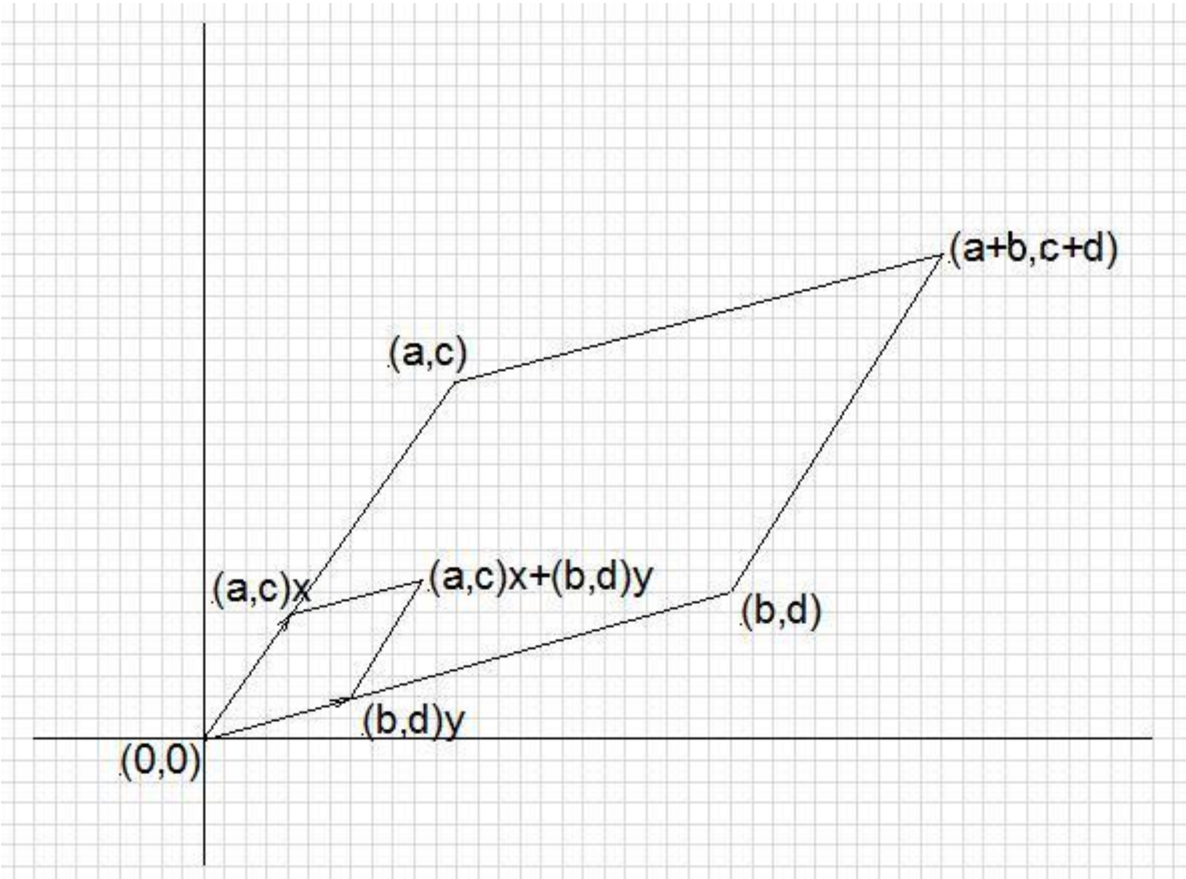
Bài 1: NPAIRS



Đặt $\begin{cases} AX + BY = x \\ CX + DY = y \end{cases} \Leftrightarrow \begin{cases} X = \dfrac{Dx - By}{AD - BC} \\ Y = \dfrac{-Cx + Ay}{AD - BC} \end{cases}$ $(AD - BC \neq 0)$ thì số cặp $(X;Y)$ thực thỏa mãn đề bài bằng số cặp $(x;y)$ nguyên thỏa mãn $0 < \dfrac{Dx - By}{AD - BC} < 1, 0 < \dfrac{-Cx + Ay}{AD - BC} < 1$.

Đó chính là số điểm nguyên nằm trong hình bình hành hình vẽ (phương trình các cạnh của hình bình hành chính là $\dfrac{Dx - By}{AD - BC} = 0, \dfrac{Dx - By}{AD - BC} = 1, \dfrac{-Cx + Ay}{AD - BC} = 0, \dfrac{-Cx + Ay}{AD - BC} = 1$).

Theo định lý Pick với n là số điểm nguyên nằm trong hình bình hành (mà ta đang cần tìm) và m là số điểm nguyên nằm trên các cạnh của hình bình hành thì $S_{hbh} = n + \dfrac{m}{2} - 1$

- Khoảng cách từ điểm có tọa độ $(B;D)$ đến đường thẳng nối hai điểm có tọa độ $(0;0)$ và $(A;C)$ (có phương trình $\frac{-Cx+Ay}{AD-BC}=0 \Leftrightarrow -Cx+Ay=0$) là $\frac{|-CB+AD|}{\sqrt{A^2+C^2}}$ nên diện tích hình bình hành là $S_{hbh}=\frac{|-CB+AD|}{\sqrt{A^2+C^2}}\cdot\sqrt{A^2+C^2}=|AD-BC|$.
- Hoặc áp dụng công thức:

$$S_{hbh}=\left|\sum_{i=0}^{n}(x_i-x_{i+1})(y_i+y_{i+1})\right|=|AD-BC|$$

Số điểm nguyên nằm trên hai cạnh song song của hình bình hành là bằng nhau.

Ta tìm số điểm nguyên trên cạnh nối hai điểm có tọa độ $(0;0)$ và $(A;C)$.

Với $GCD(A;C)=d, A=dA', C=dC', GCD(A';C')=1$ thì các điểm nguyên trên cạnh đó là $(0;0),(A';C'),(2A';2C'),...,(dA';dC')\equiv(A;C)$, như vậy có $d+1$ điểm nguyên (bao gồm cả hai đỉnh $(0;0)$ và $(A;C)$).

Tương tự như vậy trên biên của hình bình hành có tổng cộng $m=2GCD(A;C)+2GCD(B;D)$ điểm nguyên (đã trừ các hằng số +1 để mỗi đỉnh của hình bình hành chỉ tính đúng một lần).

Do đó $n=S_{hbh}-\frac{m}{2}+1=|AD-BC|-GCD(A;C)-GCD(B;D)+1$.

Bài 2: KSTR

Vì tập S_i $(1\le i\le N)$ gồm các phần tử khác nhau $\in[0;9]$ nên ta sử dụng một số nguyên A_i để biểu diễn tập S_i trong đó $Bit(i)$ của A_i bằng 1 nếu phần tử i xuất hiện trọng tập S_i

Ví dụ: $S_i=(0,2,4,5)$ thì $A_i=0000110101$

Khi đó: $S_i-S_j=A_i\ XOR\ (A_i\ AND\ A_j)$

Ví dụ: $S_i=(1,3,8)$ và $S_j=(2,9,3)$ khi đó

$$A_i = 0100001010$$

$$A_j = 1000001100$$

$$A_i\ AND\ A_j = 0000001000$$

$$A_i\ XOR\ (A_i\ AND\ A_{j)}) = 0100000010$$

hay $S_i-S_j=(1,8)$

Từ đó, ta sử dụng quy hoạch động:

$D[0..1023]=D[i]$ = số chuỗi $S_{i_1}-S_{i_2}-\cdots-S_{i_m}=i$

Bài 3: ROBOT

Nhận xét đầu tiên là ta sẽ không bao giờ đi lên hay xuống trên 1 cột khi không cần thiết. Do nhảy sang cột khác nếu có thể, hạ cùng lượng độ cao, mất cùng lượng chi phí, lại được chuyển cột khác.

Khi nào là cần thiết. Ta sẽ có 3 cách nhảy:

Từ cột u tại chiều cai H_u, giả sử nhảy bằng 1 cạnh sang cột v tới chiều cao H_v

- Nếu $H_v < 0$ thì ta sẽ đi lên ở cột u một đoạn rồi mới nhảy sang cột v ở chiều cao 0, chi phí là H_u - 0 + lượng đi lên.
- Nếu $H_v > MaxH_v$ ta sẽ đi xuống ở cột u một đoạn rồi mới nhảy sang cột v ở chiều cao $maxH_v$, chi phí là H_u – $maxH_v$.
- Nếu $0 \leq Hv \leq MaxH_v$, nhảy trực tiếp sang, chi phí là $H_u - H_v$.

Từ đây có 1 số nhận xét:

- Nếu dùng chiến thuật trên, khi xuất phát từ u tại H_u, đến được v tại $H_v > 0$ thì chi phí là $H_u - H_v$.
- Nếu đến được v tại chiều cao 0 thì chi phí còn lớn hơn nữa.
- Khi đã ở 0 thì đi tiếp vẫn luôn luôn ở chiều cao 0

Vậy nên đến được v tại vị trí càng cao thì chi phí càng nhỏ.

Rút ra thuật toán là Dijkstra, $d[u]$ là chi phí ít nhất tới được đỉnh u, tại chiều cao max nhất có thể. (do càng cao chi phí càng ít).

Luôn giữ lại chiều cao lớn nhất tại cùng một cột.

Độ phức tạp: $O(M * logN)$.

----------- HẾT -----------