第三次上机解题报告

15081070 张雨任

一、A 题

1. 题目代号与评测记录序号：{A260085}
2. 代码：

```
#include <iostream>
#include <stdio.h>
using namespace std;
struct node
{
    int data;
    node* next;
};
int main()
{
    int n;
    int len=0;
    cin>>n;
    node *L,*p,*q,*r,*s;
    L=new node;
    p=L;
    int x;
    while(~scanf("%d",&x)){
        s=new node;
        s->data=x;
        p->next=s;
        p=s;
        len++;
    }p->next=NULL;
    p=L;
    if(n==1&&len==1)cout<<x<<"\n\n";
    else
    {
        for(int i=0;i<len-n;i++){
            p=p->next;
        }
        q=p->next;
        cout<<q->data<<"\n";
        r=q->next;
        p->next=r;
        delete q;
        p=L->next;
        for(int i=0;i<len-2;i++){
            cout<<p->data<<"->";
            p=p->next;
        }cout<<p->data;
    }
}
```

3. 解题思路：
   此题不难，去掉倒数第 n 个就是去掉正数第 len-n+1 个数，输入的链表以 EOF 结束，输出有两行数要看清题。链表只有一个数的时候要单独考虑。

二、B 题

1. 题目代号与评测记录序号：{B259974}
2. 代码：

```cpp
#include <iostream>
#include <stdio.h>
using namespace std;
struct node
{
    int data;
    node* next;
};
int main()
{
    int k;
    cin>>k;
    node *L,*p,*q,*r,*s;
    L=new node;
    p=L;
    int x;
    while(~scanf("%d",&x)){
        s=new node;
        s->data=x;
        p->next=s;
        p=s;
    }p->next=NULL;
    node *last=L;
    node *pre,*right;
    int cntr =0;
    int len =0;
    q=L->next;
    while(q!=NULL){
        len++;
        q=q->next;
```

```
31      }
32      int n=len;
33      q=L->next;
34      len=len/k;
35      while(q!=NULL&&cntr<len){
36          pre=NULL;
37          right=NULL;
38          s=q;
39          int i=0;
40          while(q!=NULL&&i<k){
41              right=q->next;
42              q->next=pre;
43              pre=q;
44              q=right;
45              i++;
46          }
47          last->next=pre;
48          last=s;
49          cntr++;
50          s->next=q;
51      }
52      p=L->next;
53      for(int i=0;i<n;i++){
54          cout<<p->data<<" ";
55          p=p->next;
56      }
57  }
58
```

3. 解题思路：
   这道题考察单链表反转，只不过这次是分段反转。注意剩下不足 k 个的不反转，所以 len/k 代表要反转的次数。

三、C 题

1. 题目代号与评测记录序号：{C261303}
2. 代码：

```
#include <stdio.h>
int main()
{
    long long n;
    while(~scanf("%lld",&n)){
        if(n&(n-1)){
            printf("GzhIsHandsome\n");
        }
        else{
            printf("GzhIsSoHandsome\n");
        }
    }
}
```

3. 解题思路：

这道题找到规律的瞬间是崩溃的。

一开始是打算用递推公式做这道题，如果 s[i]代表第 i+1 个拿到球的人的位置减一，那么递推公式是：

$$s[0]=0,\ s[i]=(s[i-1]+i)\%n\ (i>=1)$$

然后如果有一个人拿到了第二个球，说明这个 n 不是所求（这个也是推测出来的……）。本来打算设置一个 bool 数组，如果这个人拿到过球，那么就他的位置-1 对应的数组设置为 true，但是数组无法开到 long long 的那个数量级，所以只能放弃。然后我发现符合条件的目前有 1, 2, 4, 8，我猜测 16 可能也是符合条件的，试了一下发现果然是这样的。所以题目就转化为给一个 n，判断是否是 2 的 x 次方且 x 为整数。那么就很简单了，可以用经典的位运算来判断。因为 2 的 n 次方的二进制一定可以写成 1000…000（n 个零），而 2 的 n 次方减一的二进制可以写成 111…111（n 个 1），两个数对应的每一位都不相同，所以结果为零。所以 n&(n-1)不为零，输出”GzhIsHandsome”，否则输出”GzhIsSoHandsome”。

四、D 题

1. 题目代号与测评记录序号：{D260482}
2. 代码：

```
#include <stdio.h>
int main()
{
    int n,m,s=0;
    while(~scanf("%d%d",&n,&m)){
        for(int i=2;i<=n;i++){
            s=(s+m)%i;
        }
        printf("%d\n",s+1);
    }
}
```

3. 解题思路：

原始的约瑟夫问题，用链表有点麻烦所以用的递推公式。总共 n 个人从零开始报数一直到 m-1，报 m-1 的人退出，出列的人编号为 m%n-1。然后变成了之前 n 人报数的子问题，即从 m%n-1 的下一个人 m%n 开始从 0 报数，即上一轮报 x 的人，这一轮报的数是 x-m%n。假设 n-1 人报数的胜利者是 y，则 n 人报数他的编号是 y+m%n，因此存在关系 s[n]=s[n-1]+m%n，又因为 s[n-1]+m%n 可能大于报数的人数 n，而编号是不能超过 n 的，所以要 mod n，即 s[n]=(s[n-1]+m%n)%n，在通过分配律可得出：

s[n]=(s[n-1]+m%n)%n=s[n-1]%n+m%n%n=s[n-1]%n+m%n=(s[n-1]+m)%n，即得到递推公式 s[n]=(s[n-1]+m)%n，特殊情况为 s[1]=0，但这些编号都是从零开始的，所以最后的结果要+1，这就是数学归纳法解决约瑟夫问题，效率比链表高一些。处理大数据量的时候，比如第二次练习赛 F 题，用链表超时了，用这种方法就可以过。

五、E 题

1. 题目代号与测评记录序号：{E263220}
2. 代码：

```
#include <stdio.h>
struct node
{
    int data;
    node *prior;
    node *next;
};

bool smallToBig(node *L,int len)
{
    node *test;
    test=L->next;
    while(test->next!=nullptr){
        if(test->data>test->next->data){
            return false;
        }
        test=test->next;
    }
    return true;
}

int main()
{
    int n;
    while(~scanf("%d",&n)){
        node *L,*p,*q,*r,*s;
        L=new node;
```

```
28              p=L;
29              int len=n;
30              int cnt=0;
31              for(int i=0;i<n;i++){
32                  s=new node;
33                  scanf("%d",&s->data);
34                  p->next=s;
35                  s->prior=p;
36                  p=s;
37              }
38              p->next=nullptr;
39              if(smallToBig(L,len)){
40                  printf("0\n");
41                  continue;
42              }
43              else
44              {
45                  while(1){
46                      p=L;
47                      while(p->next!=nullptr){
48                          p=p->next;
49                      }r=p->prior;
50                      while(r!=L){
51                          if(r->data>p->data&&p->next==nullptr)
52                              delete r->next;
53                              r->next=nullptr;
54                              p=r;
55                              r=r->prior;
56                              len--;
57                          }
58                          else if(r->data>p->data&&p->next!=nullptr){
59                              r->next=p->next;
60                              p->next->prior=r;
61                              delete p;
62                              p=r;
63                              r=r->prior;
64                              len--;
65                          }
66                          else{
67                              p=r;
68                              r=r->prior;
69                          }
70                      }
71                      cnt++;
72                      if(len==1||smallToBig(L,len)){
73                          printf("%d\n",cnt);
74                          break;
75                      }
76                  }
77              }
78          }
79      }
80
```

3. 解题思路：
   这道题没 a，t 了，拿了半分。这道题我的思路就是完全模拟删除的过程，用双向链表来实现（因为要从后往前遍历）。从后往前遍历，如果后年的小于前面的，则删掉后面的，删完一轮则计数器自增。最后只剩下一个数或者数列是从小到大排列（因为小的杀不了大的），输出计数器的值。暴力算法其实注意细节就行了。
4. 易错点：
   双向链表维护的时候要注意“有来有回”，next 指针和 prior 指针都有指向正确的地方。要特殊考虑一种情况，如果本就是从小到大排列的，那么直接输出 0 比较省时间（虽然还是 t）。删除元素要记得长度自减。

六、F 题

1. 题目代号与测评记录序号：{F261240}
2. 代码：

```cpp
#include <iostream>
using namespace std;
struct node
{
    int data;
    node *next;
};
int main()
{
    int k,m=1;
    cin>>k;
    while(1){
        //生成循环链表
        node *L,*p,*q,*r,*s;
        L=new node;
        p=L;
        bool flag=false;
        for(int i=0;i<2*k;i++){
            s=new node;
            s->data=i+1;
            p->next=s;
            p=s;
        }p->next=L->next;
        //报数
        p=L->next;
        int len=2*k;
        int shift;
        int cnt=0;
        for(int i=0;i<2*k-1;i++){
            shift=m%len;
            if(shift==0)shift=len;
            if(shift==1)shift=len+1;
            for(int j=1;j<shift-1;j++){
                p=p->next;
            }
            q=p->next;
            r=q->next;
            p->next=r;
            p=r;
            if(q->data>=k+1&&q->data<=2*k){
                cnt++;
                if(cnt==k)break;
            }
            else{
                if(m%(k+1)==0)m++;
                else m+=k;
                flag=true;
                break;
            }
            delete q;
            len--;
        }
        p=L->next;
        for(int i=0;i<len-1;i++){
```

```
55                  r=p->next;
56                  delete p;
57                  p=r;
58                  r=r->next;
59              }
60              delete L;
61              if(!flag&&cnt==k){
62                  cout<<m<<"\n";
63                  return 0;
64              }
65          }
66      }
67
68
69
```

3. 解题思路：
   这道题我的做法很暴力，用链表模拟整个过程。先默认 m 值为 1，在循环中创建链表，之后用传统办法模拟约瑟夫环的整个过程，如果删除的过程中有好人，那么跳出循环，m 自增，如果删除的是坏人，那么累加 cnt 这个计数器，如果 cnt 累加到 k 都没有被清零，说明这 k 次删除删掉的都是坏人，那么说明这个 m 值是符合题意的。这样做的结果是 tle。
   这个做法的超时点有两个：
   首先，m 的很大的时候，完全模拟报数是很费时的。比如当前轮次，10 个人，数 14 的退出，其实是和数 4 退出的情况是相同的。当 m 远远大于当前人数 n 的时候，真正报数的很浪费时间的，所以应该用取余的方法求出等价的 m 值，这个取余要注意的是当余数是 1 和 0 的情况需要单独考虑。
   其次，m 以 1 为单位累加是一个很漫长的过程。经过考虑，把 m++优化成了 if(m%(k+1)==0)m++; else m+=k; 也就是 m 只能是(k+1)的整数倍或者(k+1)+1 的整数倍。原因如下：假设模拟已经进行到某 m 值的倒数第二轮，剩下的数为 1,2, … , k, x, y，其中 1~k 是好人，x 和 y 都是坏人，但是并不知道他们的编号，只知道是 k+1~2k 中的两个数。现在有两种情况，一种是倒数第二轮删掉 y，最后一轮删掉 x，另一种是倒数第二轮删掉 x，最后一轮删掉 y。第一种情况：当前 q（指向被删除的值的指针）指向 y，接下来要删除 x，那么 q 这个指针要移动的偏移量为(k+1)，说明此时 m 必须是(k+1)的整数倍，才能使 q 指针在下一次删除的时候指向 x。第二种情况：当前 q 指针指向 x，接下来要删除 y，q 的偏移量是 1 即可在删除 x 之后指向 y，即此时 m 必须是(k+1)的整数倍+1 才能够保证在下次删除的时候 q 指向 y。因此总结来说，m 只能为 k+1 的整数倍或者是 k+1 的整数倍加 1。所以逻辑就是，如果当前 m 是(k+1)的整数倍，那么 m++；否则 m 则是(k+1)的整数倍+1，那么需要自增 k 使得 m 能够达到(k+1)的整数倍。这样来看，时间复杂度可以降低很多。
4. 易错点：
   首先是解题思路中提到的两个超时点需要进行优化，不然只能拿到 0.8 分。
   其次每次 m 变化之后需要删除结点，不然内存会爆掉。