1. Giá trị lớn nhất [CMAX.*]

Cho n đường thẳng có phương trình

$$y = a_i \cdot x + b_i \ (i = 1 \div n)$$

và m giá trị $x_1, x_2, \dots, x_m$

Hãy tính giá trị của hàm:

$$f(x) = \max\{a_1 \cdot x + b_1, a_2 \cdot x + b_2, \dots, a_n \cdot x + b_n\}$$

tại các giá trị x đã cho

Input:

- Dòng đầu tiên chứa số nguyên dương n $(n \leq 10^5)$
- n dòng tiếp theo, dòng thứ i chứa hai số nguyên a_i, b_i $(|a_i|, |b_i| \leq 10^9)$
- Dòng tiếp theo chứa số nguyên dương m $(m \leq 10^5)$
- m dòng cuối cùng, dòng thứ i chứa số nguyên x_i $(|x_i| \leq 10^9)$

Output: In ra m dòng, dòng thứ i chứa số nguyên $f(x_i)$

Example:

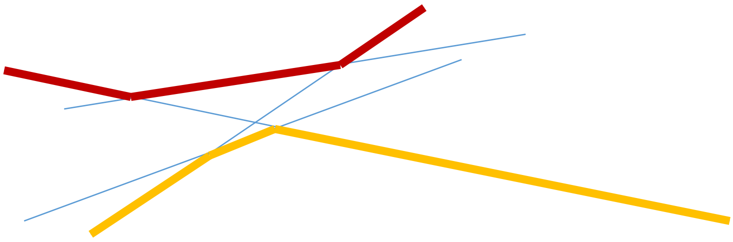
Input	Output
3 1 2 4 6 3 1 3 2 -10 100	14 -8 406

Thuật toán (Lý thuyết)

Thuật toán bao gồm hai bước là xây dựng hàm $f(x)$ và tính giá trị $f(x)$ khi $x = x_0$

a) Xây dựng hàm $f(x)$

Để minh họa ta vẽ đồ thị của hàm $f(x)$ (đường màu đỏ):



Các nhận xét quan trọng:

- Đồ thị hàm $f(x)$ là một đường gấp khúc với mỗi đoạn thẳng của đường gấp khúc này là một đoạn thẳng trên một đường thẳng đã cho.
- Hệ số góc của các đường thẳng tăng dần (theo chiều tăng của x)
- Các đường đoạn thẳng của đường $f(x)$ có tính chất "lồi" tức là một cạnh của một đa giác lồi nào đó.

Do vậy không mất tổng quát ta có thể coi các đường thẳng đã cho có hệ số góc tăng dần:

$$l_1 = \binom{a_1}{b_1} \leq l_2 = \binom{a_2}{b_2} \leq \cdots \leq l_n = \binom{a_n}{b_n}$$

và việc xây dựng hàm f qui về việc tìm dãy đường thẳng $l_1, l_2, \dots, l_m$ để duy trì tính chất cực đại (được xét ở dưới đây).

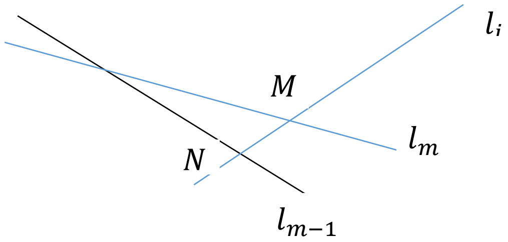
+) Đầu tiên nếu chỉ có 1 đường thẳng l_1 hiển nhiên f cũng chỉ có 1 đoạn thẳng $s_1 = 1$

+) Giả sử đến bước $i - 1$ ta đã xây dựng được đường gấp khúc "max":

$$l_1, l_2, \dots, l_m$$

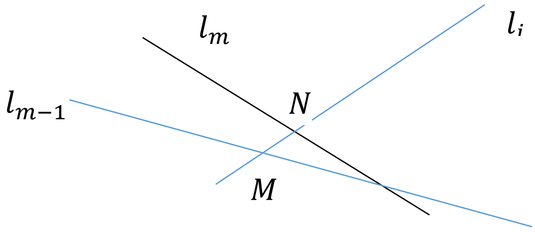
Khi thêm đường thẳng l_i hai trường hợp xảy ra như hình dưới đây:

Trường hợp 1: Hoành độ giao điểm của M lớn hơn hoành độ giao điểm của N (ở đây M là giao điểm của đường thẳng l_i với l_m còn N là giao điểm của l_i với l_{m-1}:



Trong trường hợp này, đường gấp khúc "max" thêm đoạn thẳng l_i

Trường hợp 2: Hoành độ giao điểm của M nhỏ hơn hoặc bằng hoành độ giao điểm của N



Trong trường hợp này thì đường thẳng l_m luôn nằm dưới đường thẳng l_i hoặc l_{m-1} do đó ta có thể bỏ l_m ra khỏi đường gấp khúc max (giảm m). Quá trình này lặp cho đến khi gặp trường hợp 1 hoặc $m = 0$. Khi đó đường l_i sẽ được thêm vào đường gấp khúc.

Như vậy $l_1, l_2, \dots$ sẽ hoạt động như một ngăn xếp. Mỗi đường thẳng sẽ được thêm vào ngăn xếp 1 lần và lấy ra không quá 1 lần. Do vậy khi xét đến l_n với thời gian $O(n)$ ta thu được đường gấp khúc max:

```
    m = 0;
    for (int i = 1; i <= n; ++i) {
        while (m > 0 && !ok (i))
            --m;
        L[++m] = L[i];
    }
```

Ở đây hàm ok(i) kiểm tra xem hoành độ của M có lớn hơn hoành độ của N:

```
bool ok (int i) {
    int u = m;
    if (L[i].ft==L[u].ft) return false;
    if (m==1) return true;
    double xM = 1.0 * (L[u].sc - L[i].sc) / (L[i].ft - L[u].ft);
    u = m-1;
    double xN = 1.0 * (L[u].sc - L[i].sc) / (L[i].ft - L[u].ft);
    return (xM > xN);
}
```

Trong đoạn code trên đã sử dụng:

```
#define ft first
#define sc second
```

Mảng s[1], s[2], ..., s[m] là danh sách các đường thẳng tham gia vào đường gấp khúc max theo giá trị x tăng dần.

Cuối cùng, để mô tả tường minh đường gấp khúc max (phục vụ cho các bài toán qui hoạch động sau này) ta xây dựng 3 mảng:

double x[maxn]; // Mảng tọa độ các giao điểm

int p[maxn]; // Hệ số góc của các đoạn gấp khúc

int q[maxn]; // Hằng số của đoạn gấp khúc

Code minh họa như sau:

```
x[0] = -INF, p[0] = L[1].ft, q[0] = L[1].sc;
for (int i = 1; i < m; ++i) {
    x[i] = 1.0 * (L[i].sc - L[i+1].sc) / (L[i+1].ft - L[i].ft);
    p[i] = L[i+1].ft;
    q[i] = L[i+1].sc;
}
x[m]=INF;
```

Chú ý kết quả:

- $-\infty = x[0] < x[1] < x[2] < \ldots < x[m-1] < x[m] = \infty$
- Trên đoạn $[x_i, x_{i+1}]$ phương trình đường thẳng là $y = p_i \cdot x + q_i$

b) Tính giá trị f(x) tại x=z:

Trước tiên ta tìm giá trị u lớn nhất để $x[u] \le z$ khi đó:

$$f(z) = p_u \cdot z + q_u$$

Code:

```
int u = upper_bound (x, x + m, z) - x - 1;
cout <<  p[u]*z + q[u]);
```

Chú ý rằng đối với bài toán tìm min xử lý tương tự nhưng các hệ số góc giảm dần

2. Nhỏ nhất [CMIN.*]

Cho n đường thẳng có phương trình:

$$y = a_i x + b_i \ \ (i = 1,2,\ldots,n)$$

Và m giá trị $x_1, x_2, \ldots, x_m$ hãy tính giá trị của các hàm:

$$f(x) = min\{a_1 x + b_1, a_2 x + b_2, \ldots, a_n x + b_n\}$$

Tại các giá trị x đã cho.

Input:

- Dòng thứ nhất chứa số nguyên dương $n \le 10^5$
- n dòng tiếp theo, mỗi dòng chứa hai số nguyên a, b mô tả một đường thẳng
- Dòng tiếp chứa số nguyên dương $m \le 10^5$
- m dòng cuối, mỗi dòng thứ i chứa số nguyên x_i

Output: In ra m dòng lần lượt là giá trị tìm được tương ứng với x_i

Example:

Input	Output
3 1 2 4 6 3 1 3 2	4 -34 102

-10 100	

3. Qui hoạch động lồi A [DCVA.*]

Cho dãy số vô hạn $dp_1, dp_2, \ldots$ được xác định bởi:

- dp_1 = c
- $dp_i = \min_{1\le j<i}\{dp_j + b_j \cdot a_i\}$ với $i = 2,3,\ldots$

Ở đây các dãy $b_1, b_2, \ldots$ và $a_1, a_2, \ldots$ cho trước và thỏa mãn điều kiện $b_j \ge b_{j+1}, a_i \le a_{i+1}$

Yêu cầu: Tính dp_n

Input:

- Dòng đầu tiên chứa hai số nguyên n, c $(1 \le n \le 10^5, |c| \le 100)$
- Dòng thứ hai chứa n số nguyên $a_1, a_2, \ldots, a_n$ $(|a_i| \le 100)$
- Dòng thứ ba chứa n số nguyên $b_1, b_2, \ldots, b_n$ $(|b_i| \le 100)$

Output: Một số nguyên là kết quả tìm được

Example:

Input	Output
4 -67 -2 3 6 6 6 5 4 -1	-31

Thuật toán:

Xây dựng tập hợp các đường thẳng $l_j: y_j(x) = b_j \cdot x + dp_j$. Theo giả thiết của đề bài:

$$b_1 \ge b_2 \ge \cdots \ge b_n$$

Nên hệ số góc của các đường thẳng này giảm dần. Theo công thức truy hồi ta cần tìm:

$$\min\{y_1(a_i), y_2(a_i), \ldots, y_{j-1}(a_i)\}$$

Do giả thiết $a_1 \le a_2 \le \cdots \le a_n$ nên các chỉ số u lớn nhất thỏa mãn $x[u] \le a_i$ tăng dần theo i. Vì vậy không cần phải tìm kiếm nhị phân để tính u. Thay vào đó sử dụng kỹ thuật "hai con trỏ" để tìm u tiếp theo. Ngoài ra tập hợp các đường thẳng luôn được thêm vào khi i tăng, do đó ta có thể kết hợp song song hai việc xây dựng đường gấp khúc và tính giá trị.

Tham khảo đoạn code dưới đây:

```
dp[1]=c;
k=1;
x[1]=-oo; p[1]=b[1]; q[1]=dp[1];
u=1;
for(int i=2;i≤n;++i) {
    // u là chỉ số lớn nhất mà x[u]≤a[i];
    while (u≤k && x[u]≤a[i]) ++u;
    --u;

    // Tính giá trị hàm min tại a[i]
    dp[i]=q[u]+p[u]*a[i];

    if (b[i]==p[k] && dp[i]≥q[k]) continue; // Bỏ qua dp[i]+b[i]*x

    // Tính lại đường gấp khúc min
    while (k>1 && !ok(i)) --k;
```

```
    if (u>k) u=k;
    if (k==1 && p[1]==b[i]) q[1]=dp[i]; else {
        p[++k]=b[i]; q[k]=dp[i];
        x[k]=1.0*(q[k-1]-q[k])/(p[k]-p[k-1]);
    }
}
```

Ở đây hàm ok(i) được viết như sau:

```
bool ok(int i) {
     if (b[i]==p[k]) return false;
     double xm=1.0*(dp[i]-q[k])/(p[k]-b[i]);
     double xn=1.0*(dp[i]-q[k-1])/(p[k-1]-b[i]);
     return (xn<xm);
}
```

4. Thuê đất [RENTLAND.*]

Bờm vừa lập công lớn cho Phú Ông khi phát hiện ra kho báu được chôn trên đất nhà Phú Ông. Do vậy, Phú Ông đã cho Bờm thuê đất với giá đặc biệt. Vì biết Bờm là một người không thông minh nên Phú Ông đã nghĩ ra cách tính giá đất quái dị hòng lừa Bờm. Giá thuê đất được Phú Ông tính như sau:

Giả sử Phú Ông có N $(1 \leq N \leq 50000)$ miếng đất cho Bờm thuê với kích thước $\boldsymbol{l_i} \times \boldsymbol{w_i}$ với $\boldsymbol{l_i}$ là chiều dài và $\boldsymbol{w_i}$ là chiều rộng của các miếng đất ($1 \leq l_i \leq 10^6$; $1 \leq w_i \leq 10^6$). Bờm có thể nhóm các miếng đất thành một nhóm để thuê trong cùng một lượt. Số tiền thuê trong mỗi nhóm mà Bờm phải trả được tính bằng chiều dài lớn nhất của miếng đất trong nhóm nhân với chiều rộng lớn nhất của miếng đất trong nhóm. Chẳng hạn, nếu Bờm xếp hai miếng đất có kích thước (3,4) và (4,3) vào một nhóm thì số tiền phải trả cho nhóm này là 4 x 4 =16 (xu).

Bờm muốn bạn giúp xem cần phải trả ít nhất là bao nhiêu để có thể thuê được toàn bộ các miếng đất của Phú Ông?

Input:

- Dòng đầu ghi số N
- Mỗi dòng trong N dòng sau ghi hai số nguyên $\boldsymbol{w_i}$ và $\boldsymbol{l_i}$

Output: Một số nguyên duy nhất là số tiền nhỏ nhất mà Bờm phải trả cho Phú Ông.

Example:

Input	Output	Giải thích
4 100 1 15 15 20 5 1 100	500	Các nhóm (100,1) (15,15), (20,5) (1,100) Khi đó số tiền thuê là: 100*1+15*20+1*100=500

5. Chọn hình chữ nhật [SREC.*]

Trên mặt phẳng tọa độ cho n hình chữ nhật. Hình chữ nhật thứ i có tọa độ góc trái-dưới là (0,0) và tọa độ góc trên-phải là (x_i, y_i). Mỗi hình chữ nhật được gán một giá trị a_i - trọng số của nó. Không có hai hình chữ nhật lồng nhau.

Giá trị của một tập hợp S các hình chữ nhật được tính bằng diện tích phần mặt phẳng tọa độ bị phủ bởi các hình chữ nhật này trừ đi tổng trọng số các hình chữ nhật trong tập S.

Yêu cầu: Hãy tìm tập S có giá trị lớn nhất

Input:

- Dòng 1: Số nguyên dương n ($n \leq 10^6$)
- Dòng 2...$n+1$: Dòng $i+1$ chứa ba số nguyên x_i, y_i, a_i thể hiên hình chữ nhật thứ i có đỉnh trên-phải là (x_i, y_i) và có trọng số a_i. ($1 \leq x_i, y_i \leq 10^9; 0 \leq a_i \leq x_i \cdot y_i$)

Output: In ra một số nguyên - Giá trị lớn nhất của một tập S các hình chữ nhật

Example:

Input	Output
3 4 4 8 1 5 0 5 2 10	9
4 6 2 4 1 6 2 2 4 3 5 3 8	10

6. Mái che [CWALKWAY.*]

Nhà trường vừa mới xây dựng một lối đi mới có thể được mô tả như là một trục tọa độ. Trên lối đi này có một số vị trí quan trọng cần phải có mái che để tránh mưa, nắng. Với các vị trí khác trên lối đi không nhất thiết phải có mái che phủ nó.

Nhà thầu xây dựng đã tính toán chi phí làm mái che. Nếu làm mái che từ điểm tọa độ x đến điểm toa độ y thì sẽ mất chi phí là $c + (x - y)^2$, trong đó c là một hằng số cho trước. Lưu ý rằng khi $x = y$ (chỉ che 1 điểm) thì chi phí của mái che là c.

Yêu cầu: Cho tọa độ các điểm cần có mái che, hãy xác định chi phí nhỏ nhất để che được các điểm này.

Input:

- Dòng 1: Hai số nguyên dương n, c ($1 \leq n \leq 10^6; 1 \leq c \leq 10^9$)
- Dòng 2...$n+1$: Dòng $i+1$ chứa số nguyên a_i là tọa độ của điểm che thứ i. Tọa độ các điểm đươc liệt kê theo thứu tự tăng dần và có giá trị trong khoảng $[1,10^9]$

Output: Môt số nguyên dy nhất là chi phí nhỏ nhất tìm được.

Example:

Input	Output
10 5000 1 23 45 67 101 124 560 789 990 1019	30726

7. Đặc công [COMMANDO.*]

Bạn là chỉ huy của đội quân gồm n binh sỹ, được đánh số từ 1 đến n. Đối với trận chiến phía trước, bạn có kế hoạch chia những người lính này thành nhiều đơn vị đặc công. Để thúc đẩy sự đoàn kết động viên tinh thần, mỗi đơn vị đặc công sẽ bao gồm một chuỗi các binh sỹ có thứ tự liên tục: $i, i+1, \dots j$

Người lính thứ i được đánh giá hiệu quả chiến đấu bằng một số nguyên x_i. Ban đầu, hiệu quả chiến đấu của một đơn vị đặc công $(i, i+1, \dots, j)$ được tính bằng cách cộng hiệu quả chiến đấu của các binh sỹ. Nói cách khác nó được tính theo công thức $x = x_i + x_{i+1} + \cdots + x_j$.

Tuy nhiên, sau khi chiến đấu hiệu quả của một đơn vị đặc công được tính theo công thức mới như sau: Gọi x là hiệu quả của đơn vị ở thời điểm ban đầu khi đó hiệu quả mới được tính theo công thức $a \cdot x^2 + b \cdot x + c$. Ở đây a, b, c là các hệ số cho trước $(a < 0)$.

Yêu cầu: Hãy tìm cách chia các binh sỹ thành các đơn vị đặc công sao cho tổng hiệu quả sau khi chiến đấu của các đơn vị này là lớn nhất.

Ví dụ, chẳng hạn bạn có 4 người lính với $x_1 = 2, x_2 = 2, x_3 = 3, x_4 = 4$ và $a = -1, b = 10, c = -20$. Trong trường hợp này giải pháp tối ưu là chia thành ba đơn vị đặc công: Đơn vị thứ nhất chứa binh lính 1 và 2, đơn vị thứ hai chứa binh lính 3 và đơn vị thứ ba chứa binh lính 4. Khi đó hiệu quả chiến đấu ban đầu của 3 đơn vị đó lần lượt là 4, 3, 4 còn hiệu quả sau khi đi chinh chiến của 3 đơn vị trên lần lượt là 4, 1, 4. Tổng hiệu quả sau chiến đấu của 3 đơn vị là 9 và đây là cách chia tối ưu nhất.

Input:

- Dòng 1 chứa số nguyên dương n $(n \le 10^6)$
- Dòng 2 chứa ba số nguyên a, b, c $(-5 \le a \le -1, |b| \le 10^7, |c| \le 10^7)$
- Dòng 3 chứa n số nguyên $x_1, x_2, \dots x_n$ $(1 \le x_i \le 100)$

Output: Một số nguyên là giá trị tổng độ hiệu quả sau chiến đấu của cách chia tìm được

Example:

Input	Output
4 -1 10 -20 2 2 3 4	9

Subtasks:

- Subtask 1: $n \le 1000$ [20%]
- Subtask 2: $n \le 10000$ [30%]
- Subtask 3: $n \le 10^6$ [50%]

8. Quét lá [LEAVES.*]

Trong một ngày mùa thu tuyệt đẹp, Bờm và Cuội nhận thấy rằng con hẻm trong vườn nơi họ hay dạo chơi cùng nhau có khá nhiều lá. Họ quyết định gom thành đúng K đống lá.

Có n chiếc lá nằm thẳng hàng với trọng lượng khác nhau, khoảng cách giữa hai chiếc lá liên tiếp bằng 1. Nghĩa là, chiếc lá đầu tiên có tọa độ 1, chiếc lá thứ hai có tọa độ 2, ..., chiếc lá thứ n có tọa độ n.

Bờm và Cuội thực hiện việc gom lá trước khi rời khỏi vườn, do vậy những chiếc lá chỉ có thể di chuyển về phía bên trái. Chi phí di chuyển một chiếc lá bằng tích của trọng lượng chiếc lá và khoảng cách di chuyển. Hiển nhiên một trong K đống lá sẽ nằm ở tọa độ 1, tuy nhiên những đống lá còn lại có thể nằm ở bất kỳ vị trí nào.

Yêu cầu: Tìm chi phí nhỏ nhất để di chuyển n chiếc lá thành đúng K đống.

Input:

- Dòng 1: Chứa hai số nguyên dương n, K $(n \le 10^5, K \le 10, K < n)$
- Dòng 2: Chứa n số nguyên $a_1, a_2, \dots, a_n$ $(1 \le a_i \le 100)$ lần lượt là trọng lượng của các lá 1, 2, ..., n

Output: Một số nguyên là chi phí nhỏ nhất để gom n chiếc lá thành đúng K đống

Example:

Input	Output
5 2 1 2 3 4 5	13

Subtasks:

- Subtask 1: $n \leq 1000$ [50%]
- Subtask 2: $n \leq 100000$ [50%]