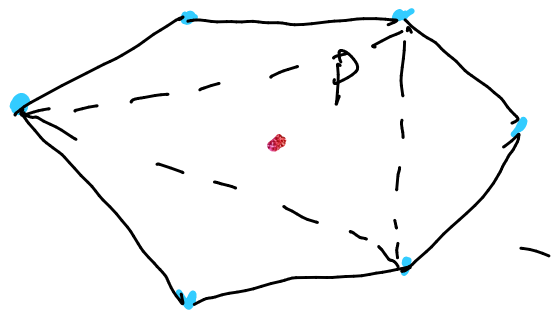
Day 18

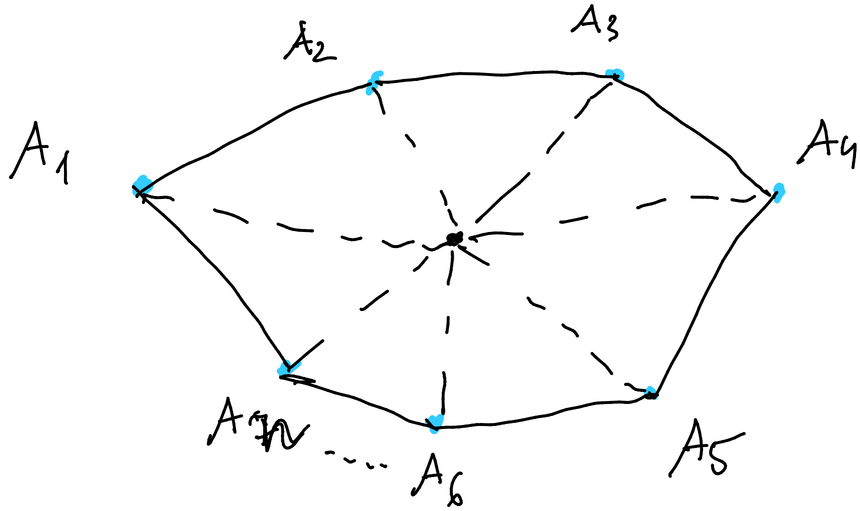
Sunday, August 25, 2024 7:50 AM

Bài A.



=> NX: Nếu P nằm trong đa giác lồi của các đỉnh xanh => ∃ △ 3 đỉnh xanh chứa P

=> [Tìm đa giác lồi => Tìm lồi nhất chứa điểm xanh => Bao lồi
   Kiểm tra P ∈ đa giác (*)



- Sử dụng diện tích để k.tra.

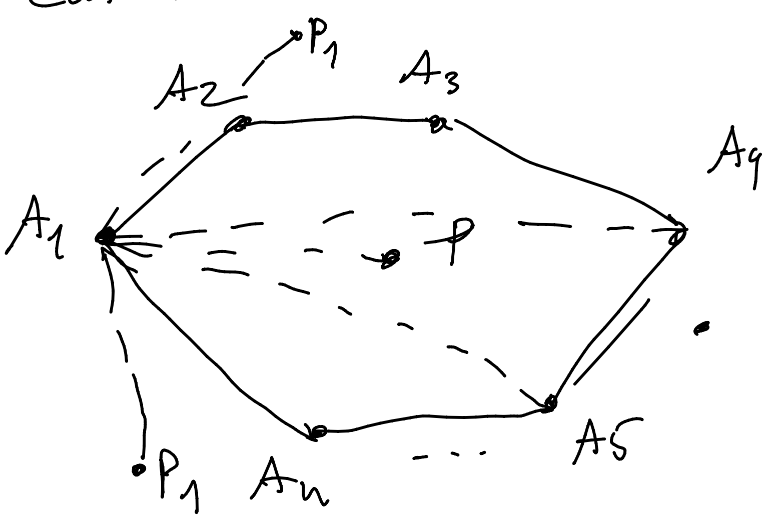
$$S_{A_1A_2\dots A_n} = S_{A_1PA_2} + S_{A_2PA_3} + \dots + S_{A_{n-1}PA_n} + S_{A_nPA_1}$$

$\circlearrowleft_{A_n P A_1}$

$\Rightarrow O(n)$

Với n điểm cần ktra $\Rightarrow O(n^2)$

$\Rightarrow$ Cần tối ưu hàm kiểm tra, về cỡ $O(\log_2 n)$



NX: Ta kh cần ktra hết đa giác.

Chỉ cần tìm A_x và A_{x+1} sao cho

(A_1, P, A_x) khác phía với (A_1, P, A_{x+1})

$\Rightarrow ccw(A_1, P, A_x) * ccw(A_1, P, A_{x+1}) < 0$

Ktra P nằm trong tam giác: $A_1 A_x A_{x+1}$

$\Rightarrow$ Sử dụng diện tích để kiểm tra.

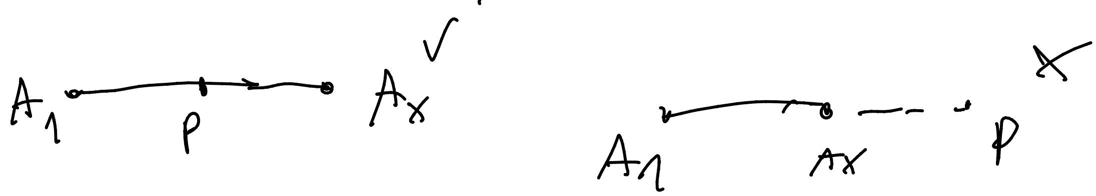
- TH1: P nằm ngoài đa giác, các đỉnh nằm về 1 phía hết về 1 phía

$\Rightarrow ccw(A_1, P, A_2) * ccw(A_1, P, A_n) > 0$

$\Rightarrow$ Loại.

- TH2: $P \in$ cùng đường thẳng chứa 2 điểm đa giác

$$ccw(A_1, P, A_x) == 0$$



$\Rightarrow$ Ktra $P \in A_1A_x$

$\Rightarrow$ Sử dụng t/c $A_1P + A_xP = A_1A_x$

- TH còn lại: Cần tìm A_x sao cho

$$\begin{cases} A_x \text{ nằm trái } A_1P \\ A_{x+1} \text{ nằm phải } A_1P \end{cases}$$

NX: Từ $A_2 \ldots A_x$ sẽ bên trái $\overrightarrow{A_1P}$

và $A_{x+1} \ldots A_n$ sẽ bên phải $\overrightarrow{A_1P}$

$\Rightarrow$ Sử dụng TKNP để tìm A_x

hi = 2 , lo = n , ans

```
hi = 2 , lo = n , ans
while( hi <= lo)
{
    mid = (     ) / 2
    if( ccw(A1, P, Amid) > 0) {
        lo = mid + 1 ;
        ans = mid;
    }
    else {
        hi = mid - 1 ;
    }
}
```

$$\begin{cases} ccw(A_{\cancel{1}}, A_x, P) == 0 \\ ccw(A_1, A_{x+1}, P) == 0 \end{cases} \to \text{check như TH2}$$

$$S_{A_1 A_x A_{x+1}} = S_{A_1 A_x P} + S_{A_1 A_{x+1} P} + S_{A_x A_{x+1} P}$$

$O(1)$ = Tích có hướng.

⇒ Hàm hướng $O(\log_2 n)$

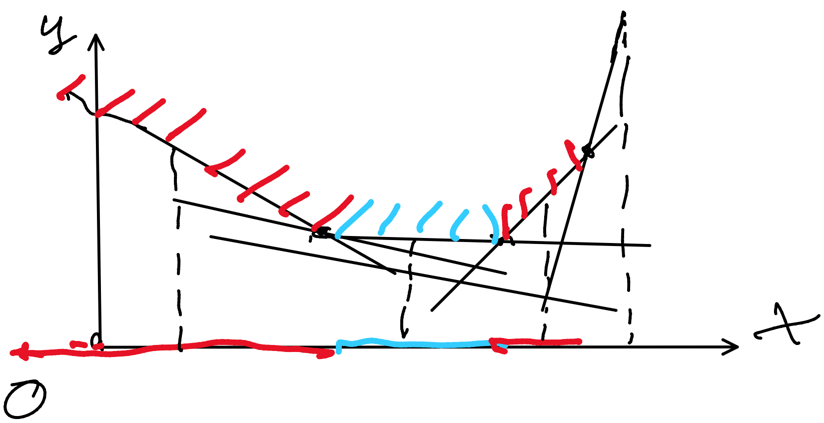
Với n điểm hướng

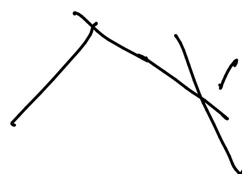
⇒ $O(n \log n)$.

Bài B

Cho a_i, b_i

Với mỗi x, cần tìm $\max\{a_i \cdot x + b_i\}$





- NX1: Các đg t sẽ chia Ox thành các đoạn. Mỗi đường có thể tối ưu cho tối đa 1 đoạn hoặc không tối ưu cho đoạn nào.

- NX2: Khi x tăng dần, nó sẽ được tối ưu bởi

- NX2: Khi x tăng dần, nó sẽ được tối ưu bởi các đường hệ số góc tăng dần.

⇒ Ý tưởng: Với mỗi x, ta xem nó ở đoạn nào là có thể tìm được a_i, b_i tương ứng.

⇒ Convex-Hull trick / optimization.

TH đơn giản: hệ số góc đã được sắp xếp ↑ hay ↓

- Tìm max: Sx ↑.
- Tìm min: Sx ↓

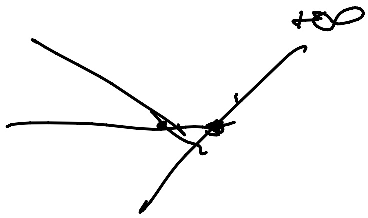
→ Sx các đg tăng dần theo a_i, Nếu $a_i = a_j$ thì sẽ xem $b_i < b_j$.

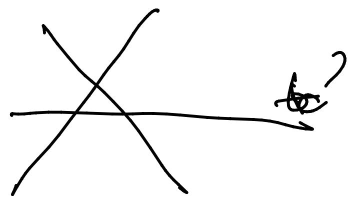
- Sd vector: st, L lại kích thước hiện tại.

Ta sẽ duyệt và thêm bớt lần lượt các đg vào stack rồi cập nhật lại đoạn.

vào stack mới cập nhật lại đoạn.

1 ... i i+1 ⇒ Luôn có đoạn đc tối ưu hơn bởi i+1 nếu so với 1 ... i

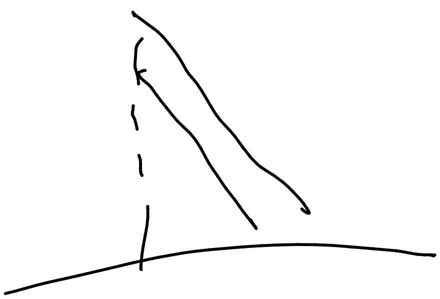




Các TH xảy ra khi thêm đt p

⇒ st[L] là đt cuối trong tập.

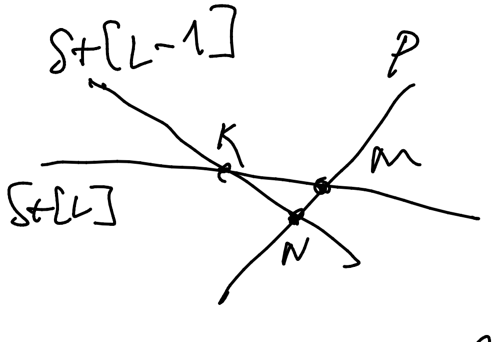
• TH1: st[L] // p ⇒ st[L].b ≤ P.b



⇒ { Loại bỏ st[L]
   --L

• TH2:

~~xét~~ xét st[L], st[L-1], p :



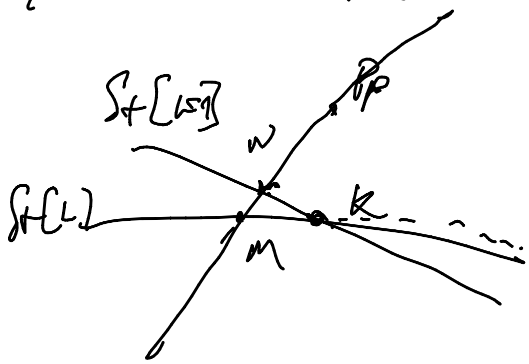
M = P ∩ st[L]

N = P ∩ st[L-1]

st[L]: K → +∞ → { st[L]: K → M

st[L]: K → +∞ → $\begin{cases} st[L]: K \to M \\ P: M \to +\infty \end{cases}$

⇒ Khi $M_x \ge N_x$ thì giữ st[L]



st[L]: K → +∞ ⇒ $\begin{cases} St[L-1]: \ldots \to N \\ P: N \to +\infty \end{cases}$

⇒ Bỏ st[L] (--L)

Cuối cùng:

st[++L] = P

⇒ xét đến khi st rỗng hoặc hết cần bỏ đi st[L].

st[L].a * x + b = P.a * x + b

while (L ≥ 0 && check(P) == 0) --L;

int check (Line P)

```
int check(line p)
{
  //TH1: song song
    if(st[L].a == p.a) return 0;
  if(L<2) return 1;
  //TH2: so sánh M_x, N_x
   M = (p.b - st[L].b)/(st[L].a - p.a);
   N = ______[L-1] ;
   if (M > N) return 1; //Giữ
   return 0;   // loại St[L]
}
```

Sau khi duyệt hết các đg, ta đc mảng St[ ] lưu các đg t/?

Cần tạo ra các đoạn

⇒ Giao điểm 2 đg t/ liền

⇒ Giao điểm 2 đg t liên tiếp.

Segment[i] : Điểm hết thuộc của đoạn i, đc tối ưu bởi st[i]

⇒ ~~Se~~ −∞ ... st[1] ∩ st[2]

st[1] ∩ st[2] ... st[2] ∩ st[3].

⇒ Segment[i] = st[i] ∩ st[i+1]

segment[k] = +∞

Quay lại truy vấn với x,

Cần tìm đoạn x ∈

⇒ Tìm segment[i] ~~gần~~ nhỏ nhất ≥ x.

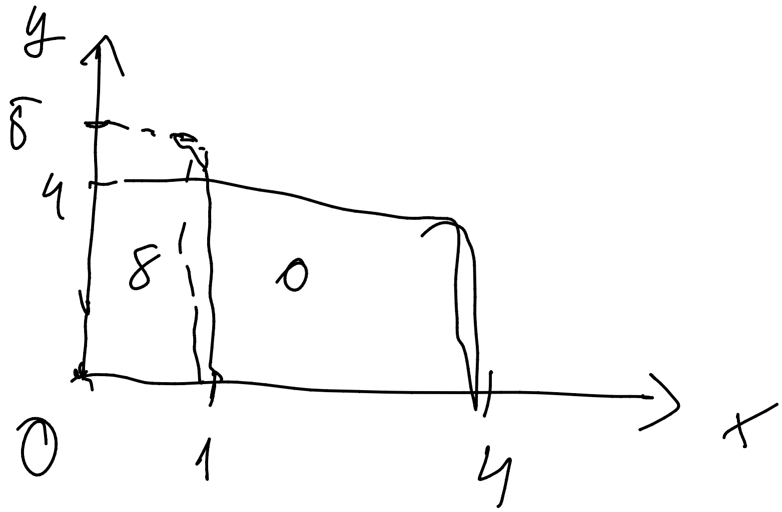
⇒ TKNP trên segment.

O(log n)

$\Rightarrow O(n \log n)$.

1 số dấu hiệu $\begin{cases} \text{Tìm giá trị để max / min} \\ \text{1 yếu tố số ↑ hay ↓} \end{cases}$

Bài C:



$1*5 + (4-1)*4 - 8 - 0 = 9$

Ko có 2 hcn lồng nhau

$\Rightarrow \left[\begin{array}{l} \begin{cases} x_1 > x_2 > \dots > x_n \\ y_1 < y_2 < \dots < y_n \end{cases} \\ \begin{cases} x_1 < x_2 \dots < x_n \\ y_1 > y_2 > \dots > y_n \end{cases} \quad (1) \end{array} \right.$

x, y vai trò như nhau nên quy hết về (1)

x, y vai trò như nhau nên quy hết về (1).

- $Dp[i]$: Gtrị lớn nhất khi xét thành phần i

$Dp[i] = \max\{ Dp[j]$