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 Final ngày 13/12

1.1 Gojo

- Chỉnh phần đầu tiên **number = number % MAXSIZE** => yêu cầu dùng công thức nhân Ấn Độ là $(a*b)\%c = ((a\%c)*(b\%c))\%c$ xử lí trong hàm *dfs*
- bỏ dòng **if(this->size() == 1) return;**

1.2 Sukuna

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

    //* TẠO ra heap mới sao chép từ heap cũ
    vector<Node* > areaTableNew(areaTable.begin(), areaTable.end());
    queue<Node* > listDelete; //! danh sách các khu cần xóa
    for(int i = 0; i < areaTable.size() && i < number; i++)
    {
        //* lấy ra phần tử đầu tiên trong heap
        Node* nodeDelete = areaTable[0];
        swap(areaTable[0], areaTable[areaTable.size() - 1]);
        areaTable.pop_back();
        this->ReHeap_down(0);


        //* đưa vào danh sách cần xóa
        listDelete.push(nodeDelete);
    }

    //* trả lại heap
    areaTable = areaTableNew;

    //* đuổi num khách hàng tại num khu vực
    while(listDelete.size()){
        //* lấy ra khu đang ở đầu đầu heap xóa number khách hàng đầu linklist
        Node* nodeDelete = listDelete.front();
        listDelete.pop();

        solution << "remove customers in the area ID = " << nodeDelete->ID <<"(len="
        ↪  <<nodeDelete->head.size()<< ")" << ": ";
        nodeDelete->remove(number);
        solution << "\n";

        //* tìm vị trí của nó trong heap
        int index = 0;
        while(areaTable[index] !=  nodeDelete) index++;

        //* trường hợp xóa hết thì xóa nó trong heap sau đó reheap down khu xuống vì
        ↪  đang ở đầu hàng
        if(nodeDelete->size() == 0)
        {
            swap(areaTable[index], 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(index);
    }
}
```

Bước 1: Tạo ra một bản sao *areaTableNew*

Bước 2: thêm các phần tử vào *listDelete* lấy đầu *heap* ra hiện đang là phần tử nhỏ nhất sau đó xóa nó (gồm reheap)

Bước 3: *areaTable* được gán lại ban đầu

Bước 4: *listDelete* lấy ra từng phần tử rồi xóa nó

Bước 5: nếu phần tử *listDelete* mà không còn khách thì xóa nó luôn đi

Bước 6: reheap lại phần tử ban đầu trong *areaTable*

1.3 Huff

- mỗi lần đếm LR, RL, LL, RR đều $count++$
- chỉnh lại *balanceTree* mỗi khi $count++$ thì xây dựng lại tối đa 3 lần

```
Node* balanceTree(Node* node, int& count)
{
    if(node == nullptr  || count >= 3) return node;
    node = balanceNode(node, count);
    node = balanceNode(node, count);
    node = balanceNode(node, count);
    node->left = balanceTree(node->left, count);
    node->right = balanceTree(node->right, count);
    return node;
}
```

1.4 Hàm hủy

Đọc code đi nha ezz thôi, trong code có đó nhớ thêm vô không thì rác đó, nếu thêm vô không chạy được thì comment nó lại đi trừ ít điểm lắm

nhóm thảo luận Code

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

CHÚC CÁC EM HỌC TỐT