không đổi và gặp nhau tại Z sau một khoảng thời gian t_0.

Chứng minh rằng: đoạn thẳng nối hai người này luôn song song với một đường thẳng cố định sau một khoảng thời gian t bất kì $(t \neq t_0)$.

- Chứng minh:



Gọi X',Y' là vị trí hai người sau khi di chuyển một khoảng thời gian $t\left(t \neq t_0\right)$

Dựa vào hình vẽ, ta có thể chứng minh $\triangle ZYX \backsim \triangle ZY'X'$

Từ đó suy ra $YX \;//\; Y'X'$

Vậy 2 người này luôn di chuyển trên cùng một đường thẳng song song với đường thẳng nối hai người lúc xuất phát(cố định)

Áp dụng (1) ta tiếp tục chứng minh một bài toán nữa:

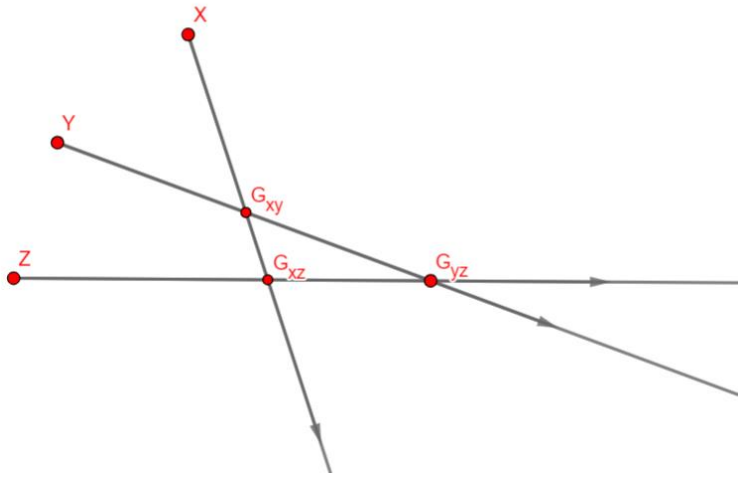
(2) Cho 3 người ở 3 điểm X,Y,Z phân biệt, đi theo ba phương đôi một không song song với nhau, mỗi người có một vận tốc xác định, không đổi. Họ đôi một gặp nhau tại G_{XY},G_{YZ},G_{ZX} sau một khoảng thời gian t_0.

Chứng minh rằng:

- Vị trí xuất phát của 3 người nằm trên cùng một đường thẳng
- Trong cùng một thời điểm, vị trí cả 3 người sau khi bắt đầu di chuyển luôn nằm trên một đường thẳng song song với một đường thẳng cố định sau một khoảng thời gian t bất kì $\left(t \neq t_0\right)$

- Chứng minh:

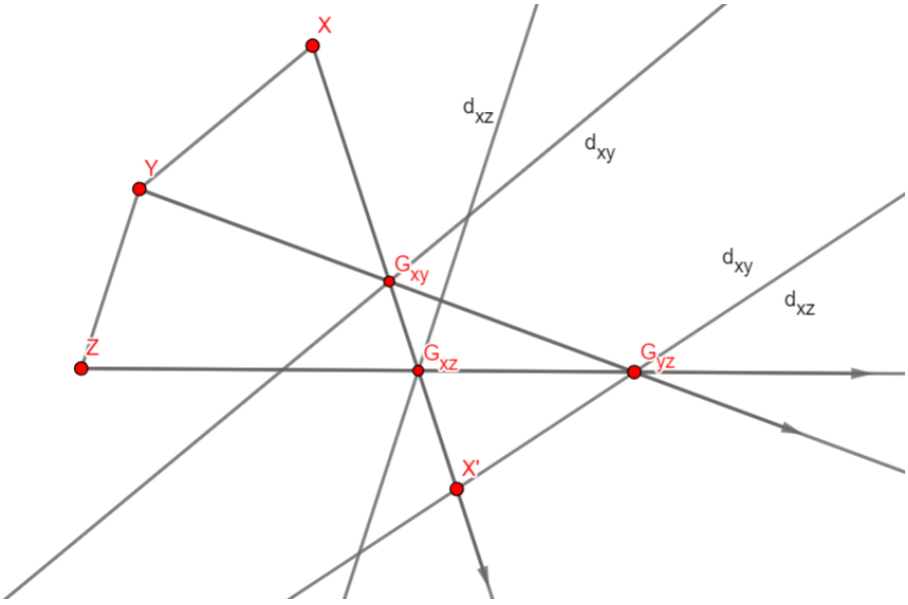
  Không mất tính tổng quát, ta giả sử thứ tự gặp nhau của 3 người như hình sau.



Gọi d_{XY}, d_{XZ} lần lượt là đường thẳng nối hai người X,Y và X,Z trong suốt quá trình di chuyển được biểu diễn như trên hình.

Ta nhận thấy khi Y và Z gặp nhau, Vị trí của Y,Z trùng nhau tại điểm G_{YZ}. Gọi X' là vị trí của X tại thời điểm đó

Vậy khi đó 2 đường thẳng d_{XY}, d_{XZ} cùng đi qua hai điểm X' và G_{YZ}



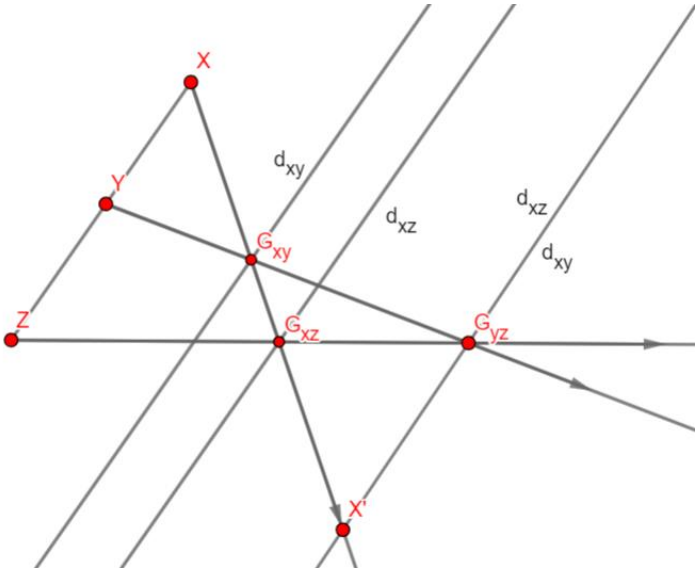
Mà theo (1):

Đường thẳng d_{XY} luôn song song với XY

Đường thẳng d_{XZ} luôn song song với XZ

Ta suy ra $XY \,//\, XZ$

Theo tiên đề Ơ-clit, X,Y,Z thẳng hàng.

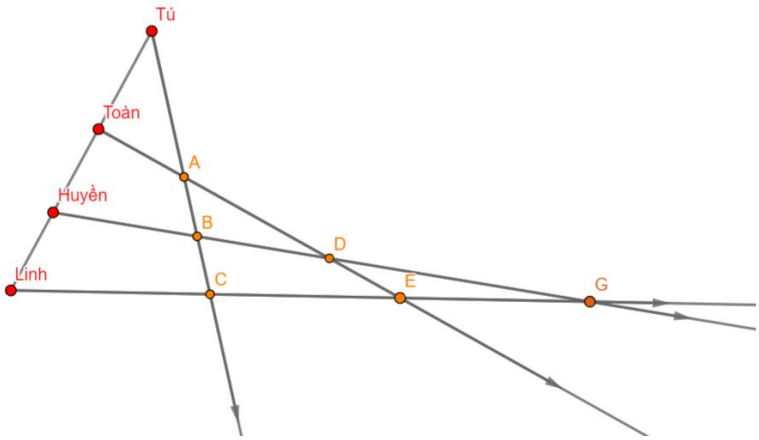


Khi đó, d_{XY}, d_{XZ} luôn song song hoặc trùng nhau

Vậy vị trí cả 3 người sau khi bắt đầu di chuyển luôn nằm trên một đường thẳng song song với đường thẳng đi qua vị trí lúc đầu của 3 người (cố định)

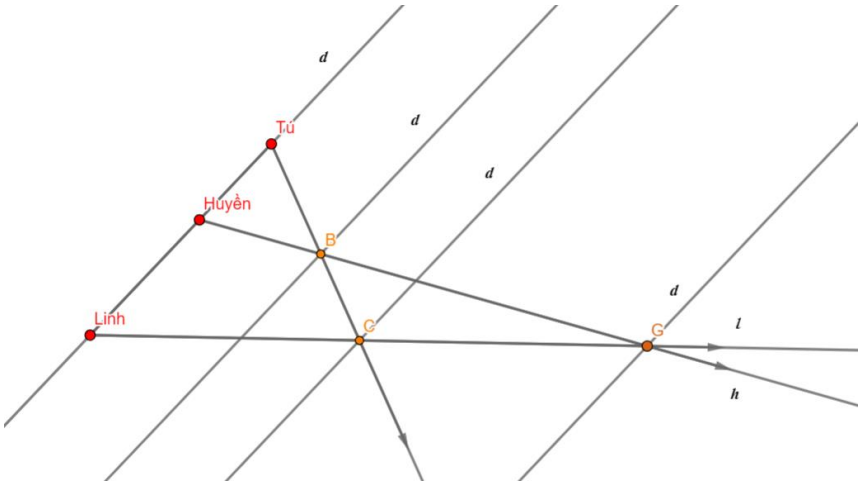
Quay lại bài toán chính:

Áp dụng (2) cho từng bộ 3 người đôi một gặp nhau Tú, Toàn, Huyền và Tú, Toàn, Linh. Ta suy ra được vị trí xuất phát của Tú, Toàn, Huyền nằm trên một đường thẳng và vị trí xuất phát của Tú, Toàn, Linh cũng nằm trên một đường thẳng.



Vậy vị trí xuất phát lúc đầu của cả 4 người đều nằm trên một đường thẳng và khi di chuyển, trong cùng một thời điểm, vị trí cả 4 người sau khi bắt đầu di chuyển luôn nằm trên một đường thẳng song song với đường thẳng nối 4 người ban đầu.

Khi đó vị trí của 3 người Tú, Huyền, Linh sẽ luôn trên nằm trên một đường thẳng (d) và đường này luôn giao với đường đi của Huyền (h) và Linh (l).



Vậy Huyền và Linh có gặp nhau và thời điểm họ gặp nhau chính là khi 3 đường thẳng d,l,h đồng quy tại chính điểm hai người này gặp nhau.

REFERENCE

[1] Gowrishankar, S., & Veena, A. (2018). Introduction to Python programming. Chapman and Hall/CRC.

[2] Sarlette, A. (2009). Geometry and symmetries in coordination control.

[3] Prince, E., & Prince, E. (2004). Matrices: Definitions and Fundamental Operations. *Mathematical Techniques in Chrystallography and Materials Science*, 1-19.

[4] Eisenhart, L. P. (2005). Coordinate geometry. Courier Corporation.