Luyện tập thiết kế thuật toán

Ngày 5 tháng 12 năm 2025

Problem 1: Fucs và gánh nặng mưu sinh

Mô tả

Server của Fucs nhận được hàng đợi bao gồm N tác vụ cần xử lý. Vì tài nguyên CPU hữu hạn nên chỉ có thể chạy tuần tự từng tác vụ và một tác vụ khi đã bắt đầu phải chạy tới khi hoàn tất (tính chất Atomic).

Metadata cho mỗi tác vụ

Runtime $r[i]$ — thời gian chạy thực tế của tác vụ i.

Timeout $t[i]$ — sau thời điểm này, tác vụ i nếu chưa hoàn thành sẽ được tính là thất bại.

Profit $p[i]$ — lợi nhuận thu được khi hoàn thành tác vụ i.

Fucs ~~không làm mà đòi có ăn~~ đang bận kéo rank TLAP nên cần bạn đọc giúp tìm cách tối đa hóa lợi nhuận.

INPUT

- Dòng 1: số nguyên dương N.
- N dòng tiếp theo: mỗi dòng chứa ba số nguyên dương r_i, t_i, p_i.

OUTPUT

- Dòng 1: tổng lợi nhuận tối đa.
- Dòng 2: lịch thực thi các tác vụ (liệt kê chỉ số, cách nhau bởi dấu cách).

Ràng buộc

- Subtask 1: $N \leq 9\,696$ và $r_i, t_i, p_i \leq 363\,636$.
- Subtask 2: $N \leq 969\,696$ và $r_i, t_i, p_i \leq 10^{12}$.

Problem 2: Time-Travel Query in Database

Bối cảnh hệ thống

AstraBank vừa hoàn thiện nền tảng quản trị dữ liệu "Time-Orbit", cho phép bộ phận kiểm toán xem lại trạng thái database tại mọi thời điểm trong quá khứ. Đồng hồ nội bộ đếm từ mốc 0 và mọi giao dịch T đều có `start_time[T]` và `commit_time[T]` trên cùng thang đo này. Với mỗi tuple, hệ thống chỉ lưu giao dịch tạo ra nó và giao dịch xóa nó (nếu có), từ đó tái hiện trạng thái bất kỳ.

Dữ liệu log

- Có N giao dịch. Mỗi giao dịch T có `start_time[T]` (thời điểm bắt đầu) và `commit_time[T]` (thời điểm commit). Nếu `commit_time[T] = -1` thì giao dịch bị hủy.
- Có M tuple đã từng tồn tại. Tuple j được tạo bởi giao dịch `create_tx[j]` và có thể bị xóa bởi `delete_tx[j]` (hoặc -1 nếu chưa bị xóa).
- Tuple j tồn tại trong cơ sở dữ liệu sau khi giao dịch tạo nó commit và biến mất khi giao dịch xóa nó commit.
- Các dấu thời gian `start_time` và `commit_time` (khác -1) đều phân biệt nhau trên toàn hệ thống.
- Mọi mốc thời gian là số nguyên không âm, không vượt quá 10^{12} giây.

Nhiệm vụ

- Với mỗi giao dịch T, xác định tập tuple tồn tại tại thời điểm `start_time[T]`.
- Một tuple j tồn tại nếu giao dịch tạo j đã commit trước hoặc đúng thời điểm đó và chưa bị xóa, hoặc giao dịch xóa commit sau thời điểm đó.

Định dạng vào/ra

- **Input**
  - Dòng 1: hai số nguyên N, M.
  - N dòng tiếp: `start_time[T]` và `commit_time[T]` cho từng giao dịch T.
  - M dòng tiếp: `create_tx[j]`, `delete_tx[j]` (-1 nếu chưa bị xóa).
- **Output**
  - N dòng; dòng T in danh sách các tuple tồn tại tại thời điểm `start_time[T]` theo thứ tự tăng dần, định dạng "`k a_1 a_2 ...a_k`".

Ví dụ

```
Input:
3 2
0 3
4 8
9 12
1 3
2 -1

Output:
0
1 1
2 1 2
```

Ràng buộc

- **Subtask 1**: $N, M \leq 9\,696$.
- **Subtask 2**: $N, M \leq 969\,696$.
- Tổng số tuple được in ra trong toàn bộ output không vượt quá $1\,000\,000$.

Problem 3: Operating System Memory Partitioning

Bối cảnh

Trên cụm máy chủ mới, hệ điều hành giám sát vùng nhớ vật lý khổng lồ mô phỏng dưới dạng lưới $H \times W$. Mỗi ô là một frame phần cứng và tại thời điểm kiểm tra có N frame đang bị chiếm dụng. Nhóm vận hành muốn vẽ duy nhất một đường dọc giữa hai cột liên tiếp và một đường ngang giữa hai hàng liên tiếp; chúng cắt nhau tạo bốn khu vực. Chi phí giám sát bằng số tiến trình lớn nhất trong bốn khu vực, cần tối thiểu hóa chi phí này.

Định dạng đầu vào/đầu ra

- **Input**
  - Dòng 1: hai số nguyên H, W.
  - Dòng 2: số nguyên N.
  - N dòng tiếp: mỗi dòng chứa r, c ($1 \le r \le H$, $1 \le c \le W$) — vị trí frame bị chiếm.
- **Output**: In ra giá trị nhỏ nhất của số frame bị chiếm lớn nhất trong bốn vùng sau khi đặt hai đường kẻ tối ưu.

Ràng buộc

- $1 \le H, W \le 10^{12}$.
- **Subtask 1**: $N \le 9\,696$.
- **Subtask 2**: $N \le 363\,636$.
- **Subtask 3**: $N \le 969\,696$.