中山大学计算机学院本科生实验报告

（2021 学年第 1 学期）

课程名称：Data structures and algorithms　　任课教师：张子臻

年级	20 级	专业（方向）	软件工程
学号	20337270	姓名	钟海财
电话	13397996670	Email	2940599563@qq.com
开始日期	2021/9/23	完成日期	2021/9/28

1. 实验题目

1）Mergesort for List　　2）猴子选大王

2.实验目的

通过对链表的进一步使用，加深对链表结构特点的理解：

1）　实现链表的归并排序，熟悉链表节点之间的连接、分割操作

2）　实现循环链表，熟悉链表中删除元素的操作

2. 程序设计

1）　Mergesort for List

设计思路：数组的归并排序分为分组，排序，合并三个操作，其中排序操作使用递归，故而链表的归并排序也分为这三个操作：

```
linkedlist* cut(linkedlist* head1) ;
linkedlist*merge(linkedlist* head1,linkedlist* head2);
```

```
void  mergesort(linkedlist *&head , int len){ //len 不使用，简便分组

if(head->next==NULL) return ;//只有一个节点，返回
```

```
    linkedlist *head2  =  cut( head ) ; //分割,并得到 head2

    mergesort(head,len);//排序：若可以继续分组，调用递归继续归并排序
    mergesort(head2,len);
    head = merge( head , head2) ;//按顺序合并
}
```

分组 cut ：通过快指针 fast 和慢指针 slow 将链表分组，开始时 slow = head ，fast= head->next ，通过循环“fast=fast->next->next,slow=slow->next”，当 fast 走到链表尾部的时候，slow 走到链表中部。同时我们得到后半段链表的头节点 head2=slow->next,并将 slow->next = NULL 将链表分割成 2 段。

排序 mergesort ：分别对两段头节点调用递归进行归并排序，且当 head->next==NULL 时（只有一个节点）直接返回，否则调用递归进行归并排序。

合并 merge ：将排序好的两段链表按顺序进行合并，且返回合并后的头指针 first，这里 merge 使用递归方式：若有一段链表为空，first=另一段不为空的链表的头节点，由于两段都已经是排好序的，故该不为空的链表的剩余部分也已经排好顺序；当处理的两段链表都不为空时，first=两段链表头节点数值小的那个，同时令 first->next=剩余两段链表（除去 first 指向的那个节点）头节点数值小的那个，这里使用递归，将两段链表按顺序合并。

代码：

```
//Mergesort for List
#include"mergeSort.h"
#include<iostream>
using namespace std ;
linkedlist* cut(linkedlist* head1) ;
linkedlist*merge(linkedlist* head1,linkedlist* head2);

void  mergesort(linkedlist *&head , int len){ //len 不使用，简便分组
if(head->next==NULL) return ;//只有一个节点，返回
   linkedlist *head2  =  cut( head ) ; //分割,并得到 head2
   mergesort(head,len);//排序：若可以继续分组，调用递归
   mergesort(head2,len);
   head = merge( head , head2) ;//按顺序合并
}

linkedlist* cut(linkedlist* head){//分割成 2 组，并得到 head2
    linkedlist*slow = head;
    linkedlist*fast = head->next ;
    while(fast!=NULL && fast->next!=NULL){
        slow=slow->next;
        fast=fast->next->next;
   }//fast 走到链表尾时，slow 走到链表中间
    linkedlist *head2  = slow->next ;//得到第二组的头节点 head2
    slow->next = NULL ;//使第一组末尾指向 NULL，分割成 2 组
    return head2 ;//返回第二组的头节点
}

linkedlist* merge(linkedlist* head1, linkedlist* head2){
    if (head1 == NULL)
        return head2;
    if (head2 == NULL)
        return head1;
    linkedlist* first = NULL;
    if (head1->data <= head2->data) {
        first = head1;
        first->next = merge(head1->next, head2);
    } else {
        first = head2;
        first->next = merge(head1, head2->next);
    }
    return first;
}
```

2） 猴子选大王

设计思路：使用尾指针 tail 实现循环链表，每次往链表尾部添加元素，且 tail->next=头节点，再通过循环链表的使用：p 最开始指向头节点，每次循环令 p2=p->next 的 next 指向 p4=p3->next，再删除节点 p3=p->next->next ,再令 p=p4，从而实现踢出报数为 3 的猴子出队并将指针 p 指向下次报数为 1 的猴子。循环持续 N-1 次，最终链表中只剩下一个节点，即为所选出的猴王。

代码：

```
//猴子选大王
#include<iostream>
using namespace std ;
struct Node{
    int data;
    Node *next;
};
class linkedlist{
private:
    Node * tail ;//尾指针 tail->next=head_node 头节点;
    int len;
public:
    linkedlist(): tail(0),len(0) {}
    ~linkedlist(){
        while(len>1){
           Delete3th(tail->next) ;
        } //最终只剩下 tail->next
        delete tail->next ;
    }
    void addlast(int e){//往链表尾添加元素并实现循环链表
        Node* node = new Node() ;
        node->data = e ;
        if(len==0){ //链表为空时添加元素
            node->next = node ;//实现循环链表
        }
        else{
        node->next = tail->next ;
        tail->next = node ;
```

```
        }
       tail = node ; //使 tail 始终为尾指针
          ++len;
    }
    void Delete3th(Node *&p){
        if(len>1){//删除 p 节点往后数的第 3 个节点 p3 并令 p=p4
            p=p->next; //p 从 p1 到 p2
            Node* p2 = p ;//记录 p2
            p=p->next;//此时 p 为 p3，要删除
            Node * temp = p;
            p2->next = p->next ;  //使 p2->next = p4
            p=p->next;   // 令 p = p4 ，p4 即为下一轮的 p1
            delete temp ;//删除 p3
            --len ;
        }
    }
    Node * getlast(){ //返回尾指针
        return  tail ;
    }
};

int main(){
    int N;
    cin >> N ;
    linkedlist Monkey;
    for(int i = 0 ; i < N  ; ++i){
        Monkey.addlast(i+1);
    }
    Node *p= Monkey.getlast()->next;//此时 p 为头节点
    for(int i = 0 ; i < N-1 ;++i){//删除 N-1 个节点，剩下的这个节点即为猴王
        Monkey.Delete3th( p );
    }
    cout<<p->data;
}
```

3. 程序运行与测试

1） Mergesort for List

序号	测试输入	输出	是否通过
1	4 1 -2 0 92 （第一个数字为需排序数字的个数）	-2 0 1 92	是
2	10 34747 8348 2829 299 9934 293 84728 288 28 3	3 28 288 293 299 2829 8348 9934 34747 84728	是

3	10 -93 -102 9 2 0 98 298 773 23 2	-102 -93 0 2 2 9 23 98 298 773	是

2) 猴子选大王

序号	输入	输出	是否通过
1	11	7	是
2	28	23	是
3	10	4	是
4	200	128	是

4. 实验总结与心得

本次实验的两个题目：链表的归并排序，猴子选大王，都是充分利用了链表各个节点之间连接的特点，如链表的归并排序是通过改变各个节点连接的关系实现排序，不用额外分配空间；猴子选大王是利用循环链表的闭环性及链表无空节点的特点，任意节点必存在一前一后的相邻节点，从而可以依次访问并删除第 3 个节点。

附录、提交文件清单