Lời giải [IT003.P21.CTTN.1] Assigment3 (Advance và Basic)

Bảo Quý Định Tân - 24520028

Ngày 8 tháng 4 năm 2025

Mục lục

1 Advance

1.1 Kiểm Kê 2

Đầu tiên ta sẽ sắp xếp theo thứ tự giảm dần theo số lượng từng loại hàng, nếu số lượng bằng nhau thì in theo mã số tăng dần.

Bây giờ các loại giống nhau sẽ nằm cạnh nhau, ta sẽ vừa duyệt vừa duy trì biến đếm các loại giống nhau và xuất ra kết quả khi gặp loại mới. Khi đã duyệt hết mảng, ta sẽ kiểm tra ta đã xuất hết chưa, nếu chưa thì ta xuất nốt loại cuối cùng.

1.2 Binary Search 2

Ta sẽ lưu một mảng mới với kiểu dữ liệu *pair<int, int>*, với số đầu tiên là a_i số thứ hai là i. Sắp xếp mảng mới này lại theo thứ tự a_i tăng dần, cùng a_i thì sắp xếp theo i tăng dần.

Để truy vấn ra vị trí xuất hiện đầu tiên của x, ta sẽ tìm kiếm nhị phân trong mảng mới: phần tử có giá trị là x đầu tiên, và vì ta đã sắp xếp mảng mới này theo thứ tự kể trên, nên giá trị x ta mới tìm ra này sẽ có vị trí i tương ứng trên mảng gốc đầu tiên.

Tương tự, để truy vấn ra vị trí xuất hiện cuối cùng của x, ta sẽ tìm kiếm nhị phân trong mảng mới: phần tử có giá trị x cuối cùng.

Còn nếu ta không tìm được trong mảng mới, phần tử nào có giá trị là x, ta xuất ra -1.

1.3 khangtd.DetectVirusin2D

Ta để ý độ dài chuỗi truy vấn chỉ từ 2 đến 10 và $1 \leq n * m \leq 10^5$ cho nên với mỗi vị trí, ta sẽ chuẩn bị trước được những chuỗi có thể xuất hiện từ nó.

Để thuận tiện trong việc lưu trữ, truy vấn tìm xem một chuỗi có xuất hiện trong một tập nhiều chuỗi hay không, ta sẽ sử dụng **Trie** để lưu trữ những chuỗi có thể tạo ra từ bảng.

1.4 khangtd.XepHang2

Có nhiều cách để cài đặt bài toán này, ở đây ta sử dụng cách cài đặt cơ bản là sử dụng **set** để mô phỏng lại bài toán.

Ta sẽ lưu vào trong **set** một *pair<int, int>* gồm hai thông tin lần lượt là thời gian ra vào và giá trị.

Bởi vì **set** sẽ tự động sắp xếp tăng dần theo first trước và second sau, nên ta sẽ tận dụng để làm bài này.

Ta duy trì một biến thời gian hiện tại, mỗi khi có một sự kiện xảy ra, ta giảm thời gian hiện tại đi 1 và gán nó vào cho người mới.

Ta cũng duy trì một mảng để biết được thời gian hiện tại của mỗi người là gì, để có thể xóa ra khỏi **set**.

1.5 khangtd.XepHang

Bài này ta cũng thực hiện các bước đầu giống bài trên, sau khi thực hiện hết các thao tác, ta sử dụng chính cái **set** đó để in ra.

2 Basic

2.1 Kiểm tra biểu thức toán học (Latex)

Bài này đơn giản ta sẽ sử dụng stack để duy trì các loại dấu ngoặc mở, mỗi lần gặp một dấu ngoặc đóng:

- Nếu stack rỗng: lỗi
- Nếu dấu ngoặc mở ở trên top của stack không tương ứng với dấu ngoặc đóng hiện tại: lỗi
- Nếu khớp thì ta bỏ dấu ngoặc mở ở trên top của stack ra ngoài

Chú ý trường hợp khi ta đã duyệt qua hết chuỗi và thực hiện nhưng chưa bị sai, ta cần phải kiểm tra một lần nữa coi stack có rỗng hay không để biết coi là còn dấu ngoặc mở nào chưa được đóng hay không?

2.2 LinkedList-Insertion

Bài này chỉ là một bài toán cài đặt cơ bản, ta duyệt ta từng phần tử để tìm ra chỗ chèn thích hợp và chèn vào.

2.3 LinkedList-NhapDaThuc

Bài này cũng là một bài toán cài đặt như đề bảo, chỉ cần cẩn thận các trường hợp xuất ra và khi tính giá trị của biểu thức với giá trị x được nhập vào, khi tính lũy thừa, ta sử dụng hàm **pow** trong **c++**.

2.4 LinkedList-Reverse

Bài này cũng là một bài toán cài đặt cơ bản, ta sử dụng ý tưởng ghép khớp từng loại nu khi nhân đôi adn. Ta sẽ duyệt từ đầu danh sách đến cuối danh sách, song song với đó là tạo ra một danh sách mới nhưng thứ tự kết nối lại ngược lại.

2.5 Dec to Bin

Đây cũng là một bài toán cài đặt đơn giản, ở đây ta sử dụng bitset để chuyển từ hệ thập phân sang nhị phân nhanh.

2.6 LinkedList: MergetwoSortedLinkedList

Bài này cũng là một bài toán cài đặt đơn giản, thay vì merge hai mảng trong khi làm merge sort mà ta hay gặp thì bây giờ là merge trên linked list.

2.7 Tree: Preorder Traversal (NLR) - Duyệt cây BST theo NLR

Bài này chỉ là một bài toán cài đặt đơn giản, ta thực hiện duyệt giống như duyệt dfs.

2.8 Tree: Preorder Traversal (NLR) II - Duyệt cây BST theo NLR không đệ quy

Nhưng vì không được sử dụng đệ quy nên ta sẽ sử dụng stack để mô phỏng quá trình duyệt.

2.9 LinkedList: Tìm nút chung của 2 danh sách liên kết đơn

Ở bài toán này vì dữ liệu nhỏ nên ta có thể duyệt $O(n^2)$ để tìm ra 2 node chung của 2 linked list.

2.10 Binary Search Tree: Nút tổ tiên thấp nhất (tiếng Việt)

Ý tưởng tiếp cập của bài toán này là ta sẽ sử dụng thuật toán tìm LCA cơ bản nhất:

- Chọn một trong 2 đỉnh có độ cao lớn hơn và đi lên cho tới khi 2 đỉnh có độ cao bằng nhau.
- Đi lên đồng thời 2 đỉnh cho tới khi 2 đỉnh này gặp nhau.

Và vì không được sử dụng đệ quy nên ta phải dùng stack để mô phỏng quá trình dfs để tính độ cao cho từng node trong cây.