2023年第六届广西大学生程序设计竞赛（正式赛）题解

难度	签到题	普通题	中等题	难题
题号	A J K	B D E H	C G	F I L M
状态	✔	✔	✘	✘

签到题

A Alpha, Beta, Omega

题目大意：

抽牌游戏。每次抽 k 张牌， 抽到一张 α 就 放入一张新的 β， 抽到一张 β 就 放入一张新的 Ω， 抽到一张 Ω 代表游戏胜利。询问在运气最差的时候，至多抽多少次才会胜利。

解题思路：

运气最差，自然是 先抽完 α， 再抽完 β， 最后才抽到 Ω 。

其实不用特意处理 抽 α 抽完了 还要同时抽 β 的情况。

抽到一张 α 就 放入一张新的 β， 就说明 α 要被抽两次（ α 和 新的β ）。

就直接 (a*2+b)/k+1 就行。

参考代码c++

```
#include <iostream>
using namespace std;
int main()
{
    int a,b,o,k;
    cin>>a>>b>>o>>k;
    if(a+b<k) cout<<1;
    else cout<<((a*2+b)/k+1);
    return 0;
}
```

J June

题目大意：

取两个字符串，按指定方式输出。

解题思路：

送分题。

参考代码c++

```
#include<iostream>
using namespace std;

int main(){
    string s,t;
    cin>>s>>t;
    cout<<"Good luck to "<<s<<" in "<<t<<" and have fun!";
}
```

K Keyboard

题目大意：

不断将当前字符串进行复制并粘贴到其后面，询问是否有机会能够与另一个字符串相吻合

解题思路：

其实不用 特意判断 第一次复制粘贴的情况， 只要当前字符串长度小于另一个字符串长度就循环复制粘贴自身就行。

当然也可以提前判断下是否满足幂数倍，不判断其实也能过题。

参考代码c++

```
using namespace std;
string s,t;

int main()
{
    cin>>s>>t;
    while(s.size()<t.size())s+=s;
    if(s==t)cout<<"Smart People's Big Win!";
    else cout<<"Lazy Dog's Great Failure...";
    return 0;
}
```

普通题

B Brilliant Idea

题目大意：

这道题目让我们玩一个字符串的游戏，每次可以选择一个子串，把这个子串中的字符都变成该子串中的任意一个字符。游戏的目标是尽可能多地进行操作。猜猜在所有长度为 n 且只由小写字母构成的字符串中，有多少个字符串能够达到以下三种情况中的一种：

- 通过变化不能达到任何两个字符相同的状态；
- 通过有限次变化能够达到任何两个字符不同的状态；
- 无论进行多少次变化，都无法达到任何两个字符不同的状态。

解题思路：

n=2 时是一种特例，单独输出即可。
n>2 时

- 第一问需要我们计算出长度为 n 的字符串中有多少个没有相邻相同字母的字符串。其实只有 aa，bb ... zz，这些情况满足，即 26 种。
- 第二问需要我们判断给定的字符串能否经过有限次操作后变成只包含一个字母。其实推导以下发现并不能满足，要么属于第一种情况，要么属于第三种情况，即 0 种。
- 三问需要我们判断给定的字符串是否可以无限次进行操作。如果该字符串的长度为偶数，那么直接构造一类字符串即可。

参考代码c++

```
#include <iostream>
using namespace std;
long long a,b,c=26ll,sum=1,mod=1e9+7;
int main()
{
    cin>>a;
    if(a==2) cout<<"26 650 0";
    else{
        b=a%mod;
        while(b>0){
            if(b&1) sum=(sum*c)%mod;
            c=(c*c)%mod;
            b>>=1;
        }
        cout<<"26 0 "<<(sum-26+mod)%mod;
    }
    return 0;
}
```

D Dream Team

题目大意：

将 3n 个人分成 n 个团队，每个团队有三个人，并且要最小化所有人的焦虑值之和。对于一个人，他的焦虑值等于他所在团队中他不认识的人的数量。

解题思路：

本题可以使用并查集相似的思路进行求解。

对于一个团队中的学生而言，它们都是互相熟悉的，如果我们把其中的一个学生看做代表，那么这个代表所在的连通块就是这个团队能够扩展到的全部学生。因此，我们可以考虑将同一个连通块内的所有学生分到同一个组中，这样不会增加任何一个学生的焦虑值。

对于一个学生 u 而言，我们可以使用并查集等方法求出它所在的连通块 c_u ，然后对于每个团队 t，我们判断 t 中是否存在一个学生 v 所在的连通块 c_v 与 c_u 相同。如果存在这样的 v，那么 t 中的所有学生都能够被分配到和 u 相同的组中，此时团队 t 的焦虑值为 0；否则 t 中的每个学生都需要被分配到和 u 不同的组中，其焦虑值为团队成员之间不熟悉的人数。

综上所述，我们可以遍历每个学生 u，对于每个团队 t 判断它们是否能被分到同一个组内。然后将所有团队的焦虑值加起来，这就是答案。

参考代码c++

```
#include <iostream>
using namespace std;
int n,m,x,y,a,b,f[3000006],num[3000006];
int fnd(int a){
    return f[a]==a?a:f[a]=fnd(f[a]);
}
int main()
{
    cin>>n>>m;
    n*=3;
    for(int i=1;i<=n;i++) f[i]=i;
    for(int i=1;i<=m;i++){
        cin>>x>>y;
        int fx=fnd(x),fy=fnd(y);
        if(fx!=fy) f[fx]=fy;
    }
    for(int i=1;i<=n;i++) num[fnd(i)]++;
    for(int i=1;i<=n;i++){
        if(f[i]==i){
            if(num[i]%3==2) a++;
            else if(num[i]%3==1) b++;
        }
    }
    if(a>=b) cout<<b*4+(a-b)/3*8;
    else cout<<(a+b)*2;
    return 0;
}
```

E Evil Capitalist

题目大意：

有一个公司雇佣了 n 个员工，每个员工的工资为 a_i 元。如果一个员工的工资比他下面的员工的工资高，那么这个员工就会感到不满意。为了防止员工罢工，公司可以选择进行若干次两人谈话，让他们互相知道彼此的工资，并且对不满意的员工进行适当的工资调整，使得最后只有一个员工感到不满意，其他员工都感到满意。问公司最少需要支付多少元才能实现该目标。

解题思路：

这道题目是一个经典的贪心算法问题。我们可以先将所有员工按照工资从高到低排序，并将所有人都初始化为不满意状态。接下来，我们遍历每个员工，如果该员工还没有被处理过，那么我们就找到他下面第一个比他工资低的员工，然后进行以下操作：

如果两个员工工资相同，那么他们都变成满意状态。
如果下方员工比当前员工工资低，那么下方员工变成满意状态，当前员工工资上涨到下方员工的工资。然后继续向下查找，直到没有比当前员工工资低的员工。
这样处理完所有员工之后，就可以得到最终需要支付的总金额了。时间复杂度为 O(nlogn)，其中 n 为员工数量，主要消耗在排序操作上。

参考代码c++

```
#include <bits/stdc++.h>
using namespace std;
long long n,sum,mx,cnt,now,a[200007];
int main() {
    cin >> n;
    for(int i=0;i<n;i++) cin>>a[i];
    sort(a,a+n);
    a[n]=2e9+7;
    for(int i=1;i<=n;i++){
        if(a[i]==a[i-1]) cnt++;
        else{
            now=cnt%2?0:a[i]-a[i-1];
            sum+=now;
            mx=max(mx,now);
            cnt = 0;
        }
    }
    cout<<sum-mx;
    return 0;
}
```

H Hide and Seek

H Hide and Seek

题目大意：

这道题目的题意是，给定一个正整数 x （1≤x≤a），以及 m 个约束条件，每个约束条件形如 b_j mod x =c_j 。求在满足所有约束条件的前提下，[1,a] 中有多少个正整数同时满足所有条件。保证存在至少一个满足条件的正整数。

解题思路：

使用循环读入每个约束条件，计算出 b_j −c_j 的最大公约数 gcd，并计算出满足所有约束条件的最小正整数 l。

从 1 到 gc 的平方根，枚举 gcd 的每个因子 i。如果 i 满足要求，则令 ans 自增 1；如果 gcd/i 也满足要求且 gcd/i 不等于 i，则同样令 ans 自增 1。最后输出 ans，即为满足约束条件的正整数的数量。

gcd=0 时应当直接输出 a−l+1。

参考代码c++

```
#include<bits/stdc++.h>
using namespace std;
long long m,a,b,c,gc,l=1,ans=0;
int main(){
    cin>>m>>a;
    while(m--){
        cin>>b>>c;
        gc=__gcd(b-c,gc);
        l=max(l,c+1);
    }
    for(int i=1;i<=gc/i;i++){
        if(gc%i==0){
            if(i<=a&&i>=l)ans++;
            if(gc/i!=i&&gc/i<=a&&gc/i>=l)ans++;
        }
    }
    if(gc==0) cout<<a-l+1;
    else cout<<ans;
}
```