The lazy programmer

Link submit: https://www.spoj.com/problems/LAZYPROG/

Solution:

C++	https://ideone.com/yVGiT9
Java	
Python	

Tóm tắt đề:

Có N hợp đồng và mỗi hợp đồng có một deadline là d_i.

Lập trình viên được giao để hoàn thành các hợp đồng trên được cho là lười biếng vì anh ấy không làm việc nhanh nhất trong khả năng cho phép của mình. Anh ấy cần b_i đơn vị thời gian để hoàn thành hợp đồng thứ i.

Thật may vì anh chàng này rất tham tiền. Nếu giám đốc trả thêm cho anh ấy x_i dollars, anh ấy sẽ hoàn thành hợp đồng thứ i trong $x_i = (b_i - a_i*x_i)$ đơn vị thời gian. Khoản tiền x này sẽ được trả riêng cho từng hợp đồng i để hoàn thành đúng deadline d_i tương ứng trong trường hợp cần. Nếu được trả thêm (b_i/a_i) dollars, anh ấy sẽ hoàn thành hợp đồng i ngay tức khắc.

Giám đốc muốn sắp xếp lại các hợp đồng sao cho tất cả hợp đồng đều được hoàn thành đúng thời hạn và tổng các khoản tiền x phải trả thêm là bé nhất có thể. Bạn hãy giúp giám đốc nhé!

Input:

Dòng đầu tiên là số nguyên t ($1 \le t \le 45$) là số testcases.

Mỗi testcase sẽ được mô tả như sau:

- Dòng đầu chứa số nguyên N ($1 \le N \le 100000$) là số lượng hợp đồng.
- N dòng tiếp theo, mỗi dòng mô tả một hợp đồng, bao gồm ba số nguyên a_i, b_i, d_i ($1 \le a_i, b_i \le 10000$; $1 \le d_i \le 1000000000$).

Ràng buộc: có 90% testcases có $1 \le N \le 10000$.

Output:

Với mỗi testcase, in ra một dòng tương ứng số thực S (làm tròn đến 2 chữ số thập phân), với S là tổng lượng tiền tối thiểu cần trả thêm để tất cả các hợp đồng đều hoàn thành đúng thời hạn.

Ví dụ:

1 2 20 50 100 10 100 50	5.00

Giải thích ví dụ:

Ở ví dụ này chỉ có một testcase, test này ta có hai hợp đồng.

- Hợp đồng thứ nhất có a = 20, b = 50, d = 100. Ta có b = 50 nên chỉ cần 50 đơn vị thời gian là hoàn thành xong hợp đồng này và deadline d = 100. Vậy ta không phải trả thêm bất cứ khoản tiền x nào và hoàn thành sớm hơn cả deadline d.
- Hợp đồng thứ hai có a = 10, b = 100, d = 50. Ta thấy deadline d là 50 nhưng cần đến b = 100 đơn vị thời gian thì hợp đồng này mới được hoàn thiện. Nếu không trả thêm một khoản x dollars thì để rút ngắn thời gian làm việc thì ta sẽ bị trễ deadline. Vậy cần trả thêm một khoảng tiền x, với x = (100 – 50) / 10 = 5.00 dollars, tức là ta chỉ (100 – 10 * 5.00) = 50 đơn vị thời gian để hoàn thành đúng deadline d.

Hướng dẫn giải:

Ta có nhận xét sau: nếu trả thêm x dollars thì công việc thứ i sẽ được hoàn thành trong khoảng thời gian là $b_i – a_i * x$, nói cách khác là công việc sẽ rút ngắn được $a_i * x$ đơn vị thời gian. Vậy ta ưu tiên chọn công việc có a_i lớn hơn, vì với cùng một khoản x dollars bỏ ra, a_i nào càng lớn thì thời gian rút ngắn được càng nhiều.

Giải quyết bài này, ta sẽ thực hiện các bước sau:

Bước 1: chuẩn bị

- Sắp xếp lại các hợp đồng theo deadline d_i tăng dần để đảm bảo deadline d_i nào tới sớm hơn thì phải được thực hiện trước.
- Tạo một priority queue, gọi là pq để lưu thông tin hợp đồng i, ưu tiên theo a. Nói cách khác là xây dựng một heap-max ưu tiên theo a.
- Khởi tạo time = 0, sum_min = 0 (với time dùng để lưu tổng các b_i, sum_min dùng để lưu tổng các khoản trả thêm bé nhất có thể).

Bước 2: xử lí, duyệt qua các hợp đồng i

- Cộng dồn b[i] vào time và thêm thông tin của hợp đồng i vào priority queue pq.
- Trong khi time > d[i], tức là ta cần hạ time xuống dần sao cho time đúng bằng d[i] để hoàn thành hợp đồng i đúng thời hạn:
    - Gọi top là phần tử đầu tiên của pq. Pop phần tử này ra khỏi pq.

- Nếu (top.b > time – d[i]), ta sẽ chọn (time – d[i]) để số tiền x cần chi ra thấp hơn.
  - Cộng thêm một khoản x = (time – d[i]) / top.a.
  - Giảm top.b đi một lượng thời gian là (time – d[i]).
  - Bỏ top lại vào pq.
  - Gán lại time = d[i].
- Ngược lại: nếu (top.b <= time – d[i]), ta sẽ chọn top.b để số tiền x cần bỏ ra thấp hơn.
  - Cộng thêm một khoản x = top.b / top.a vào sum_min.
  - Hạ time xuống một lượng là top.b.
  - Gán top.b = 0.

Bước 3: xuất kết quả, in sum_min làm tròn hai chữ số phần thập phân. Khởi tạo lại time = 0, sum_min = 0 và khởi tạo lại pq rỗng để chuẩn bị cho testcase kế tiếp.

Độ phức tạp: **O(t*NlogN)** với t là số lượng testcases và N là số lượng hợp đồng ứng với mỗi testcase.