Processing Queries

Link submit: http://codeforces.com/problemset/problem/644/B

Solution:

C++	http://ideone.com/rsFRdu
Java	http://ideone.com/toAaS6
Python	http://ideone.com/lC9tnj

Tóm tắt đề:

Có n truy vấn mà server cần xử lý. Truy vấn thứ i tại thời điểm t_i sẽ cần d_i đơn vị thời gian để giải quyết. Cho rằng các thời điểm t_i phân biệt với nhau.

Đối với một truy vấn, sẽ có 3 trường hợp xảy ra:

1. Nếu server đang ở trạng thái chờ, truy vấn sẽ được xử lí ngay lập tức.
2. Nếu server đang ở trạng thái bận và có ít hơn b truy vấn cần phải xử lí thì truy vấn mới sẽ được thêm vào cuối hàng đợi.
3. Nếu server đang ở trạng thái bận và đã có b truy vấn cần phải xử lì thì truy vấn mới bị từ chối.

Với mỗi truy vấn, tìm thời điểm mà server sẽ xử lí xong. Nếu truy vấn không được xử lí thì in ra -1.

Input:

Dòng đầu tiên gồm 2 số nguyên *n* và *b* ($1 \le n, b \le 200.000$) – tổng số lượng truy vấn và số lượng truy vấn tối đa có thể xử lí.

n dòng tiếp theo, mỗi dòng gồm 2 số nguyên t_i và d_i ($1 \le t_i, d_i <= 10^9$) với t_i là thời gian truy vấn được gửi đi và d_i là thời gian mà server cần để xử lí truy vấn đó.

Output:

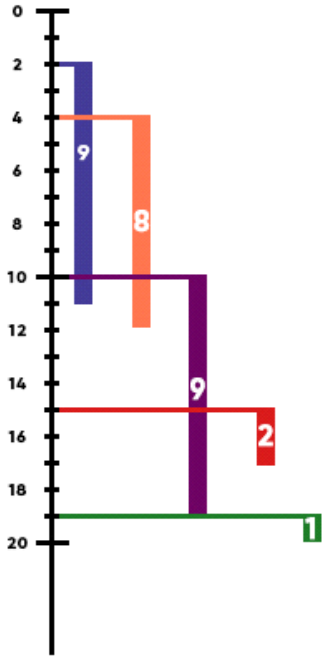
In ra trên cùng một dòng là n số $e_1, e_2, \ldots, e_n$ với e_i là thời điểm server xử lí xong truy vấn thứ i. Nếu truy vấn không được xử lí thì in ra -1.

Ví dụ:

5 1 2 9 4 8 10 9 15 2 19 1	11 19 -1 21 22

Ví dụ 1:

1. Máy chủ sẽ bắt đầu xử lí truy vấn đầu tiên tại thời điểm 2, và kết thúc vào thời điểm 11.



2. Tại thời điểm 4 giây, truy vấn thứ 2 xuất hiện và được đẩy vào hàng đơi.

3. Tại thời điểm 10, truy vấn thứ 3 xuất hiện. Tuy nhiên, vào lúc này máy chủ đang xử lí truy vấn đầu tiên. Ta lại có b = 1, và đã có truy vấn thứ 2 đang trong hàng đợi chờ xử lí, do vậy truy vấn thứ 3 bị từ chối xử lí.

4. Tại thời điểm 11, máy chủ kết thúc việc xử lí truy vấn đầu tiên và bắt đầu xử lí truy vấn thứ 2 đang nằm trong hàng đợi.

5. Tại thời điểm thứ 15, truy vấn thứ 4 xuất hiện và được đẩy vào hàng đợi do máy chủ đang xử lí truy vấn thứ 2.

6. Tại thời điểm 19, có 2 việc xảy ra cùng lúc: máy chủ vừa mới xử lí xong truy vấn thứ 2 và truy vấn thứ 5 xuất hiện. Như đã nói ở đề bài, đầu tiên máy chủ kết thúc xử lí truy vấn 2, lấy truy vấn 4 ra khỏi hàng đợi để xử lí và truy vấn 5 được thêm vào hàng đợi.

7. Máy chủ kết thúc xử lí truy vấn 4 tại thời điểm 21. Truy vấn 5 được lấy ra khỏi hàng đợi và xử lí.

9. Máy chủ kết thúc xử lí truy vấn 5 tại thời điểm 22.

Ví dụ 2:

1. Máy chủ bắt đầu xử lí truy vấn đầu tiên tại thời điểm 2 và kết thúc nó tại thời điểm 10.

2. Tại thời điểm 4, truy vấn thứ 2 xuất hiện và được thêm vào hàng đợi.

3. Tại thời điểm 10, máy chủ vừa mới kết thúc truy vấn đầu tiên, truy vấn thứ 2 được lấy ra khỏi hàng đợi để xử lí. Cùng lúc đó, truy vấn 3 xuất hiện. Lúc này, hàng đợi đã rỗng và truy vấn 3 được thêm vào hàng đợi để xử lí.

4. Tại thời điểm 15, truy vấn 4 xuất hiện nhưng do lúc này máy chủ đang bận xử lí truy vấn 2, nên truy vấn 4 bị từ chối.

5. Tại thời điểm 18, máy chủ kết thúc xử lí truy vấn 2 và tiền hành xử lí truy vấn 3.

6. Máy chủ kết thúc xử lí truy vấn 3 tại thời điểm 27.

Hướng dẫn giải:

Sử dụng hàng đợi để xử lí bài toán này.

Bước 1: Khởi tạo

a) Khởi tạo hàng đợi $Q = \{0\}$, ta sẽ đẩy truy vấn đầu tiên vào hàng đợi.

b) Khởi tạo biến $s = t[0]$, với ý nghĩa s là thời điểm mà ta đang xét đến.

c) Mảng lưu kết quả $res[i] = -1$ với $0 \leq i < n$.

Bước 2: Lặp khi hàng đợi truy vấn khác rỗng

a) Gán $i = front$, giá trị đầu trong hàng đợi, đồng thời lấy giá trị đó ra khỏi hàng đợi. Việc này đồng nghĩa với việc ta sẽ cho máy chủ xử lí truy vấn thứ i.

b) Cập nhật kết quả $res[i] = s + d[i]$, là thời điểm kết thúc quá trình xử lí truy vấn i. Cập nhật lại thời gian hiện tại $s = res[i]$.

c) Tiếp theo ta tìm những truy vấn j xuất hiện trong khi máy chủ đang xử lí truy vấn i, $t[j] < s$. Kiểm tra nếu kích thước hàng đợi nhỏ hơn b cho trước thì đẩy j vào hàng đợi.

d) Ngoài ra, ta phải xét đến trường hợp nếu hàng đợi truy vấn rỗng và ta không có truy vấn j nào xuất hiện trước thời điểm s. Trong trường hợp đó, ta sẽ đẩy truy vấn j đang xét đến vào hàng đợi và nếu $t[j] > s$ thì sẽ cập nhật lại $s = t[j]$.

Độ phức tạp: **O(n)** với n là số lượng truy vấn cần xử lý.