【门徒计划】线程池与任务队列

链表复习题

Leetcode-86-分隔链表

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* partition(ListNode* head, int x) {
        ListNode r1,r2;
        ListNode *p1=&r1,*p2=&r2;
        ListNode *p=head,*q;

        while(p){
            q=p->next;
            // big ? small
            if(p->val < x){
                p->next = p1->next;
                p1->next = p;
                p1 = p;
```

```
            } else {
                p->next = p2->next;
                p2->next = p;
                p2 = p;
            }
            p=q;
        }
        p1->next = r2.next;
        return r1.next;
    }
};
```

Leetcode-138-复制带随机指针的链表

```
/*
// Definition for a Node.
class Node {
public:
    int val;
    Node* next;
    Node* random;

    Node(int _val) {
        val = _val;
        next = NULL;
        random = NULL;
    }
};
*/

class Solution {
public:
    Node* copyRandomList(Node* head) {
        if(head==nullptr){
            return nullptr;
        }
        Node *p = head;
        Node *new_head;
        while(p){
            //copy node
            Node *q = new Node(p->val);
            q->random = p->random;
            q->next = p->next;
            p->next = q;
            p = q->next;
        }
        // 回到头节点,修正
```

```
        p = head;
        while(p) {
            // p1 -> q1 -> p2 -> q2
            // q1 : p1->next
            // q1.random = p1.random.next
            if(p->random) p->next->random = p->random->next;
            (p = p->next) && (p = p->next);
        }
        // 拆链表
        new_head = head->next;
        p = head;
        while(p){
            Node* q = p->next;
            p->next = q->next;
            if(p->next) q->next = p->next->next;
            p=p->next;
        }
        return new_head;
    }
};
```

队列的封装与使用

Leetcode-622-设计循环队列

```
class MyCircularQueue {
public:
    vector<int> arr;
    int head,tail,count;

    MyCircularQueue(int k):arr(k),head(0),tail(0),count(0) {}

    bool enQueue(int value) {
        if(isFull()){
            return false;
        }
        arr[tail]=value;
        tail++;
        //if tail> arr.size()
        tail = tail % arr.size();
        count ++;
        return true;
    }

    bool deQueue() {
        if(isEmpty())return false;
```

```
        head++;
        head=head%arr.size();
        count--;
        return true;
    }

    int Front() {
        if(isEmpty())return -1;
        return arr[head];
    }

    int Rear() {
        if(isEmpty())return -1;
        int real_tail=(tail-1+arr.size()) % arr.size();
        return arr[real_tail];
    }

    bool isEmpty() {
        return (count==0);
    }

    bool isFull() {
        //if(count == arr.size())return true;
        //return false;
        return count == arr.size();
    }
};

/**
 * Your MyCircularQueue object will be instantiated and called as such:
 * MyCircularQueue* obj = new MyCircularQueue(k);
 * bool param_1 = obj->enQueue(value);
 * bool param_2 = obj->deQueue();
 * int param_3 = obj->Front();
 * int param_4 = obj->Rear();
 * bool param_5 = obj->isEmpty();
 * bool param_6 = obj->isFull();
 */

```

Leetcode-641-设计循环双端队列

```
class MyCircularDeque {
public:
    vector<int> arr;
    int head,tail,count;
    MyCircularDeque(int k):arr(k),head(0),tail(0),count(0) {}
```

```
6
7       bool insertFront(int value) {
8           if(isFull())return false;
9           head = (head -1 + arr.size()) % arr.size();
10          arr[head] = value;
11          count++;
12          return true;
13      }
14
15      bool insertLast(int value) {
16          if(isFull()){
17              return false;
18          }
19          arr[tail]=value;
20          tail = (tail+1) % arr.size();
21          count ++;
22          return true;
23      }
24
25      bool deleteFront() {
26          if(isEmpty())return false;
27          head=(head+1)%arr.size();
28          count--;
29          return true;
30      }
31
32      bool deleteLast() {
33          if(isEmpty())return false;
34          tail = (tail-1 + arr.size()) % arr.size();
35          count--;
36          return true;
37      }
38
39      int getFront() {
40          if(isEmpty())return -1;
41          return arr[head];
42      }
43
44      int getRear() {
45          if(isEmpty())return -1;
46          int real_tail=(tail-1+arr.size()) % arr.size();
47          return arr[real_tail];
48      }
49
50      bool isEmpty() {
51          return (count==0);
52      }
53
54      bool isFull() {
```

```
        //if(count == arr.size())return true;
        //return false;
        return count == arr.size();
    }
};

/**
 * Your MyCircularDeque object will be instantiated and called as such:
 * MyCircularDeque* obj = new MyCircularDeque(k);
 * bool param_1 = obj->insertFront(value);
 * bool param_2 = obj->insertLast(value);
 * bool param_3 = obj->deleteFront();
 * bool param_4 = obj->deleteLast();
 * int param_5 = obj->getFront();
 * int param_6 = obj->getRear();
 * bool param_7 = obj->isEmpty();
 * bool param_8 = obj->isFull();
 */
```

Leetcode-1670-设计前中后队列

```
class Node {
public:
    int val;
    Node * next;
    Node * pre;
    Node(int val=0,Node *next=nullptr, Node
*pre=nullptr):val(val),next(next),pre(pre){}
    void insert_pre(Node *p){
        p->pre = pre;
        p->next = this;
        if(this->pre) {
            this->pre->next = p;
        }
        this->pre = p;
    }
    void insert_next(Node * p){
        p->pre = this;
        p->next = this->next;
        if(this->next){
            this->next->pre = p;
        }
        this->next=p;
    }
    void delete_pre(){
        if(this->pre == nullptr) return;
        Node *p = this->pre;
```

```
        this->pre = p->pre;
        if(p->pre){
            p->pre->next=this;
        }
        delete p;

    }
    void delete_next(){
        if(this->next==nullptr) return;
        Node *p=this->next;
        this->next = p->next;
        if(p->next){
            p->next->pre = this;
        }
        delete p;
    }
};

class Queue {
public:
    Node head,tail;
    int count;
    Queue():count(0){
        head.pre = nullptr;
        head.next = &tail;
        tail.pre = &head;
        tail.next = nullptr;
    }
    void push_front(int val){
        head.insert_next(new Node(val));
        count++;
    }
    void push_back(int val){
        tail.insert_pre(new Node(val));
        count++;
    }
    int pop_front(){
        if(isEmpty())return -1;
        int ret;
        ret=head.next->val;
        head.delete_next();
        count--;
        return ret;
    }
    int pop_back(){
        if(isEmpty())return -1;
        int ret;
        ret =tail.pre->val;
        tail.delete_pre();
```

```
        count--;
        return ret;
    }
    int front(){
        return head.next->val;
    }
    int back(){
        return tail.pre->val;
    }
    bool isEmpty(){
        return count==0;
    }
    int size(){
        return count;
    }
};

class FrontMiddleBackQueue {
public:
    Queue q1,q2;
public:
    FrontMiddleBackQueue() {}

    void pushFront(int val) {
        q1.push_front(val);
        update();
    }

    void pushMiddle(int val) {
        if (q1.size()>q2.size() ){
            q2.push_front(q1.pop_back());
        }
        q1.push_back(val);
        update();
    }

    void pushBack(int val) {
        q2.push_back(val);
        update();
    }

    int popFront() {
        if(isEmpty())return -1;
        int ret=q1.pop_front();
        update();
        return ret;
    }

    int popMiddle() {
```

```
        if(isEmpty())return -1;
        int ret=q1.pop_back();
        update();
        return ret;
    }

    int popBack() {
        if(isEmpty())return -1;
        int ret=-1;
        if(q2.isEmpty()){
            ret = q1.pop_back();
        }else{
            ret = q2.pop_back();
        }
        update();
        return ret;
    }

    bool isEmpty(){
        return ((q1.size()==0)&&(q2.size()==0));
    }

    void update(){
        if(q1.size()<q2.size()){
            q1.push_back(q2.pop_front());
        }
        if(q1.size()==q2.size()+2){
            q2.push_front(q1.pop_back());
        }
        return;
    }
};
/**
 * Your FrontMiddleBackQueue object will be instantiated and called as such:
 * FrontMiddleBackQueue* obj = new FrontMiddleBackQueue();
 * obj->pushFront(val);
 * obj->pushMiddle(val);
 * obj->pushBack(val);
 * int param_4 = obj->popFront();
 * int param_5 = obj->popMiddle();
 * int param_6 = obj->popBack();
 */
```

Leetcode-933-最近的请求次数

```cpp
class RecentCounter {
public:
    queue<int> q;
    RecentCounter() {}

    int ping(int t) {
        q.push(t);
        while( t - q.front()>3000){
            q.pop();
        }
        return q.size();
    }
};

/**
 * Your RecentCounter object will be instantiated and called as such:
 * RecentCounter* obj = new RecentCounter();
 * int param_1 = obj->ping(t);
 */
```

智力发散题

Leetcode-17.09-第 k 个数

```cpp
class Solution {
public:
    vector <int> q;
    int min(int a,int b ){
        if(a>b)return b;
        return a;
    }
    int getKthMagicNumber(int k) {
        q.push_back(1);
        int head3=0,head5=0,head7=0;
        while(q.size()<k){
            int min_ans = q[head3]*3;
            min_ans = min(min_ans,q[head5]*5);
            min_ans = min(min_ans,q[head7]*7);

            if(min_ans==q[head3]*3)head3++;
            if(min_ans==q[head5]*5)head5++;
            if(min_ans==q[head7]*7)head7++;

            q.push_back(min_ans);
```

```
        }
        return q[k-1];
    }
};
```

Leetcode-859-亲密字符串

```
class Solution {
public:
    bool has_repeat(string a,string b){
        int count[26]={0};
        for(int i=0;a[i];i++){
            count[a[i]-'a']++;
            if(count[a[i]-'a']==2)return true;
        }
        return false;
    }
    bool buddyStrings(string a, string b) {
        if(a.size() != b.size())return false;
        if(a == b) return has_repeat(a,b);
        //找两处不一样的地方，然后，看是否只有两处，且，这两处是字母交换位置
        int i,j;
        i=0;
        while(a[i]==b[i]){
            i++;
        }
        //a[i] <> b[i]
        j=i+1;
        if(j>=a.size())return false;
        while(j<a.size() && a[j]==b[j]){
            j++;
        }
        //只有一处差异
        if(j==a.size()) return false;
        //发现两处不一致
        //不能通过交叉换位获得相等
        if(a[i]!=b[j]||a[j]!=b[i]){
            return false;
        }
        j++;
        //有三个以上的差异
        while(j<a.size()){
            if(a[j]!=b[j])return false;
            j++;
        }
        return true;

```

```
    }
};
```

Leetcode-860-柠檬水找零

```
class Solution {
public:
    bool lemonadeChange(vector<int>& bills) {
        int count5=0,count10=0;
        bool ret=true;
        for(int i=0;i<bills.size();i++){
            switch(bills[i]){
                case 5:
                    count5++;
                    break;
                case 10:
                    if(count5>0){
                        count5--;
                        count10++;
                    }else{
                        return false;
                    }
                    break;
                case 20:
                    if(count10>0&&count5>0){
                        count10--;
                        count5--;
                    }else if(count5>=3){
                        count5-=3;
                    }else{
                        return false;
                    }
                    break;
                default:
                    break;
            }
        }
        return ret;
    }
};
```

Leetcode-969-煎饼排序

```cpp
class Solution {
public:
    void reverse(vector<int>& arr,int n,vector<int>& ind){
        for(int i=0,j=n-1;i<j;i++,j--){
            swap(arr[i],arr[j]);
            ind[arr[i]]=i;
            ind[arr[j]]=j;
        }
    }
    vector<int> pancakeSort(vector<int>& arr) {
        vector <int> ind(arr.size()+1);
        vector <int> ret;
        for (int i=0;i<arr.size();i++){
            ind[arr[i]]=i;
        }
        for (int i=arr.size();i>=1;i--){
            if(ind[i]!=0){
                ret.push_back(ind[i]+1);
                reverse(arr, ind[i]+1, ind);
            }
            if(i!=1){
                ret.push_back(i);
                reverse(arr,i,ind);
            }
        }
        return ret;
    }
};
```