# Quản lý session đăng nhập của người dùng sử dụng RB-Tree

Red-Black Tree (Cây đỏ-đen) là một cấu trúc dữ liệu **cân bằng** (self-balancing binary search tree) với các quy tắc đặc biệt để đảm bảo hiệu suất ổn định (O(log n) cho tìm kiếm, chèn, xóa). Khác với AVL Tree (cân bằng nghiêm ngặt), RB-Tree cân bằng "lỏng" hơn, giúp **giảm số lần quay cây** khi chèn/xóa, phù hợp với các ứng dụng **cần cân bằng giữa đọc và ghi**.

**Đặc Điểm Của Red-Black Tree**

- **5 quy tắc cơ bản**:
    1. Mỗi nút có màu **đỏ** hoặc **đen**.
    2. Gốc (root) luôn **đen**.
    3. Mọi lá (NIL/null) là **đen**.
    4. Nếu một nút **đỏ**, cả 2 con phải **đen** (không có 2 nút đỏ liên tiếp trên một đường đi).
    5. Mọi đường đi từ gốc đến lá phải có **số nút đen bằng nhau** (đảm bảo cây không quá lệch).
- **Ưu điểm**:
    - Cân bằng tốt hơn BST thông thường, tránh trường hợp suy biến thành **danh sách liên kết** (độ phức tạp O(n)).
    - **Chèn/xóa nhanh hơn AVL Tree** (do ít phải cân bằng lại).
    - Tìm kiếm gần bằng AVL Tree (chậm hơn không đáng kể).
- **Nhược điểm**:
    - Code phức tạp hơn BST thông thường (cần xử lý recoloring và rotation).
    - Tìm kiếm chậm hơn AVL Tree (do cây có thể cao hơn).

RB-Tree được sử dụng rộng rãi trong các **thư viện chuẩn** và **hệ thống yêu cầu hiệu suất ổn định** khi dữ liệu thay đổi liên tục.

Yêu cầu: Cài đặt lại cây RB để minh họa việc quản lý session đăng nhập của người dùng.

Mỗi session sẽ là một nút trong RB-Tree, gồm các thông tin:

- sessionID: chuỗi định danh duy nhất cho session.
- userID: mã người dùng.
- loginTime: thời điểm đăng nhập.
- lastActiveTime: lần cuối tương tác.
- sessionData: dữ liệu lưu kèm theo (nếu cần).

```c
typedef enum { RED, BLACK } Color;

typedef struct SessionNode {
    char sessionID[50];
    char userID[50];
    long loginTime;
    long lastActiveTime;

    Color color;
    struct SessionNode* left, * right, * parent;
} SessionNode;
```

**Nghiệp vụ dựa trên Session ID (trong cookie hoặc token):**

- Khi user đăng nhập thành công, hệ thống tạo một **Session** (sessionID) và gửi cho client.
- Trên mỗi request sau đó, client gửi lại sessionID (qua cookie hoặc header).
- Hệ thống kiểm tra sessionID có tồn tại và hợp lệ hay không.
- Nếu sessionID không còn hợp lệ thì xóa khỏi cây, ngược lại cập nhật lại `lastActiveTime`.

Để kiểm tra xem người dùng hiện có đang đăng nhập không:

- Duyệt toàn bộ cây để tìm userID có session với `lastActiveTime` nhỏ hơn giới hạn cho phép (cách này có thể rất tồi nếu số lượng người dùng lớn).
- Hoặc, tạo ra 1 cây mới để lưu trữ danh sách các user đang online (nên dùng theo cách này).

Chương trình cần xử lý 2 tình huống sau

- Chỉ cho phép đăng nhập tại 1 thiết bị duy nhất.
- Cho phép đăng nhập trên nhiều thiết bị (cần bổ sung thêm thông tin deviceID):
    - Mỗi thiết bị có một session.
    - Khi logout ở một nơi → chỉ xóa session đó.
    - Khi chọn logout ở tất cả các thiết bị → xóa toàn bộ

Dữ liệu đầu vào có thể theo format cho việc chỉ được đăng nhập trên 1 thiết bị

```
08:00:00 bob login device=PC
08:00:05 bob browse page=home
08:00:10 bob search keyword="áo thun"
08:00:20 bob login device=Phone
08:00:22 bob browse page=cart
08:00:30 bob logout
```

Và cho việc đăng nhập trên nhiều thiết bị

```
2025-05-08T09:00:00Z bob 4f59b search "cardiology"
2025-05-08T09:00:03Z bob 0b795 view_profile
2025-05-08T09:00:12Z alice 4b6c8 download_report "report123.pdf"
2025-05-08T09:00:17Z david 7460a search "cardiology"
2025-05-08T09:00:26Z alice 4b6c8 download_report "report123.pdf"
2025-05-08T09:00:30Z alice 4b6c8 download_report "report123.pdf"
2025-05-08T09:00:36Z charlie 43933 update_settings
2025-05-08T09:00:45Z bob 0b795 login success
2025-05-08T09:00:50Z charlie 43933 browse "/home"
2025-05-08T09:00:59Z eva a0e2e login failed
2025-05-08T09:01:05Z charlie 710cb logout
```