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

RESTAURANT OPERATIONS
(Phần 2)

TP. HỒ CHÍ MINH, THÁNG 04/2023

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 hashtable.
- Chọn lựa và vận dụng các cấu trúc dữ liệu phù hợp để đạt được các kết quả mong muốn.

2 Giới thiệu

Trong bài tập lớn này, các em sẽ mô phỏng việc xử lý yêu cầu đặt bàn và sắp xếp vị trí chỗ ngồi cho khách hàng trong một nhà hàng thông qua các lệnh được mô tả ở phần 2.2. Quy trình vận hành tại nhà hàng như sau:

1. Đầu tiên, khách hàng có thể gọi điện đặt bàn trước hoặc đến trực tiếp nhà hàng để nhân viên sắp xếp chỗ ngồi **dựa vào tên của khách**.
2. Khách hàng khi ngồi vào bàn sẽ có thể gọi món. Dựa vào số lượng món, thứ tự đến và thứ tự gọi món mà khách hàng có thể ở lại dùng bữa lâu hơn hoặc phải nhường bàn cho các vị khách khác và ra về.

2.1 Hướng dẫn chung

Mỗi một yêu cầu từ khách hàng, nhân viên hay chủ nhà hàng được biểu thị thông qua các lệnh xử lý dữ liệu. Mỗi lệnh là một dòng trên tập tin thông tin (từ in đậm trong mô tả), bắt đầu bằng một từ khoá và theo sau là các thông số (từ đặt trong dấu $<$ và $>$). Giữa lệnh và các thông số cách nhau đúng một khoảng trắng. Không có khoảng trắng nào trước từ khoá lệnh và cũng không có khoảng trắng nào đi sau thông số cuối cùng. Các thông số bắt buộc có trong lệnh sẽ được đặt trong dấu $<$ và $>$. Khi một lệnh không cung cấp đúng số thông số hoặc không có kiểu thông số đúng như mô tả hoặc có các khoảng trắng không như mô tả thì lệnh sẽ không đươc xử lý và sẽ được bỏ qua. Ngược lại, các lệnh sẽ được xử lý theo mô tả trong mục 2.2.

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.

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.
- **ID**: số thứ tự của bàn, có giá trị từ 1 cho đến MAXSIZE.
- **MAXSIZE**: số lượng bàn tối đa mà nhà hàng có thể phục vụ, được khai báo sẵn, luôn có giá trị chẵn và lớn hơn 20.

2.2 Danh sách lệnh

REG <NAME>:
Lệnh này dùng để đặt bàn cho khách theo thông tin mà khách hàng cung cấp. Quy trình xử lý lệnh đặt bàn được diễn ra như sau:

1. Chuẩn hóa tên khách hàng:

- Tên của khách sẽ được mã hóa dựa trên mã hóa Huffman. Chuyển đổi kết quả sau khi mã hóa để được một giá trị hệ nhị phân (chỉ lấy tối đa 15 số tính từ bên phải sang nếu kết quả có nhiều hơn 15 số). Sau đó, chuyển đổi giá trị thuộc hệ nhị phân trên sang hệ thập phân và được kết quả **Result**.

2. Chọn khu vực:

- Nhà hàng sẽ bố trí chỗ ngồi cho khách theo hai khu, khu thứ nhất hướng xuống biển (gọi là khu 1) và khu thứ hai hướng lên núi (gọi là khu 2). Nếu giá trị Result là một giá trị lẽ thì sẽ được bố trí ngồi ở khu 1, ngược lại, khách hàng sẽ ngồi ở khu 2. Mỗi khu đều có sức chứa tối đa là MAXSIZE/2, với MAXSIZE là số khách hàng tối đa mà nhà hàng có thể phục vụ.
- Trường hợp nếu một trong hai khu vực đã đầy khách thì sẽ bố trí khách qua khu còn lại bất kể giá trị Result là chẵn hay lẽ.

3. Chọn bàn:

- Sau khi chọn được khu vực thì khách hàng sẽ được nhân viên sắp xếp ngồi vào bàn. Số bàn (ID) sẽ được tính theo công thức ID = Result % MAXSIZE (CT1).

- Trường hợp bàn ứng với ID được tính theo công thức CT1 đã có một vị khách khác ngồi, tuy nhiên nhà hàng vẫn còn bàn trống. Thì nhân viên sẽ ưu tiên chọn bàn cho khách với ID lớn hơn ID hiện tại trước. Sau đó nếu ID không thể tăng được nữa (ID = MAXTABLE) thì sẽ quay lại xem xét tiếp từ bàn đầu tiên (ID = 1). Ví dụ nhà hàng có 5 bàn được đánh số từ 1 đến 5 và giá trị ID là 3, nhưng bàn này đã có khách đặt. Thì nhân viên sẽ ưu tiên chọn bàn cho khách theo thứ tự: $4 \longrightarrow 5 \longrightarrow 1 \longrightarrow 2$.

4. Cách bố trí bàn tại mỗi khu vực:

- Đối với khu vực 1: Một hashtable sẽ được triển khai với chỉ mục của hashtable ứng với từng khách hàng là result % 3. Trường hợp xảy ra đụng độ, sử dụng Linear Probing để giải quyết.
- Đối với khu vực 2: Một cây AVL sẽ được triển khai, giá trị Result của từng khách sẽ là khóa dùng để xây dựng cây AVL. Nếu giá trị thêm vào sau bằng giá trị đã có thì thêm vào bên phải của cây.

5. Khách đến nhà hàng nhưng nhà hàng đã hết bàn:

- Trường hợp nếu như khách hàng đến nhà hàng nhưng nhà hàng đã kín bàn thì khách hàng sẽ được nhân viên bố trí tại cái bàn được xác định thông qua ba cơ chế FIFO, LRCO và LFCO. Cụ thể:
- **FIFO (First In First Out)**: Chọn khách hàng vào nhà hàng sớm nhất để mời về và thay bằng khách hàng vừa mới vào.
- **LRCO (Least Recently Customer Order)**: Chọn khách hàng thực hiện việc gọi món trễ nhất. Ví dụ: A, B, C vào nhà hàng và order theo thứ tự A, B, C, B, C, và A thì B sẽ được mời về và thay bằng khách hàng vừa mới vào.
- **LFCO (Least Frequently Customer Order)**: Chọn khách hàng vào nhà hàng sớm nhất nhưng gọi món ít nhất. Một min-heap phải được sử dụng để sắp xếp các khách hàng dựa trên số lần khách hàng gọi món. Gọi số lần khách hàng gọi món 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 hoặc bằng so với phần tử con. Trường hợp phần tử cha có hai con, thì phần tử cha sẽ hoán đổi với phần tử con có giá trị nhỏ nhất. Nếu giá trị NUM của hai phần tử con bằng nhau, thì phần tử con nhỏ nhất được quy ước là phần tử con bên trái. Kích thước tối đa của min-heap là MAXSIZE.

Việc chọn bàn theo cơ chế nào sẽ dựa vào giá trị OPT được tính theo công thức OPT = Result % 3. Cụ thể:

- OPT = 0 thì nhân viên sẽ bố trí vị trí cho khách theo cơ chế FIFO
- OPT = 1 thì nhân viên sẽ bố trí vị trí cho khách theo cơ chế LRCO
- OPT = 2 thì nhân viên sẽ bố trí vị trí cho khách theo cơ chế LFCO
- Khi thay thế một khách cũ bằng một khách mới thì nhân viên sẽ tiến hành cập nhật lại thông tin ở các khu vực tương ứng.

Giả sử, mỗi lần khách hàng gọi món chỉ có thể gọi được một món. Cú pháp của lệnh gọi món giống với lệnh đặt bàn là **REG <NAME>**, với NAME là tên của khách muốn gọi món.

CLE <NUM>:
Chủ nhà hàng có thể đuổi một vị khách bất kỳ nếu muốn. Với NUM là giá trị biểu thị cho ID của bàn và nếu NUM ứng với bàn đang trống thì bỏ qua lệnh này. Trường hợp NUM < 1 thì đuổi hết khách ở khu 1, NUM $>$ MAXSIZE thì đuổi hết khách ở khu 2 và nếu các khu tương ứng đều không có khách thì bỏ qua lệnh này. Lưu ý rằng lệnh CLE này không liên quan đến cơ chế thay thế được mô tả trong lệnh REG.

PrintHT:
In các giá trị theo chỉ mục từ nhỏ đến lớn của HashTable theo cú pháp: "ID-Result/n" với ID là số thứ tự của bàn và Result là giá trị được đề cập ở lệnh REG. Trường hợp nếu tại chỉ mục chưa có dữ liệu thì bỏ qua việc in thông tin tại chỉ mục đó.

PrintAVL:
In các giá trị trong cây AVL theo thứ tự từ trên xuống dưới, từ trái qua phải theo cú pháp: "ID-Result/n" với ID là số thứ tự của bàn và Result là giá trị được đề cập ở lệnh REG.

PrintMH:
In các giá trị trong Min-heap theo tiền thứ tự (preOrder) theo cú pháp: "ID-NUM/n" với ID là số thứ tự của bàn và NUM là giá trị được đề cập ở lệnh REG tại cơ chế LFCO.

2.3 Yêu cầu

Để hoàn thành bài tập lớn này, các em 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 trong file restaurant.cpp để hiện thực bài toán. Tuy nhiên, không được thay đổi prototype của hàm simulate.
- Đảm bảo rằng chỉ có một lệnh **include** trong file **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

3 Nộp bài

Các em 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 của các em nhằm đảm bảo rằng kết quả của em có thể biên dịch và chạy được. Các em 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 em nên nộp bài càng sớm càng tốt. Các em sẽ tự chịu rủi ro nếu nộp bài sát hạn chót. Khi quá thời hạn nộp bài, hệ thống sẽ đóng nên các em sẽ không thể nộp nữa. Bài nộp qua email không được chấp nhận.

4 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:

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

5 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.

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!

——————————HẾT——————————