ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Bài tập lớn 2

JJK RESTAURANT OPERATIONS

(Phần 2 - Hồi tưởng)

TP. HỒ CHÍ MINH, THÁNG 11/2023

Đặc Tả Bài Tập Lớn 2

Phiên bản 1.0

1 Chuẩn đầu ra

Sau khi hoàn thành bài tập lớn này, sinh viên sẽ có khả năng:

- Hiện thực cấu trúc dữ liệu dạng cây và làm quen với bảng băm.
- Lựa chọn và vận dụng các cấu trúc dữ liệu phù hợp để đạt được các kết quả mong muốn.

2 Lời ngỏ

Chào các giải thuật sư, do đã quá lâu mà tác giả vẫn chưa cho Gojo xuất hiện lại cho nên cảm hứng để viết phần tiếp theo của chuỗi bài tập lớn này cũng cạn dần. Những thuật thức như *thập chủng đồ thị*, *xích huyết ma trận*,... vì vậy cũng chìm vào quên lãng. Do đó, chúng ta sẽ quay lại với cách xử lý mạch truyện truyền thống khi không biết viết gì tiếp theo chính là **hồi tưởng lại quá khứ**.

Hình 1: Gojo và Sukuna thời mới mở nhà hàng.[1]

Nội dung câu chuyện lần này sẽ xoay quanh việc Gojo và Sukuna thuở mới vào nghề, mỗi người đều sở hữu một nhà hàng riêng, trước khi quyết định cùng nhau hợp tác như trong nội dung bài tập lớn số 1.

3 Giới thiệu

Trong bài tập lớn này, sinh viên sẽ tiếp tục mô phỏng việc xử lý yêu cầu đặt bàn và sắp xếp vị trí chỗ ngồi cho các khách hàng trong hai nhà hàng thông qua các lệnh được mô tả ở phần 3.1.

Ý nghĩa của từ viết tắt và kiểu dữ liệu của các thông số được mô tả như sau:

- **NAME**: một chuỗi ký tự trong bảng chữ cái Alphabet (bao gồm cả chữ viết thường và viết hoa) liên tục không có khoảng trắng, biểu thị tên của khách hàng.
- **NUM**: một số nguyên dương với ý nghĩa khác nhau ứng với từng lệnh xử lý khác nhau. Và ứng với mỗi lệnh thì giá trị NUM này sẽ có các khoảng giá trị khác nhau.
- **MAXSIZE**: là một số nguyên có giá trị lớn hơn 0, biểu thị số lượng khu vực tối đa mà mỗi nhà hàng có thể phục vụ.

Lưu ý: sau khi xử lý xong tất cả các lệnh trong tập tin đầu vào và xuất kết quả ra màn hình, chương trình phải đảm bảo huỷ tất cả các đối tượng dữ liệu được cấp phát động, không để lại rác trong bộ nhớ trước khi kết thúc chương trình.

3.1 Danh sách lệnh

LAPSE <NAME>:
Lệnh này dùng để xác định khách hàng sẽ đến nhà hàng nào dựa vào tên của khách hàng. Quy trình xử lý lệnh này được diễn ra như sau:

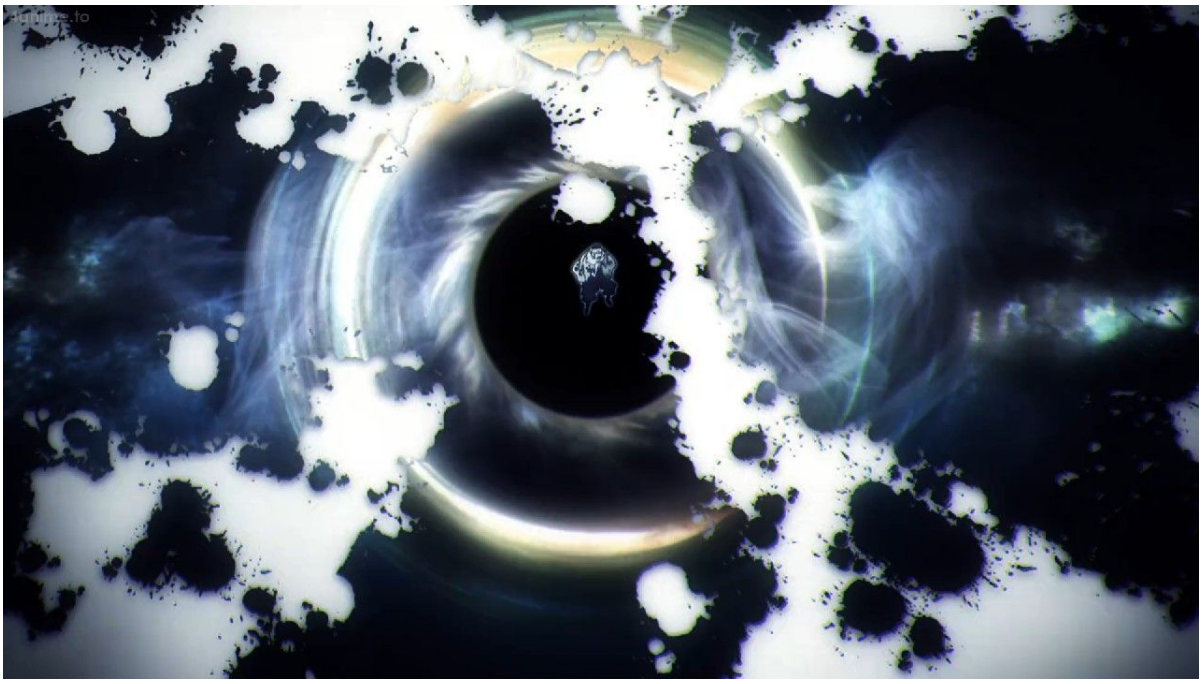
1. Chuẩn hóa tên khách hàng:

1. Đầu tiên, nhân viên sẽ liệt kê tần suất xuất hiện của từng ký tự riêng biệt trong tên của khách hàng (phân biệt chữ hoa và chữ thường). Sau đó sắp xếp lại chúng theo thứ tự tăng dần (nếu có nhiều ký tự với tần suất xuất hiện như nhau thì ký tự nào xuất hiện trước sẽ đứng trước).

   Ví dụ: Với khách hàng có tên **"abaaabbbDd"** thì sau khi xử lý sẽ được danh sách:
   a : 4 $\longrightarrow$ b : 4 $\longrightarrow$ D : 1 $\longrightarrow$ d : 1.

2. Và để tăng cường tính bảo mật thông tin của khách hàng, nhân viên sẽ tiếp tục sử dụng mật mã Caesar để mã hóa từng ký tự trong danh sách vừa tạo. Với số bước dịch chuyển của từng ký tự chính bằng số lần xuất hiện của ký tự đó. Giả sử, kết quả sau khi mã hóa là danh sách X.

3. Dựa vào danh sách X này, bắt đầu xây dựng cây Huffman theo đúng mô tả trong tài liệu hướng dẫn [4] ở chương 5.6. Tuy nhiên, sẽ có một số thay đổi nhất định như sau:
    - Trong quá trình xây dựng cây, khi xảy ra trường hợp các phần tử có cùng giá trị, thì phần tử nào xuất hiện trước sẽ được ưu tiên xét trước.
    - Bên cạnh đó, nhân viên sẽ luôn thực hiện việc xoay cây nếu như phát hiện có trường hợp cây bị mất cân bằng (cho cả cây con trong quá trình xây dựng và cây kết quả). Cơ chế xác định mất cân bằng trên cây và xoay cây dựa theo cách hoạt động của cây AVL. Gọi cây kết quả thu được là cây X.
4. Dựa vào cây X, thực hiện chuyển đổi từng ký tự trong tên của khách hàng (**đã được chuyển đổi theo mã hóa Ceasar**) theo cách tương tự như mô tả trong tài liệu hướng dẫn [4] ở chương 5.6.2, thu được một giá trị theo hệ nhị phân. Lấy tối đa 10 ký tự trong chuỗi nhị phân đó theo chiều từ phải sang trái và chuyển đổi kết quả thu được sang hệ thập phân. Gọi kết quả thu được là Result.
5. Lưu ý, tên của khách hàng phải chứa tối thiểu từ 3 ký tự khác nhau trở lên. Nhân viên sẽ từ chối nhận khách nếu như tên của khách hàng không thỏa điều kiện trên.



Hình 2: Hình ảnh một du khách choáng ngợp trước cảnh quan nhà hàng của Gojo. [2]

2. Chọn nhà hàng:

Mỗi nhà hàng sẽ có cách trang trí khác nhau, trong khi nhà hàng của Gojo (Vô lượng không xứ - nhà G) hướng đến phong cách sang trọng, thanh lịch thì nhà hàng của Sukuna (Phục ma Ngự chùa - nhà S) lại hướng đến phong cách u ám, ma mị.

- Nếu giá trị Result là một giá trị lẽ thì khách sẽ thích ngồi ở nhà G, ngược lại, khách hàng sẽ ngồi ở nhà S. Mỗi nhà hàng đều sẽ có MAXSIZE khu vực.
- Vì cả hai chủ nhà hàng đều thuần thạo trong việc sử dụng bành trướng lãnh địa cho nên họ không giới hạn số lượng khách có thể phục vụ tại mỗi khu vực.

Hình 3: Khung cảnh ma mị tại nhà hàng của Sukuna.[1]

3. Chọn khu:

Sau khi chọn được nhà hàng thì khách hàng sẽ được nhân viên sắp xếp vào các khu vực nhất định. Số thứ tự khu vực (ID) sẽ được tính theo công thức **ID = Result % MAXSIZE + 1** với ID bắt đầu từ 1 đến MAXSIZE. Và mỗi nhà hàng sẽ có cách bố trí chỗ ngồi cho khách hàng của mình khác nhau, cụ thể như sau:

Đối với nhà hàng **Vô lượng không xứ**:

- Gojo chọn sử dụng bảng băm để phân chia khu vực cho khách.
- Tuy nhiên, khi có nhiều khách được bố trí vào cùng một khu vực thì Gojo sử dụng thuật thức Binary Search Tree (BST) với giá trị Result của từng khách sẽ là khóa dùng để xây dựng cây BST. Nếu giá trị thêm vào sau bằng giá trị đã có thì thêm vào bên phải của cây. Node có giá trị nhỏ nhất của cây con bên phải sẽ được dùng để thay thế với node cần xóa khi thực hiện tác vụ xóa một node trong cây BST.

Đối với nhà hàng **Phục ma Ngự chùa**:

- Sukuna sử dụng min-heap để bố trí khu vực cho khách. Mỗi khu vực được xem như một node trong min-heap và có nhãn từ "1" cho đến "MAXSIZE". Khi có một vị khách được sắp xếp ngồi vào hoặc bị đuổi khỏi khu vực nào thì giá trị của node đó sẽ được tăng hoặc giảm đi 1 đơn vị tương ứng. Và chỉ khu vực có khách mới được đưa vào min-heap.
- Ví dụ: Nhà S hiện tại có 6 khu vực (từ 1 đến 6), khi có 3 khách vào khu vực số 1 thì node với nhãn là "1", có giá trị là 3 được đưa vào min-heap. Tiếp đó, nếu có 2 khách vào khu vực số 2 thì node với nhãn "2" được đưa vào min-heap và đồng thời được đưa lên làm root.
- Trường hợp số lượng khách tại một khu vực thay đổi thì nhân viên sẽ thực hiện việc re-heap. Giả sử, gọi số lần mà một khu vực nào đó nhận khách là NUM, khi áp dụng re-heap up (di chuyển một phần tử từ node lá lên node gốc), phần tử con sẽ hoán đổi vị trí với phần tử cha của nó khi giá trị NUM của phần tử con nhỏ hơn so với phần tử cha của nó. Khi áp dụng re-heap down (di chuyển một phần tử xuống node lá), phần tử cha sẽ hoán đổi với phần tử con của nó khi giá trị NUM của phần tử cha lớn hơn. **Trường hợp phần tử cha có hai con, thì phần tử cha sẽ hoán đổi với phần tử con có giá trị lớn hơn. Nếu các phần tử có cùng giá trị NUM thì phần tử lớn hơn được quy ước là phần tử được thêm vào heap sớm hơn**. Kích thước tối đa của min-heap là MAXSIZE.
- Ngoài ra, khi một khu nào đó được thêm vào min-heap nhưng sau một khoảng thời gian, không còn khách nào ngồi ở khu vực đó thì nhân viên sẽ xóa khu vực đó ra khỏi min-heap.

KOKUSEN:
Lệnh này được dùng để nhân viên của nhà G đuổi một số vị khách mà Gojo nghĩ là gián điệp từ nhà S sang để quậy phá nhà hàng của mình.

Quy trình vận hành thuật thức diễn ra như sau:

Hình 4: Gojo khi phát hiện có người quậy phá nhà hàng của mình.[3]

- Đầu tiên, nhân viên sẽ chuyển đổi cây BST (gọi là cây BST gốc) sang một mảng các giá trị theo hậu thứ tự **post-order**. Sau đó, nhân viên sẽ tính toán xem có bao nhiêu hoán vị của mảng vừa tạo mà nếu thêm lần lượt các phần tử đó theo thứ tự chỉ mục từ nhỏ đến lớn, có thể sinh ra cùng một cây BST như cây BST gốc.
- Ví dụ: Giả sử cây BST gốc đang là (2 (1 3)) Với "2" là root, "1" là con trái và "3" là con phải và mảng hiện tại có giá trị (1,3,2). Nhân viên sẽ tìm được 2 hoán vị có thể sinh ra cây BST tương tự như cây gốc từ mảng trên là (2,1,3) và (2,3,1). Với mảng (1,2,3) sẽ cho ra một cây BST khác nên không được tính. Cho nên trường hợp này, kết quả trả về là 2 (giả sử gọi kết quả thu được là Y).
- Sau đó tiến hành xóa Y khách hàng tại khu vực đó theo thứ tự FIFO. Nếu Y $>$ tổng số lượng node trong cây thì xóa hết cây.
- Lặp lại quy trình trên cho tất cả các khu vực có khách còn lại trong nhà G.

KEITEIKEN <NUM>:

Lệnh này được dùng để nhân viên nhà S đuổi NUM khách ở NUM khu vực trong nhà hàng. Với NUM có giá trị từ 1 đến MAXSIZE.

Hình 5: Sukuna khi phát hiện có người quậy phá nhà hàng của mình.[1]

Quy trình vận hành thuật thức diễn ra như sau:

- Nhân viên sẽ chọn khu vực nào chưa được sử dụng lâu nhất và có số lượng khách ít nhất để tiến hành đuổi khách. Ví dụ, nhà S có 3 khu vực,lần lượt có 3 khách vào khu vực số 1, 5 khách vào khu vực số 2 và 3 khách vào khu vực số 3.
- Khi nhân viên nhận được lệnh **KEITEIKEN 1** thì nhân viên sẽ tìm và phát hiện rằng khu vực đang có số lượng khách ít nhất là 1, 3. Tuy nhiên, khu vực 3 vừa có khách vào (vừa được sử dụng) cho nên nhân viên sẽ chọn khu vực 1 và sẽ đuổi 1 khách trong khu vực này (nếu có nhiều khách cần đuổi, nhân viên sẽ đuổi khách theo thứ tự FIFO).
- Nếu NUM lớn hơn số lượng khách trong khu vực thì xem như nhân viên đuổi toàn bộ khách trong khu vực đó.
- Khi lệnh kết thúc, in thông tin của các vị khách bị đuổi đầu tiên đến cuối cùng theo định dạng: "Result-ID/n", với Result là giá trị Result được đề cập ở lệnh **LAPSE** còn ID là số thứ tự của khu vực.

HAND:

In các giá trị trong cây Huffman theo thứ tự từ trên xuống dưới, từ trái qua phải theo định dạng: "char-freq/n" của khách hàng vừa đến nhà hàng gần đây nhất (bao gồm cả nhà G và nhà S). Với char là ký tự (trước khi được mã hóa) và freq là số lần ký tự đó xuất hiện trong tên khách hàng.

LIMITLESS <NUM>:

In ra cây BST tại khu vực NUM của nhà G theo trung thứ tự (in-order) với định dạng "Result/n" tại từng node. Nếu NUM không tồn tại trong nhà G hoặc tại khu vực NUM không có khách thì không cần làm gì cả.

CLEAVE <NUM>:

In thông tin của NUM khách hàng (in theo thứ tự LIFO) của các khu vực được duyệt theo tiền thứ tự (pre-order) trong min-heap với định dạng: "ID-Result/n" với ID là số thứ tự của từng khu vực, Result là giá trị được tính ở lệnh LAPSE. Nếu NUM lớn hơn số lượng khách đang có thì in hết khách trong khu vực.

4 Lời kết

Hình 6: Have a break, Have a Gojo.[2]

5 Yêu cầu

Để hoàn thành bài tập lớn này, sinh viên phải:

- Tải xuống tập tin initial.zip và giải nén nó.
- Sau khi giải nén sẽ được 4 files: main.cpp, main.h, restaurant.cpp và test.txt. Sinh viên **KHÔNG ĐƯỢC** sửa đổi các file main.cpp, main.h.
- Sinh viên được quyền chỉnh sửa bất kỳ nội dung nào trong file restaurant.cpp để hiện thực bài toán.
- Đảm bảo rằng chỉ có một lệnh **include** trong file restaurant.cpp đó là **#include "main.h"**. Ngoài ra, không cho phép có một **include** nào khác trong file này.
- Môi trường dùng để chấm bài là g++ (MinGW-W64 i686-ucrt-posix-dwarf) 11.2.0.

6 Nộp bài

Sinh viên chỉ nộp file: **restaurant.cpp**, trước thời hạn được đưa ra trong đường dẫn "Assignment 2 Submission". Có một số testcase đơn giản được sử dụng để kiểm tra bài làm nhằm đảm bảo rằng kết quả có thể biên dịch và chạy được. Sinh viên có thể nộp bài bao nhiêu lần tùy ý nhưng chỉ có bài nộp cuối cùng được tính điểm. Vì hệ thống không thể chịu tải khi quá nhiều em nộp bài cùng một lúc, vì vậy sinh viên nên nộp bài càng sớm càng tốt. Sinh viên sẽ tự chịu rủi ro nếu nộp bài sát hạn chót. Khi quá thời hạn nộp bài, hệ thống sẽ đóng nên sẽ không thể nộp nữa. Bài nộp qua email không được chấp nhận.

7 Harmony

Trong đề thi có thể có các câu hỏi liên quan đến nội dung bài tập lớn (phần mô tả câu hỏi sẽ được nêu rõ là dùng để harmony cho bài tập lớn, nếu có) . Điểm của các câu hỏi harmony sẽ được scale về thang 10 và sẽ được dùng để tính lại điểm của các bài tập lớn. Cụ thể:

- Gọi x là điểm bài tập lớn.
- Gọi y là điểm của các câu hỏi harmony sau khi scale về thang 10.

Điểm cuối cùng của bài tập lớn sẽ được tính theo công thức trung bình điều hòa sau:

$$\mathbf{Assignment\_Score = 2 * x * y \ / \ (x + y)}$$

8 Xử lý gian lận

Bài tập lớn phải được sinh viên TỰ LÀM. Sinh viên sẽ bị coi là gian lận nếu:

- Có sự giống nhau bất thường giữa mã nguồn của các bài nộp. Trong trường hợp này, TẤT CẢ các bài nộp đều bị coi là gian lận. Do vậy sinh viên phải bảo vệ mã nguồn bài tập lớn của mình.
- Sinh viên không hiểu mã nguồn do chính mình viết, trừ những phần mã được cung cấp sẵn trong chương trình khởi tạo. Sinh viên có thể tham khảo từ bất kỳ nguồn tài liệu nào, tuy nhiên phải đảm bảo rằng mình hiểu rõ ý nghĩa của tất cả những dòng lệnh mà mình viết. Trong trường hợp không hiểu rõ mã nguồn của nơi mình tham khảo, sinh viên được đặc biệt cảnh báo là KHÔNG ĐƯỢC sử dụng mã nguồn này; thay vào đó nên sử dụng những gì đã được học để viết chương trình.
- Nộp nhầm bài của sinh viên khác trên tài khoản cá nhân của mình.
- Nếu vi phạm các ràng buộc được mô tả ở mục 5, sinh viên sẽ nhận điểm 0.

Trong trường hợp bị kết luận là gian lận, sinh viên sẽ bị xử lý theo kết luận của Hội đồng giảng viên giảng dạy môn học này.

KHÔNG CHẤP NHẬN BẤT KỲ GIẢI THÍCH NÀO VÀ KHÔNG CÓ BẤT KỲ NGOẠI LỆ NÀO!

Tài liệu tham khảo

[1] https://www.deviantart.com/

[2] https://www.pxfuel.com/

[3] https://www.wallpaperflare.com/

[4] Data Structures & Algorithm Analysis by Clifford A. Shaffer - Edition 3.2 (C++ Version)

——————————HẾT——————————