Cấu Trúc Dữ Liệu Và Giải Thuật

Bài tập lớn 2

JJK RESTAURANT OPERATIONS (Phần 2 - Hồi tưởng)

nhóm thảo luận Code
https://www.facebook.com/groups/211867931379013

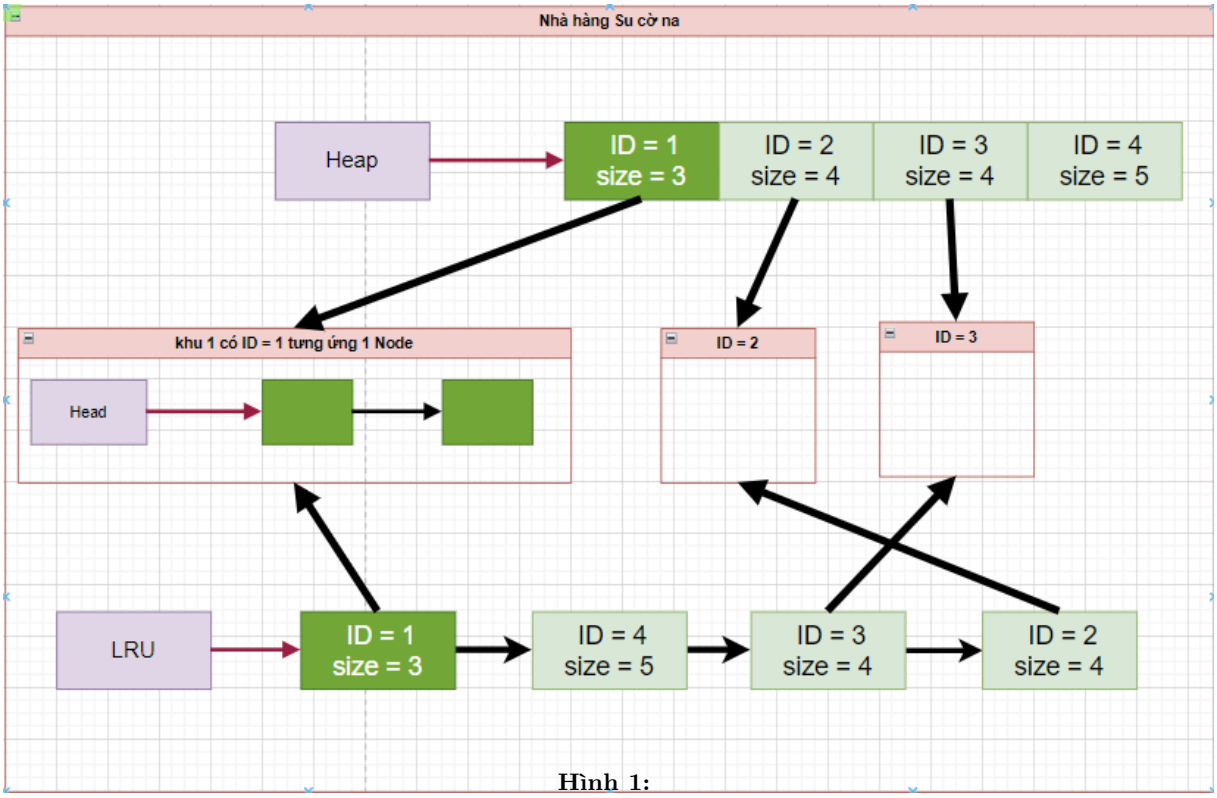
Tp. Hồ Chí Minh, Tháng 11/2023

Mục lục

1 Nhà hàng Sukuna

1.1 Class RESTAURANT Sukuna

- **Cấu trúc:** Nhà hàng của Su cờ na có dạng là nhiều khu được đưa vào *heap* để quản lý với trong một khu lại lại quản lí bằng *LinkList* ý tưởng như hình bên dưới



Hình 1:
Cấu trúc lưu trữ nhà hàng Su cờ na

1. với *heap* sẽ lưu trữ dưới dạng *vector(array)* (cũng là mảng mà thôi) với kích thước được thay đổi theo số lượng khu bực phục vụ nên khu nào có khách thì mới được theo dõi
2. đối với các khu được theo dõi thì ta theo dõi luôn khu vực nào mới có khách hàng vào nhất này gọi là cơ chế **Least Recently Used** nhằm mục đích để đuổi các khu nào mà khách ít vào nhất trong một khoảng thời gian gần đây
3. *LRU* (**Least Recently Used**) sử dụng dưới danh sách liên kết khu nào có khách vừa đến thì ở *head* danh sách.
4. Bên trong mỗi *Node* heap là một danh sách liên kết có chứa *ID* của khu và khách hàng dạng danh sách liên kết với khách nào mới vô thì thêm ở đầu
5. **Thư viện vector** được dùng như một *array* được cấp phát động.
6. **Thư viện Danh sách liên kết**.
7. **Thư viện iterator**.

- **Code:**

```
class RESTAURANT_Sukuna{
        class Node;
private:
        vector<Node* > areaTable;  //! nơi lưu trữ các khu vực
        list<Node* > LRU; //!  Least Recently Used này là cơ chế khu vực nào có khác vào nhà hàng lâu nhất
private:
    void ReHeap_down();
    void ReHeap_up();
    void moveTop();
    void removeNode();
public:
    void insertAreaTable();
    void remove_KEITEIKEN();
    void print_LIMITLESS();
private:
        class Node{
        private:
                int ID;
                list<int> head;
                friend class RESTAURANT_Sukuna;
        public:
                Node(int ID) : ID(ID) {}
                int size() const { return head.size(); }
                //* thêm vô đầu danh sách
                void insert(int result);
                //* xóa ở cuối với số lượng là number cơ chế FIFO vô sớm thì cút sớm
                void remove(int number);
                //* print ra number khách hàng mới đến gần nhất theo cơ chế LIFO các khách hàng gần nhất
                void print(int number);
        };

};

```

1. **class Node :** lưu trữ như 1 node của heap bên trong gồm có danh sách liên kết
   - **ID :** tên khu vực mà *node* đó đang quản lí
   - **head:** lưu trữ danh sách liên kết
   - **hàm size():** lấy ra số lượng khách trong khu đó
   - **hàm insert():** thêm khách hàng vào đầu danh sách
   - **hàm remove(int number):** xóa number khách hàng ra khỏi danh sách
   - **hàm print(int number):** in ra number khách hàng mới đến gần nhất
2. **Các hàm:** nói kĩ hơn ở phần sau.

1.2 LAPSE

Hàm này sẽ thêm khách hàng vào nhà hàng của Sukana.

1.2.1 insertAreaTable trong class RESTAURANT_Sukuna

- **Mô tả:** Với đầu vào là *result* xử lí ở bước phần trước, tìm ra *ID* thông qua công thức thầy cho **ID = result % MAXSIZE + 1;** kiểm tra xem *ID* này đã lưu trữ trong *heap* hay chưa nếu chưa thì tiếng hành thêm khu vực nào vào cuối *heap* (cuối danh sách) sau đó ta cần gọi hàm *insert* của class *Node* và tiếng hành cập nhật danh sách *LRU* thông qua hàm *moveTop* nhằm thêm vào danh sách nếu *ID* chưa tồn tại hoặc là dịch *node* đó lên đầu danh sách.
- **Code:**

```
void insertAreaTable(int result)
{
    int ID = result % MAXSIZE + 1;
    //*bước 1: kiểm tra xem khu vực đó có trong bàn hay chưa lưu vị trí heap
     ↪ vào index
    int index = -1;
    //TODO TODO TODO TODO TODO bước 1

    //*bước 2: xem thử có khu này trong heap chưa để thêm vô
    if(index == -1){
        areaTable.push_back(new Node(ID));
        index = areaTable.size() - 1;
    }

    //*bước 3: thêm khách hàng mới vào khu khách hàng muốn thêm vào và tiến hàn
     ↪ reheap down bàn này xuống vì có số khách đông hơn
    areaTable[index]->insert(result);
    this->moveTop(areaTable[index]);
    this->ReHeap_down(index);
}
```

1.2.2 insert trong class Node

Hàm này thêm khách hàng vào thôi nếu bỏ nó thêm trực tiếp vô class su cờ na cũng được nói chung là ezzz .

1.2.3 moveTop

- **Mô tả:** Có 2 trường hợp là nếu tìm không thấy thì nó chưa tồn tại cần thêm nó ở đầu, trường hợp 2 là tìm thấy thì đưa *Node* đó lên đầu
- **Chú ý:** nên dùng **Thư viện iterator** hoặc có thể dùng $for(i : LRU)$
- **Code:**

```
//* dịch node nó lên đầu danh sách
void moveTop(Node* node)
{
    //TODO: BƯỚC 1 Tìm vị trí của node trong danh sách
    //TODO: BƯỚC 2 nếu nó tồn tại thì dịch nó lên đầu danh sách, nếu không thì
     ↪ insert ở đầu danh sách
}
```

1.2.4 ReHeap_down

- **Mô tả:** reheap thôi chắc các bạn cũng đã học rồi  .
- **Chú ý:**
  - với giá trị xét là **areaTable[index].size()** giống như *int* để các bạn so sánh vậy nhưng hàm này so sánh *size* tương ứng với số lượng khác
  - này là *min–heap* nên **areaTable[index].size()** nào bé hơn thì ở trên này chắc dễ rồi *min–heap* thôi
  - nếu 2 thằng bằng nhau thì chọn ra khu nào có khác vào gần đây nhất sử dùng **list<Node* > LRU**; này đang lưu từ mới nhất đến sớm nhất các khu đang phục vụ khách nên chỉ cần dùng **Thư viện iterator** để so sánh gợi ý cách này thôi có thể dùng *for*

1.3 KEITEIKEN

Hàm này sẽ đuổi khách hàng nhà hàng Su cờ na.

1.3.1 remove_KEITEIKEN trong class RESTAURANT_Sukuna

- **Mô tả:** xóa *number* xóa number khu có số lượng khách hàng ít nhất với mỗi khu thì cần xóa tiếp *number* khách hàng
- **Code:**

```
void remove_KEITEIKEN(int number)
{
    if(areaTable.size() <= 0) return;

    //* đuổi num khách hàng tại num khu vực
    int numberRemove = number;
    while(areaTable.size() != 0 && number != 0){
        //* lấy ra khu đang ở đầu đầu heap xóa number khách hàng đầu linklist
        solution << "remove customers in the area ID = " << areaTable[0]->ID <<
        ↪  ": ";
        areaTable[0]->remove(numberRemove);
        solution << "\n";

        //* trường hợp xóa hết thì xóa nó trong heap sau đó reheap down khác
        ↪  hàng xuống vì đang ở đầu hàng
        if(areaTable[0]->size() == 0)
        {
            swap(areaTable[0], areaTable[areaTable.size() - 1]);
            //! xóa nó khỏi danh sách liên kết
            this->removeNode(areaTable[areaTable.size() - 1]);
            delete areaTable[areaTable.size() - 1];

            //! xóa trong heap nữa
            areaTable.pop_back();
        }
        this->ReHeap_down(0);
        number --;
    }
}
```

- **Hướng giải quyết theo các bước sau:** yêu cầu làm **Bước 1**

Bước 1: lấy ra khách hàng đầu heap xóa number khách hàng đầu linklist

Bước 2: trường hợp xóa hết thì xóa khu đó trong heap sau đó reheap down khu mới xuống vì đang ở đầu hàng này cơ chế hoán đổi phần tử đầu và cuối trong *Heap* thôi

Bước 3: Tiên hành xóa *Node* đó trong *LRU* nếu đã xóa nó trong *heap*

1.3.2 remove trong class Node

- **Mô tả:** Hàm này xóa number phần tử trong danh sách liên kết thôi theo cơ chết *FIFO*.
- **Chú ý:** thêm **solution** để in ra để có thể kiểm tra test và sử dụng các hàm của danh sách liên kết mà làm nha **Thư viện Danh sách liên kết**.
- **Code:**

```
void remove(int number)
{
    //TODO: xóa number khác hàng ở cuối danh sách tương ứng với vô sớm nhất
    //^ gợi ý dùng hàm của linklist có sẵn
    //* thêm solution << head.back() << " "; để in ra
}
```

1.3.3 removeNode

- **Mô tả:** Hàm này dùng để xóa *Node* ra khỏi danh sách *LRU* vì không quản lí khu vực này nữa
- **Chú ý:** không cần dùng *delete* nha đang xóa ở phía sau gọi hàm đó, nên dùng **Thư viện iterator**

```
//* xóa một node ra khỏi danh sách liên kết không gần gọi delete nha vì đã dùng
 → bên dưới hàm xóa
void removeNode(Node* node)
{
    //TODO:
}
```

1.4 CLEAVE

- **Mô tả:** hàm này gọi theo thứ tự *print_CLEAVE* trong class *RESTAURANT_Sukuna* gọi tới *print_pre_order* trong để tiến hành in theo cơ chế tiền thứ tự, và sau đó gọi hàm *print* của từng node in danh sách theo cơ chết *LIFO*
- **Code:**

```
//* in inoder cho heap
void print_pre_order(int index, int number)
{
    if(index >= this->areaTable.size()) return;

    this->areaTable[index]->print(number);
    print_pre_order(index * 2 + 1, number);
    print_pre_order(index * 2 + 2, number);
}
void print_CLEAVE(int number){print_pre_order(0, number);}

//* class Node
void print(int number)
{
    for(list<int>::iterator it = head.begin(); number > 0 && it != head.end();
     ↪  ++it, --number)
    {
        solution << *it << "-" << ID << "\n;
    }
}
```

1.5 Test case

Test	Mô tả	Lỗi
201	insert một node và print rỗng và print 1 node	in 1 phần tử và insert cập nhật *head* và *LRU*
202	insert nhiều phần tử chỉ 1 khu	kiểm tra *ID* bị sai hay *insert*
203	insert nhiều phần tử nhiều khu	kiểm tra *ID* bị sai
204	insert nhiều phần tử nhiều khu	kiểm tra *ID* bị sai
205	insert nhiều phần tử chỉ 1 khu có result bằng	kiểm tra insert *head*
206	reomve một phần tử	lỗi reomve và không chịu print ra theo gọi ý anh
207	reomve một phần tử danh sách nhiều phần tử	lỗi remove
208	reomve 2 phần tử danh sách nhiều phần tử	lỗi remove
209	reomve - insert - remove	lỗi remove và insert
210	nhiều khu nhiều phần tử	lỗi reheap
211-300	random	random

Bảng 1: *test phần nhà hàng Sukuna*

nhóm thảo luận Code

https://www.facebook.com/groups/211867931379013

CHÚC CÁC EM HỌC TỐT