信息学测试解题报告（201919）

1、拍卖（auction. cpp）解题报告

【分析】

根据题意，需要找一个定价，使得能让 L 全买出去。

数据规模并不大，可以枚举每一个价格来判断多少人愿意高于这个价格，为了提高效率可以先排序，然后依次来枚举每个定价，这样效率是 O(NlogN)。

注意：飞船是有限个，如果愿意出价的人数比飞船数量多是不行的。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;

int n,m,ans,p;
int a[2000];

int main()
{
    freopen("auction.in","r",stdin);
    freopen("auction.out","w",stdout);
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++)
        scanf("%d",&a[i]);
    sort(a+1,a+m+1) ;
    ans=0;p=0;
    for(int i=1;i<=m;i++)
    {
        if(m-i+1>n)continue;
        int temp=(m-i+1)*a[i];
        if(temp>ans)
        {
            p=a[i];
            ans=temp;
        }
    }
    cout<<p<<" "<<ans;
    return 0;
}
```

2、挖煤（coal.cpp）解题报告

【算法分析】

动态规划：

对于一个煤矿，挖它相当于让之后的 a_i 都乘上(1-k%)；

设 f[i]表示从 i 开始的最大收益，从后向前 DP。

状态转移：

- 对于煤矿： f[i]=max(f[i+1],f[i+1]*(1-k%)+a[i])
- 对于补给站：f[i]=max(f[i+1],f[i+1]*(1+c%)-a[i]);

答案：f[1]*w 。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;

double n,k,c,w;
double t[200002],a[200002];
double f[200002];


int main()
{
    freopen("coal.in","r",stdin);
    freopen("coal.out","w",stdout);

    scanf("%lf%lf%lf%lf",&n,&k,&c,&w);
    for(int i=1;i<=n;i++)
        scanf("%lf%lf",&t[i],&a[i]);

    for(int i=n;i>=1;i--)
    {
        if(t[i]==1)
            f[i]=max(f[i+1],f[i+1]*(1-k/100)+a[i]);
        else
            f[i]=max(f[i+1],f[i+1]*(1+c/100)-a[i]);
    }
    printf("%.2lf\n",f[1]*w);
    return 0;
}
```

3、找整数（find.cpp）

【算法分析】

建立两个单调队列 L 和 H（其中 L 单调递增，H 单调递减），并时刻维护其单调性；

（1）依次读入每个整数 a[i]

（2）同时维护 L 和 H 的单调性。

（3）用 head 表示选择序列的起点，初始 head=1；ans 表示答案（ans=i-head+1）

（4）如果 a[H.front()]-a[L.front()]>k，则 head++，同时在 L 和 H 中删去 head 前的整数。

（5）跳到（1）

【双端队列】

单调队列用双端队列 deque 实现，使用方法如下：

需要包括头文件<deque>

定义：deque<数据类型> 变量名;

比如：deque<int> que; //定义了一个整型的双端队列;

操作函数：

que.clear() 移除容器中所有数据。 que.empty() 判断容器是否为空。

que.size() 返回容器中实际数据的个数。

que.front() 返回容器 que 的第一个元素的引用。

que.back() 返回容器 que 的最后一个元素的引用。

que.pop_front() 删除头部数据。que.pop_back() 删除最后一个数据。

que.push_back(x) 在尾部加入一个数据 x。que.push_front(x)在头部插入一个 x。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;

int n,k,a[100001],ans=0,head=1;                    //head 为 a 中取出连续整数的起始位置
deque<int>L,H;                                     //双端队列

int main()
{
    freopen("find.in","r",stdin);
    freopen("find.out","w",stdout);
    scanf("%d%d",&n,&k);
    for(int i=1;i<=n;i++)                          //处理每个整数
    {
        scanf("%d",&a[i]);                         //读入整数 a[i]
        while((!L.empty())&&(a[L.back()]>=a[i]))   //L 存储递增序列的下标
        {
            L.pop_back();                          //删除队尾，维护单调性
        }
        L.push_back(i);                            //第 i 个整数进入队列 L 的队尾
```

```
        while((!H.empty())&&(a[H.back()]<=a[i]))   //H 存储递减序列的下标
        {
            H.pop_back();                          //删除队尾，维护单调性
        }
        H.push_back(i);                            //第 i 个整数进入队列 H 的队尾

        while(a[H.front()]-a[L.front()]>k)         //如果最大减最小的差不合法
        {
            head++;                                //head 右移
            while(L.front()<head)
            {
                L.pop_front();                     //删除队首，即删掉 L 中 head 前面的下标
            }
            while(H.front()<head)
            {
                H.pop_front();                     //删除队首，即删掉 H 中 head 前面的下标
            }
        }
        ans=max(i-head+1,ans);                     //取 head->i 范围的连续整数
    }
    printf("%d",ans);
    return 0;
}
```

4、最短循环节（string.cpp）解题报告

【算法分析】

方法 1：暴力枚举删除的字符，用 KMP 求解，期望得分：40 分

方法 2：使用字符串哈希求解，期望得分：100 分，方法如下：

枚举循环节长度 L，分两种情况讨论：

- 删掉的字符不在[1,L]中，用 hash 判断之后每段是否与[1,L]相同，出现第一个不同时，二分出第一个不同的字符删掉，之后继续判断即可；
- 删掉的字符在[1,L]中，循环节必然为[L+2,L+L+1]，同理判断即可。

对于一个 L 需要判断 O(n/L)次以及二分 O(1)次，总时间复杂度 O(nlogn)。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
typedef unsigned long long ULL; //利用 unsigned long long 自然溢出，相当于对 2^64 取模

const int N=2e5+50;
const ULL base=31;              //哈希所用的进制
int n;
char s[N];
ULL hv[N],pw[N];

inline ULL get_hash(int l,int r){
    return hv[r]-hv[l-1]*pw[r-l+1];   // O(1)取得一个区间的哈希值
}

inline int find_min()
{
    for(int i=1;i<=n;i++) hv[i]=hv[i-1]*base+s[i]; // 处理前缀哈希值
    for(int i=1;i<n;i++)                           //枚举循环节长度
    {
        int tag=0;                                   //记录是否删去过字符
        for(int j=i+1;j<=n;j+=i)                   //暴力枚举每个循环节，判断是否相等
        {
            int len=min(i,n-j+1);
            if(get_hash(1,len)==get_hash(j,j+len-1)) continue;
            if(tag) {tag=2; break;}      // 已经删去过字符仍不合法，此时非法
            int l=1, r=len, ans=len;     //二分找第一个不同点
            while(l<=r)
            {
                int mid=(l+r)>>1;
                if(get_hash(1,mid)==get_hash(j,j+mid-1)) l=mid+1;
                else ans=mid, r=mid-1;
            }
            tag=1;                       //删去这个不同点，并把删除标记置为 1
            if(j+ans-1!=n){              //删去之后还需判断后面的一段是否相同
                int ed=min(j+i,n);
                if(get_hash(ans,ed-j)!=get_hash(j+ans,ed)) {tag=2; break;}
            }
            j++;
        }
        if(tag<2) return i;
    }
}
```

```
inline void solve()
{
    scanf("%d %s",&n,s+1);
    if(n==1) {puts("0"); return;}
    int len=find_min();             //求强制开头一段作为循环节的最小长度
    reverse(s+1,s+n+1);
    len=min(len,find_min());        //求强制结尾一段作为循环节的最小长度
    printf("%d\n",len);
}

int main()
{
    freopen("string.in","r",stdin);
    freopen("string.out","w",stdout);
    pw[0]=1;
    for(int i=1;i<N;i++) pw[i]=pw[i-1]*base;  //预处理 base^i
    int T;
    scanf("%d",&T);
    while(T--)
        solve();
}
```