**CÂU HỎI TRẮC NGHIỆM**

**1. Treap là một cấu trúc dữ liệu kết hợp giữa?**

A. Cây nhị phân tìm kiếm (BST) và mảng

**B. Cây nhị phân tìm kiếm (BST) và heap**

C. Cây AVL và heap

D. Cây đỏ-đen và mảng

**2. Cây Treap duy trì cấu trúc nào sau đây?**

A. BST về key

B. Heap về priority

C. BST và heap đồng thời

**D. Cả A và B**

**3. Khi nào một cây Treap cần phải quay (rotate)?**

A. Khi cây bị suy biến thành danh sách liên kết

**B. Khi cây không thỏa mãn điều kiện heap**

C. Khi cây không thỏa mãn điều kiện BST

D. Khi cây cần chèn một phần tử mới

**4. Độ phức tạp trung bình của việc chèn một phần tử vào Treap là bao nhiêu?**

A. O(1)

**B. O(log n)**

C. O(n)

D. O(log n) trong trường hợp xấu nhất

**5. Tại sao Treap có thể được coi là một cây nhị phân tìm kiếm ngẫu nhiên?**

A. Vì key được chọn ngẫu nhiên

**B. Vì priority được chọn ngẫu nhiên**

C. Vì mỗi node là ngẫu nhiên

D. Vì sử dụng random pointer

6. Phép chia Treap thành hai cây con dựa trên một giá trị x gọi là gì?

A. Join

B. Rotate

**C. Split**

D. Merge

**7. Điều gì xảy ra nếu priority không được chọn ngẫu nhiên mà luôn cố định theo một hàm của key (ví dụ priority = key)**

A. Treap luôn cân bằng

B. Treap trở thành cây Heap

**C. Treap sẽ không khác gì BST thông thường**

D. Treap sẽ bị lỗi runtime

**8. Giả sử priority được chọn tăng dần theo thời gian, cấu trúc Treap trở thành?**

A. Treap ngẫu nhiên

B. Gần giống Max Heap

**C. Một BST bị lệch như danh sách liên kết**

D. Cây AVL

**9. Khi so sánh Treap với AVL và Red-Black Tree, lợi thế lớn nhất của Treap là?**

**A. Dễ cài đặt hơn do không cần update height/color**

B. Tìm kiếm nhanh hơn

C. Xoay ít hơn

D. Có thể đạt O(1) khi tìm kiếm

**10. Giả sử ta cần thực hiện thao tác split và merge nhiều lần trên một Treap. Cấu trúc nào có thể cải tiến hiệu suất hơn Treap?**

A. Binary Heap

B. Splay Tree

C. Cartesian Tree

**D. Treap có thể mở rộng thêm size và dùng Implicit Treap**

**11. Treap có thể được sử dụng để xây dựng một cấu trúc dữ liệu hỗ trợ thao tác nào dưới đây trong O(log n) thời gian?**

A. Tìm phần tử lớn nhất

B. Tìm phần tử nhỏ nhất

**C. Tìm phần tử có thứ hạng k (k-th smallest element)**

D. Tìm giá trị trung vị theo thời gian thực trong O(1)

**12. Trong một bài toán yêu cầu undo thao tác insert/xóa gần nhất trong O(log n), phương án tốt nhất là:**

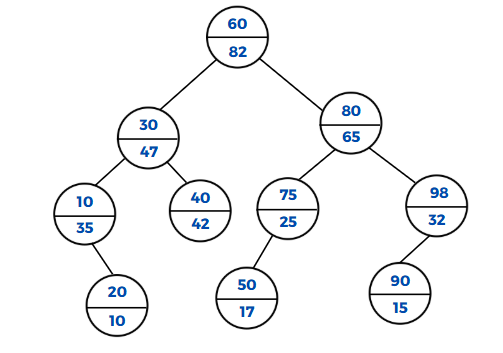
**A. Sử dụng Persistent Treap**

B. Thêm biến cờ để đánh dấu

C. Duyệt lại toàn bộ cây

D. Không thể thực hiện với Treap

13.



**Nếu thêm node (70, 75) thì qua bao nhiều lần thực hiện phép xoay thì cây cân bằng**

A.4

**B.3**

C.6

D.7

**14. Trong Implicit Treap (Treap ẩn), key không được lưu trữ trực tiếp mà được xác định bởi:**

A. Giá trị của priority

**B. Vị trí trong mảng (index)**

C. Tổng các phần tử

D. Giá trị gốc (root)

**15. Tính cân bằng ngẫu nhiên (randomized balancing) của Treap phụ thuộc chủ yếu vào yếu tố nào?**

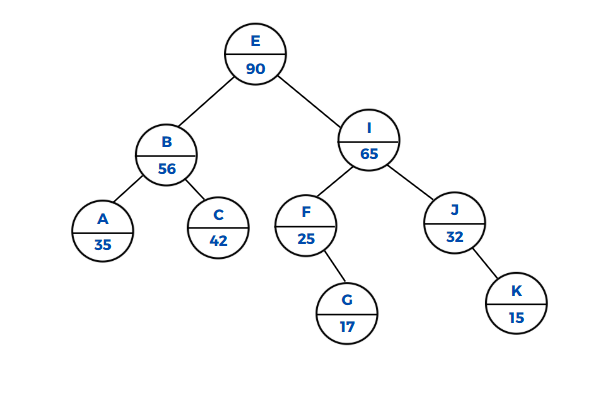
A. Cách sắp xếp các key khi chèn

B. Giá trị của các key

C. Cách cập nhật thứ tự duyệt giữa trái và phải

**D. Việc gán ngẫu nhiên priority cho mỗi node khi chèn**

16.



Nếu thêm node (D, 60) thì có bao nhiêu phép xoay tất cả để cây cân bằng

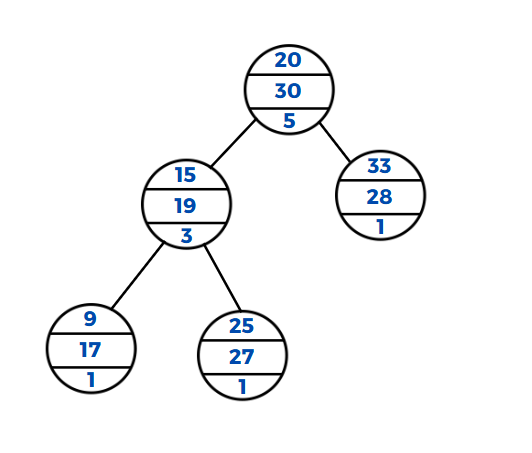
**A.2**

B.5

C.3

D.7

**17.**



**Chèn giá trị 22 tại vị trí index = 2, Split cây hiện tại tại vị trí giữa những node nào?**

**A.(15,19,3) và (20,30,5)**

B.(15,19,3) và (9, 17, 1)

C.(33,28,1) và (20,30,5)

D.(15,19,3) và (25, 27, 1)

**18. Ứng dụng của Treap và các biến thể của Treap là ?**

A. Dùng để quản lý hệ thống tập tin như NTFS

B. Dùng trong các trình biên dịch để tối ưu mã máy

**C. Dùng để thực hiện các thao tác trên mảng như chèn, xóa ở vị trí bất kỳ phân đoạn hiệu quả với độ phức tạp log(n)**

D. Dùng thay thế hoàn toàn các cấu trúc dữ liệu như Heap và Hash Table trong thực tế

**BÀI TẬP**

<https://drive.google.com/drive/u/0/folders/1e8vNzqMa2hKKVEpKB-OJ_yxeVyj9jcn7>