Bài 1: Lật bánh

File dữ liệu vào:	LATBANH.INP
File kết quả:	LATBANH.OUT
Hạn chế thời gian:	1s
Hạn chế bộ nhớ:	512M

ATM muốn dùng bữa. Anh ấy vừa trở về từ cửa hàng, nơi anh ấy mua một chiếc bánh rán, việc còn lại chỉ là rán chín lên. Dòng chữ trên bao bì nói rằng chiếc bánh cần được rán trong chảo ở lửa vừa phải trong chính xác $2N$ giây và đầu tiên bạn cần rán chính xác N giây ở một mặt, và chính xác N giây ở mặt còn lại. ATM đã tìm được một chiếc chảo và đốt lửa vừa phải, nhưng sau đó anh nhận ra rằng mình có thể không lật được chiếc bánh chính xác N giây sau khi bắt đầu rán.

ATM quá bận rộn và có thể bị phân tâm khi chỉ xoay miếng thịt ở một số thời điểm nhất định. Cụ thể, có K khoảng thời gian mà anh ấy có thể xoay miếng thịt, khoảng thời gian thứ i trong số đó bắt đầu từ thời điểm l_i giây kể từ khi bắt đầu nấu đến r_i giây. ATM quyết định rằng không cần thiết phải lật chính xác một lần chiếc bánh khi đang nấu, thay vào đó, anh ta sẽ lật nó nhiều lần để chiếc bánh được rán đúng N giây ở một mặt và N giây ở mặt còn lại.

Giúp ATM, tìm xem liệu anh ta có thể hoàn thành kế hoạch của mình không, nếu anh ta chỉ có thể lật chiếc bánh rán trong khoảng thời gian đã định, thì số lần tối thiểu anh ta sẽ phải lật chiếc bánh là bao nhiêu.

Dữ liệu

- Dòng đầu tiên chứa hai số nguyên N và K ($1 \le N \le 5 \cdot 10^5$, $1 \le K \le 100$)
- K dòng tiếp theo chứa mỗi dòng chứa hai số nguyên l_i và r_i ($0 \le l_i \le r_i \le 2 \cdot N$), điều đó có nghĩa là ATM có thể lật bất cứ lúc nào trong khoảng thời gian $[l_i, r_i]$. Dữ liệu đảm bảo rằng $l_i > r_{i-1}$ với $2 \le i \le k$.

Kết quả

Đưa ra số lần lật ít nhất cần thực hiện. Nếu không thể thì in ra -1.

Ví dụ

LATBANH.INP	LATBANH.OUT
10 2 3 5 11 13	2
10 3 3 5 9 10 11 13	1

Hạn chế

- Subtask 1: 30% số điểm có $N \le 10$
- Subtask 2: 20% số điểm có kết quả không quá 2 lần lật
- Subtask 3: 20% số điểm có $N \le 1000$
- Subtask 4: 20% số điểm có $N \le 10^5$
- Subtask 5: 10% số điểm còn lại không có ràng buộc gì thêm.

- Subtask 1: Duyệt 2^{2N}.
- Subtask 2: Kiểm tra trường hợp kết quả = 1 nếu thời điểm N có khoảng lật, với kết quả = 2 thì tìm 2 đoạn mà có chênh lệch trong khoảng N bằng cách duyệt K^2.
- Subtask 3: Quy hoạch động $dp[i][j]$ là số lần lật ít nhất đến thời điểm i và mặt trên đã được rán j giây:
  - Đến thời điểm i nếu không lật bánh thì $dp[i][j] = dp[i-1][j]$
  - Nếu có thể lật bánh thì $dp[i][j] = dp[i-1][i-j] + 1$
- Subtaks 4, 5: ta có nhận xét rằng trong một khoảng thời gian có thể lật từ $[l_i, r_i]$ thì tối đa ta chỉ lật 2 lần, do đó ta chỉ cần quan tâm đến các thời điểm cuối cùng của mỗi khoảng. gọi $dp[i][j]$ là số lần lật ít nhất đến thời điểm r_i và mặt trên đã được rán j giây:
  - $dp[i][j] = dp[i-1][j]$ nếu không lật
  - $dp[i][j] = \min_{l_i \le t \le r_i}(dp[i-1][t-j]) + 1$ khi dùng 1 lật
  - $dp[i][j] = \min_{j-(r_i-l_i) \le j' < j}(dp[i-1][j']) + 2$ khi dùng 2 lật

  Ta sẽ dùng deque để tính nhanh được $dp[i][j]$ với độ phức tạp $O(NK)$. Một số cách dp khác có thể sẽ cần dùng cấu trúc cây để tính nhanh thì thuật toán là $O(NK \log N)$ thì chỉ qua được subtask 4.

Bài 2: Chiến tranh

File dữ liệu vào:	`CHIENTRANH.INP`
File kết quả:	`CHIENTRANH.OUT`
Hạn chế thời gian:	1s
Hạn chế bộ nhớ:	512M

Có L quốc gia đi chinh phạt các vùng đất mới. Cụ thể, có N thành phố được kết nối với nhau bằng những con đường hai chiều. Tại thời điểm 0, quốc gia i đang chiếm đóng thành phố c_i, và các quốc gia bắt đầu đi chinh phạt các vùng đất mới theo quy tắc sau:

1. Nếu một quốc gia chiếm đóng được một thành phố, họ sẽ cử các chiến binh di chuyển đến các thành phố khác có đường đi với thành phố vừa chiếm đóng, tất cả đều di chuyển cùng tốc độ.
2. Nếu các chiến binh của hai quốc gia khác nhau gặp nhau trên một con đường, họ sẽ chiến đấu với nhau mãi mãi trên con đường đó.
3. Nếu các chiến binh của hai quốc gia khác nhau tới một thành phố cùng một lúc, họ sẽ chiến đấu với nhau mãi mãi trên thành phố đó.
4. Nếu các chiến binh của một quốc gia đến thành phố đang có chiến tranh xảy ra thì họ cũng tham gia vào cuộc chiến.
5. Nếu các chiến binh đến một thành phố chưa có quốc gia nào khác chiếm đóng thì họ sẽ chiếm đóng thành phố này và tiếp tục cử các chiến binh di chuyển đến các thành phố khác như quy tắc 1.

Chú ý với các quy tắc trên thì chỉ có duy nhất một quốc gia chiếm được một thành phố. Nếu chiến binh của hai quốc gia cùng đến một thành phố thì họ sẽ chiến đấu mãi mãi ở thành phố đó mà không di chuyển đến các thành phố khác.

Hãy xác định các cặp quốc gia chiến đấu với nhau.

Dữ liệu

- Dòng đầu tiên chứa N, L, M $(1 \leq N \leq 10^5, 2 \leq L \leq 50, 1 \leq M \leq 10^5)$
- L dòng tiếp theo chứa c_i là thành phố mà quốc gia i chiếm đóng ban đầu.
- M dòng tiếp theo mỗi dòng chứa ba số nguyên u, v, w mô tả một con đường giữa hai thành phố u và v có thời gian di chuyển là w $(0 \leq u \neq v < N, 1 \leq w \leq 2000)$. Dữ liệu đảm bảo không có đường đi nào nối một thành phố với chính nó, không có nhiều hơn một đường đi nối hai thành phố.

Kết quả

- Ghi ra nhiều dòng, mỗi dòng là hai số a, b mô tả quốc gia a sẽ chiến đấu với quốc gia b $(a < b)$. Các cặp được liệt kê theo thứ tự từ điển.

Ví dụ

CHIENTRANH.INP	CHIENTRANH.OUT
8 4 8 0 2 4 6 0 1 100 1 2 100 2 3 100 3 4 100 4 5 100 5 6 100 6 7 100 7 0 1000	0 1 0 3 1 2 2 3

Hạn chế

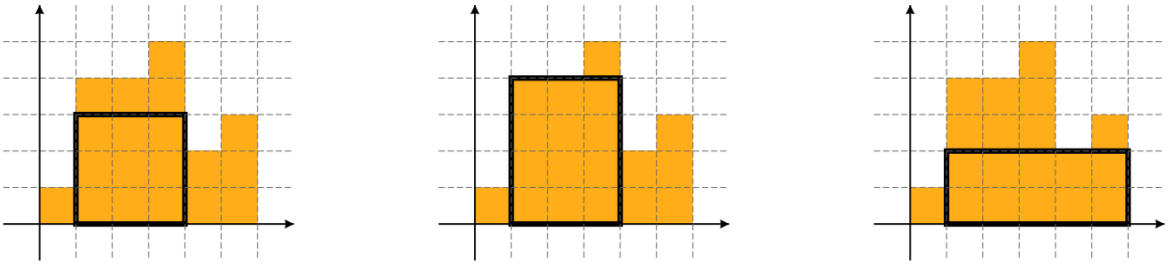
- Subtask 1: 20% số điểm có đồ thị là một đường thẳng $(0, 1), (1, 2), ..., (N - 2, N - 1)$
- Subtask 2: 20% số điểm có $w = 1$
- Subtask 3: 20% số điểm không có chiến tranh tại các thành phố
- Subtask 4: 20% số điểm có $1 \leq N, M \leq 2000$
- Subtask 5: 20% số điểm còn lại không có ràng buộc gì thêm.

- Subtask 1: Kết quả là các cặp quốc gia liên tiếp nhau khi sort theo c_i
- Subtask 2: Với $w = 1$ thì ta sẽ sử dụng BFS để loang từ các thành phố mà các quốc gia chiếm đóng ban đầu ra các thành phố khác. Gọi d_u là thời gian nhỏ nhất mà thành phố u có quốc gia đến chiếm đóng, S_u là danh sách các quốc gia đến được thành phố u. Khi u loang đến v:
  - Nếu v chưa có quốc gia nào đến thì ta tính được $d_v = d_u + 1$, thêm quốc gia chiếm u vào S_v
  - Nếu v đã có quốc gia chiếm đóng, nếu $d_v == d_u + 1$ thì thêm quốc gia chiếm đóng u vào S_v, nếu S_v có nhiều hơn 1 quốc gia thì không loang từ v sang các thành phố khác nữa. Nếu $d_v == d_u + 1$ thì hai quốc gia chiếm đóng u và v sẽ chiến đấu với nhau tại con đường nối u và v. Nếu $d_v < d_u + 1$ thì có nghĩa là v đang có chiến tranh giữa các quốc gia, ta chỉ việc thêm quốc gia chiếm đóng u vào S_v.
- Subtask 3: Thay BFS bằng Dijkstra và tính d_u, vì hai quốc gia không gặp nhau cùng lúc tại một thành phố nên mỗi thành phố sẽ chỉ có một quốc gia chiếm đóng, ta sẽ tìm được quốc gia chiếm đóng thành phố u trong quá trình Dijkstra, sau đó duyệt các con đường nối hai thành phố (u, v) và kiểm tra quốc gia chiếm đóng hai thành phố này có khác nhau không, nếu khác nhau thì hai quốc gia này sẽ chiến đấu với nhau.
- Subtask 4: Vì $M \leq 2000$ và $w \leq 2000$ nên ta có thể tạo các thành phố trung gian giữa hai thành phố và làm như subtask 2.
- Subtask 5: Vấn tư tưởng thay BFS bằng Dijkstra, và xét các trường hợp như subtask 2, ta chỉ cần thay đổi một chút điều kiện là được.

Bài 3: Biểu đồ

File dữ liệu vào:	BIEUDO.INP
File kết quả:	BIEUDO.OUT
Hạn chế thời gian:	2s
Hạn chế bộ nhớ:	512M

Histogram là một biểu đồ biểu diễn thống kê của một loại dữ liệu, trong đó với mỗi giá trị i từ 1 đến n người ta thống kê xem trong dữ liệu đó có bao nhiêu lần xuất hiện giá trị i gọi là h_i. Biểu đồ được vẽ với n cột đánh số từ 1 đến n từ trái qua phải, mỗi cột có độ rộng bằng 1, cột thứ i có độ cao là h_i.



Các nhà phân tích dữ liệu thường cố tìm ra các đặc điểm riêng của từng loại dữ liệu để có thể khai thác dữ liệu đó vào các bài toán thực tế. Có một cách họ thường làm là với một số tự nhiên p, họ xác định có bao nhiêu hình chữ nhật khác nhau mà các đỉnh của các hình chữ nhật là các tọa độ nguyên, có một cạnh nằm trên trục Ox, hình chữ nhật này phải được phủ hoàn toàn bởi biểu đồ, và diện tích của hình chữ nhật phải không nhỏ hơn p.

Yêu cầu: cho biểu đồ histogram và số nguyên p, hãy xác định số hình chữ nhật khác nhau thỏa mãn các điều kiện trên.

Dữ liệu

- Dòng đầu tiên chứa hai số nguyên n và p.
- Dòng thứ hai chứa n số nguyên h_i.

Kết quả

- Ghi ra một số nguyên duy nhất là số lượng hình chữ nhật khác nhau thỏa mãn các điều kiện trên.

Ví dụ

BIEUDO.INP	BIEUDO.OUT
6 9 1 4 4 5 2 3	3

Hạn chế

- Subtask 1: 20% số điểm có $1 \le n \le 3000, 1 \le p \le 10^{12}, 1 \le h_i \le 10^9$
- Subtask 2: 20% số điểm có $1 \le n \le 10^5, 1 \le p \le 10^8, 1 \le h_i \le 10^3$
- Subtask 3: 20% số điểm có $1 \le n \le 10^5, p = 1, 1 \le h_i \le 10^9$
- Subtask 4: 20% số điểm có $1 \le n \le 10^5, 1 \le p \le 10^6, 1 \le h_i \le 10^9$
- Subtask 5: 20% số điểm có $1 \le n \le 10^5, 1 \le p \le 10^{14}, 1 \le h_i \le 10^9$

- Subtask 1: Duyệt n^2 chọn hai cột (i, j) là cột bắt đầu và cột kết thúc của hình chữ nhật, => chiều cao nhỏ nhất sẽ là $\frac{p}{j-i+1}$ và lớn nhất là $\min(h_i, ..., h_j)$ => số hình chữ nhật thỏa mãn là $\sum_{1 \le i \le j \le n} \max(0, \min(h_i, ..., h_j) - \frac{p}{j-i+1})$

- Subtask 2: Duyệt $n \times \max(h_i)$ chọn cột i là cột cuối cùng của hình chữ nhật, ta đếm xem với chiều cao H thì có bao nhiêu cột j thỏa mãn bằng cách duy trì mảng $t[H]$ là cột gần nhất mà chiều cao nhỏ hơn H, khi đến cột i ta sẽ update $t[H] = i$ với $H > h_i$. Như vậy số cột j thỏa mãn với cột i có độ cao H là $i - t[H]$.

- Subtask 3: Dùng stack ta sẽ có được danh sách $i_1 < i_2 < ... < i$ với $h_{i_1} < h_{i_2} < ... < h_i$, như vậy với cột i nếu ta chọn j từ $i_1 + 1$ đến i_2 thì ta có thể tạo được hình chữ nhật có chiều cao từ $h_{i_1} + 1$ đến h_{i_2} => số hình chữ nhật có thể tạo ra là $(i_2 - i_1) \times (h_{i_2} - h_{i_1})$ => số hình chữ nhật tạo được với cột i là $i_1 \times h_1 + (i_2 - i_1) \times (h_{i_2} - h_{i_1}) + (i_3 - i_2) \times (h_{i_3} - h_{i_2}) + ...$ Để tính nhanh được ta cần cộng dồn khi update stack.

- Subtask 4: Với chiều cao H ta sẽ có các đoạn $[l, r]$ là các cột liên tiếp có chiều cao $\ge H$ (Các đoạn $[l, r]$ được xây dựng bằng DSU khi xét độ cao giảm dần):
  - Với $H \ge p$ thì ta làm như subtask 3 với từng đoạn $[l, r]$
  - Với $H < p$ với mỗi đoạn $[l, r]$ thì số hình chữ nhật có chiều cao H thỏa mãn diện tích $\ge p$ là $C^2_{r-l+1-p/H+1}$, với mỗi độ cao H ta duyệt qua tất cả các đoạn $[l, r]$ để tính kết quả. Ta có thể sử dụng BIT hoặc ST để tính được tổng $C^2_{r-l+1-p/H+1}$ với các đoạn $[l, r]$. Với đoạn độ dài d và $\frac{p}{H} = k$ thì số hình chữ nhật thỏa mãn là $\frac{(d-k)\times(d-k+1)}{2} = \frac{d^2-2kd+d-k+k^2}{2}$, ta có thể bỏ qua mẫu 2 do tổng $\sum C^2_d$ của tất cả các đoạn sẽ chia hết cho 2 nên ta sẽ cần tính tổng $\sum((d^2 + d) - 2kd + (k^2 - k))$, ta sử dụng cây BIT hoặc ST để tính tổng cho mỗi thành phần.

- Subtask 5: Ý tưởng giống subtask 4, ta có nhận xét rằng chiều rộng của các hình chữ nhật chỉ từ 1 tới N => các chiều cao cần xét chỉ là $\frac{P}{i}$.