

10.13 一面题解

——by Handy

Editor: Yue_chen

比赛链接: <https://ac.nowcoder.com/acm/contest/92607>

密码: hut1013

A 题: lhz 学长的伯牙绝弦

题目链接: <https://ac.nowcoder.com/acm/contest/92607/A>

本题逻辑很简单, 只要注意一下坑就行。

1、学号是由 11 位数字字符组成的字符串。

重点 1: 11 位数字

```
if(s.size()==11)
```

重点 2: 数字字符

```
bool p(string s){  
    for(int i=0;i<=10;i++) if(s[i]<'0' || s[i]>'9') return false;  
    return true;  
}
```

完成筛选后就可进行判断是不是老登辣!!!

2、24 级新生的学号以 24 开头。

3、特殊的, 存在 23 级的预科生也属于 24 级新生, 他们的学号以 23 开头且第 5 位必须是 7。

```
if((s[0]=='2'&& s[1]=='4') || (s[0]=='2'&& s[1]=='3'&& s[4]=='7'))
```

如果都不符合就输出-1, 表示没有新生通过这道题目(都送伯牙绝弦了竟然没人想喝!!!)

完整代码附上

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
bool p(string s){
```

```
        for(int i=0;i<=10;i++) if(s[i]<'0' | | s[i]>'9') return false;
        return true;
    }

    int main() {
        int n;
        scanf("%d",&n);
        for(int i=1;i<=n;i++){
            string s;
            cin>>s;
            if(s.size()==11&&p(s)){
                if((s[0]=='2'&&s[1]=='4') | | (s[0]=='2'&&s[1]=='3'&&s[4]=='7')){
                    printf("%d",i);
                    return 0;
                }
            }
        }
        printf("-1");
        return 0;
    }
}
```

B 题：zqt 学长请你吃 1+1 套餐

手动将充足的数据存表。接下来枚举 3 个数都取那些值，第一层循环枚举最大的数，第二次次大数，第三次最小数。按照这样的枚举顺序每次找出的数一定是比上一个数大的最小数。用 num 记录当前枚举到第几小数，当 num == n 时就可以输出并退出了。

代码如下：

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
long a[13]; long
```

```
a[13]={0,1,11,111,1111,11111,111111,1111111,11111111,111111111,1111111111,11111111111,111111111111,1111111111111};
```

```
int n;
```

```
int num = 0;
```

```
long long ans;
```

```
int main() {
```

```
    scanf("%d",&n);
```

```
    for(int x=1;x<=14;x++)
```

```
        for(int y=1;y<=x;y++)
```

```
            for(int z=1;z<=y;z++)
```

```
                if(++num==n){
```

```
                    printf("%lld",a[x]+a[y]+a[z]);
```

```
                    return 0;
```

```
                }
```

```
    return 0;
```

```
}
```

C 题：范德蒙德卷积

当你看到这么一大串陌生的知识点的时候，你可能已经懵了，这又是什么范德蒙德矩阵，又是什么范德蒙德卷积。你可能在想：我靠我一个都没学过，这题怎么做啊？算了算了跳下一题吧。那么恭喜你，你错过了一道水题。当你仔细看最后的递推公式的时候，你就会发现，原来这题这么简单。

公式如下：

$$a(n) = \begin{cases} 1, & n = 0 \\ 2, & n = 1 \\ 2 \times a(n-1) - a(n-2), & n > 1 \end{cases}$$

其实不难发现， $n > 1$ 时移个项，就是 $a_n + a_{n-2} = 2 * a_{n-1}$ 。聪明的你一定发现了这是一个等差数列，并且对于所有的 $n \geq 2$ 成立。所以 a_n 的通项公式就是 $a_n = n + 1$ ($n \geq 0$) 所以本题输出就是 $(n+1)$ 对 998244353 余数，即 $(n+1) \% 998244353$ 。
(记得要开 long long!!!)

代码如下：

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

int main() {
    long long n;
    scanf("%lld", &n);
    printf("%lld", (n+1)%998244353);
    return 0;
}
```

D 题：炉石传说

伟大，无需多言！！！！
我是 xmy 学长的狗！！！！

代码如下：

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

int main() {
    printf("想进实验室");
    return 0;
}
```

E 题: Yue_chen VS Qiu_yi

伟大的 xmy 学长（即 Yue_chen 学长）竟然还需要耍赖？很明显是 Yue_chen 学长是在让 Qiu_yi 学长。

对于这道题我们可以很轻易的观察出来，xmy 学长是否使某场比赛 unrated 并不影响其他场比赛的积分情况，即每场比赛的积分情况是独立的。所以 xmy 学长应当尽可能的将 Qiu_yi 学长比自己分多的比赛 unrated。使得自己的 rating 比 Qiu_yi 学长的 rating 高。

当然了，如果 xmy 学长在不要赖的情况下就能打败 Qiu_yi 学长。那么他并没有理由去 unrated 任何一场比赛。

这里我们用 $c[i]$ 表示每场比赛 xmy 学长与 Qiu_yi 学长 rating 之的变化量的差值，用 ans 表示 n 场比赛后 xmy 学长的 rating 值与 Qiu_yi 学长的差值。

每当 xmy 学长 unrated 第 i 场比赛后，便可以消去该场比赛和 Qiu_yi 学长的 rating 值的变化量的差异，即可以用 $ans -= c[i]$ 。当 xmy 学长按照 $c[i]$ 从小到大的顺序消去 x 场比赛的使得 $ans > 0$ 的时候，xmy 学长便可以打败 Qiu_yi 学长。如果可怜的 xmy 学长 unrated 所有场比赛后依然不能使 $ans > 0$ ，那么 xmy 学长便不能打败 Qiu_yi 学长（哦，伟大的 xmy 学长要被 Qiu_yi 学长薄纱了呢）

代码如下：

```
#include <bits/stdc++.h>
#define MAXN 10000005
using namespace std;

int a[MAXN], b[MAXN];
int c[MAXN];

int main() {
    int n;
    scanf("%d", &n);
    for(int i=1; i<=n; i++) scanf("%d", &a[i]);
    for(int i=1; i<=n; i++) scanf("%d", &b[i]);
    long long ans=0;
    for(int i=1; i<=n; i++){
        c[i]=a[i]-b[i];
        ans+=c[i];
    }
    if(ans>0){
```

```
        printf("YES\n0");
        return 0;
    }
    sort(c+1,c+n+1);
    int i=0;
    while(c[++i]<0){
        ans-=c[i];
        if(ans>0){
            printf("YES\n%d",i);
            return 0;
        }
    }
    printf("NO");
    //for(int i=1;i<=n;i++) printf("%d ",c[i]);
    //printf("\n%d",ans);
    return 0;
}
```

F 题: LOVE 99999999999999999999899999999999...

怎么说呢。。。其实这题就是斐波那契额数列，以 20 位为一组，每一组按位存入菲薄那些数列的一项,一共需要 $1600/20=80$ 项。

就像这样：

```
0.
00000000000000000000
00000000000000000001
00000000000000000001
00000000000000000002
00000000000000000003
00000000000000000005
00000000000000000008
00000000000000000013
00000000000000000021
...
00008944394323791464
00014472334024676221
00023416728348467685
```

然后按照 l 到 r 把每一位加起来就行了。

代码如下：

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

long long f[85];
int a[85][20];

int main(){
    f[1]=1;f[2]=1;
    for(int i=3;i<=80;i++) f[i]=f[i-1]+f[i-2];
    for(int i=1;i<=80;i++){
        int k=19;
        while(f[i]){
            a[i][k--]=f[i]%10;
            f[i]/=10;
        }
    }
    int n;
```

```
scanf("%d",&n);
while(n--){
    int l,r;
    scanf("%d%d",&l,&r);
    int ans=0;
    for(int i=l-1;i<=r-1;i++) ans+=a[i/20][i%20];
    printf("%d\n",ans);
}
return 0;
}
```

这里额外附一份命题人 F 题解：

通过提示可以发现，将 1/998999 的小数部分以 3 个为一组，将每一组视为一个数，前 9 个数分别为 0, 1, 1, 2, 3, 5, 8, 13, 21。容易看出这是一个从 0 为第一项，1 为第二项的斐波那契数列。由于题目所给的数的分母部分是 99999999999999999999999999999999（十九个 9+8+二十个 9），和 998999 非常相似，所以可通过计算 $\frac{1}{89}$, $\frac{1}{9899}$, $\frac{1}{99989999}$ 等去找出规律。容易发现组成一组斐波那契数列的数字的数量和 8 后面的 9 的数量相等。因此我们可以先求出斐波那契数列的前 80 ($1600/20$) 项，再将每一位斐波那契数列的数加上前缀 0，补至 20 位，最后拼接在一起就是所求的小数点后 1600 位数字。可以把每一位数字存在数组 nums 中，便于计算。

本题到此直接暴力计算数组中第 l 到第 r 个元素的和即可得到答案（数据还是太小了，没卡住暴力）。本题主要考察的是前缀和的简单应用，求出数组 `nums` 的前缀和数组 `pre` 即可通过 `pre[r]-pre[l-1]` 来 $O(1)$ 求出答案。

```
1. #include<iostream>
2.
3. using namespace std;
4.
5. int main()
6. {
7.     //先进行预处理
8.     //将斐波那契数列前 80 项求出
9.     //本题中斐波那契数列第一项是从 0 开始的
10.    //并且斐波那契数列的第 80 位大约为 1e17，需要开 long long
11.    long long dp[80]={0,1};
12.    for (int i=2;i<=79;++i) dp[i]=dp[i-1]+dp[i-2];
13.
14.    //将斐波那契数列的每一项以分成 20 个一组的数存在 pre 数组中
15.    int pre[1605]={0};
16.    for (int i=1600,j=79,cnt=0;i>=1;--i)
17.    {
18.        if (dp[j]>0)
19.        {
20.            pre[i]=(int)dp[j]%10;
21.            dp[j]/=10;
22.        }
23.        ++cnt;
24.        if (cnt==20)
25.        {
26.            cnt=0;
27.            --j;
28.        }
29.    }
30.
31.    //求出前缀和，使查询的时间复杂度降至 O(1)
```

```
32.     for (int i=1;i<=1600;++i) pre[i]+=pre[i-1];
```

```
33.
```

```
34.     int n;cin>>n;
```

```
35.
```

```
36.     while (n--)
```

```
37.     {
```

```
38.         int l,r;cin>>l>>r;
```

```
39.         //通过前缀和可求得结果
```

```
40.         cout<<pre[r]-pre[l-1]<<endl;
```

```
41.     }
```

```
42.
```

```
43.     return 0;
```

```
44. }
```

G 题：学弟征婚

这道题的题解想了想发现题目已经打意思说的很清楚了，唯一的难点就是如何求 gcd。

1.GCD 定义

整数 a 和 b 的最大公约数是指能同时整除 a 和 b 的最大整数，记为 $\gcd(a,b)$ 。例如， $\gcd(15,81)=3, \gcd(0,44)=44, \gcd(0,0)=0, \gcd(-6,-15)=3, \gcd(-17,289)=17$ 。

注意：由于 $-a$ 的因子和 a 的因子相同，因此 $\gcd(a,b)=\gcd(|a|,|b|)$ 。编码时只需要关注正整数的最大公约数。

2.GCD 性质

(1) $\gcd(a,b)=\gcd(a,a+b)=\gcd(a,k*a+b)$ 。

(2) $\gcd(ka,kb)=k*\gcd(a,b)$ 。

(3) 定义多个整数的最大公约数： $\gcd(a,b,c)=\gcd[\gcd(a,b),c]$ 。

(4) 若 $\gcd(a,b)=d$ 则 $\gcd(a/d,b/d)=1$ ，即 a/d 与 b/d 互素。这个性质很重要。

(5) $\gcd(a+cb,b)=\gcd(a,b)$ 。

3.GCD 编程

下面介绍 3 种 GCD 算法。不过，编程时可以不用自己写 GCD 代码，而是直接使用 C++ 函数 `std::__gcd(a,b)`。注意：参数 a 和 b 都应该是正整数，否则可能会返回负数。

1) 欧几里得算法

用辗转相除法求 GCD、即 $\gcd(a,b)=\gcd(b,a \bmod b)$ 。代码如下。

```
int gcd(int a,int b){           //一般要求 a>=0,b>0.若 a=b=0，代码也正确，则返回 0
    return b? gcd(b, a% b):a;
}
```

这是最常用的方法，它极为高效，拉梅定理给出了复杂度分析。

拉梅定理 用欧几里得算法计算两个正整数的最大公约数，需要的除法次数不会超过两个整数中较小的那个十进制数的位数的 5 倍。

推论 用欧几里得算法求 $\gcd(a,b), a>b$ ，需要 $O((\log_2 a)^3)$ 次位运算。

本题代码如下：

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
int a[1005];
```

```
int gcd(int a,int b){
```

```
    return b? god(b, a% b):a;
```

```
}
```

```
int main() {
```

```
    int t;
```

```
    scanf("%d",&t);
```

```
    while(t--){
```

```
        memset(a,0,sizeof(a));
```

```
        int n;
```

```
        scanf("%d",&n);
```

```
        bool p=true;
```

```
        for(int i=1;i<=n;i++){
```

```
            scanf("%d",&a[i]);
```

```
            for(int j=1;j<i;j++)
```

```
                if(__gcd(a[i],a[j])==1 && p){
```

```
                    p=false;
```

```
                    printf("YES\n");
```

```
            }
```

```
        }
```

```
        if(p) printf("NO\n");
```

```
    }
```

```
    return 0;
```

```
}
```

H 题: Cmy 学长的签到题。

(本题由 Yue_chen 助写)

传送门: <https://ac.nowcoder.com/acm/contest/92607/H>

题目描述:

现在的 C 学长有一个字符串 S 且 $|S| = 1$ ($|S|$ 表示字符串 S 的长度)。学长对该每次操作从 S 中选择一个子串 a (S 本身也是 S 的子串) 替换为一个新的字符串 b , (a, b 需要满足以下条件: $|a| \leq |b|$ 且 $|a| > 0$), 学长将所有 n 次操作 a, b 都记录了下来, 但是实在不够聪慧, 在记录时没有将 a, b 分开记录, 也没有记录下顺序, 只剩下一个包含所有 n 次操作, 长度为 $2n$ 字符串序列 c 以及一个 n 次修改后的字符串。

现在 C 学长想请厉害的新生们帮他找回最开始的那个字符串。

对于这种看不懂题面 (出题人很屎) 的题咱们可以从样例入手:

输入:

```
1
2
qmp
hmg
hm
h
qmpg
```

输出:

```
h
```

对该样例考虑 0 次操作时:

```
h
```

对该样例考虑 1 次操作时:

```
h
hmg
hmg
```

对该样例考虑 2 次操作时:

```
h
hmg
hm
qmp
qmpg
```

因为无论操作多少次, 这个例子的答案一定为 h , 所以你也可以不断往下操作...

很容易发现，替换进来的串总有一天要出去。

1) 再次被替换出去。

2) 作为最终串出去。

而出去的串全会在操作序列里。

所以，当咱们进行替换操作的时候，任何替换进来的随机串（除了初始串以外）所有的串都会出现两次（或者说所有的字母都出现了偶数次）。

所以我们只需要记录这些字母出现的次数，最后找到那个奇数次的字母输出就好。

本题完整代码：

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

using i64 = long long;

void solve() {
    int n; cin>>n;

    int cnt[26] = {0}; //记录 26 个字母的出现次数
    for(int i=1; i<=n; ++i) {
        string s; cin>>s;
        for(int j=0; j<(int)s.size(); ++j) {
            cnt[s[j] - 'a']++; //累计出现次数
        }
    }

    for(int i=0; i<26; ++i) {
        if(cnt[i] % 2) { //cnt[i]是奇数
            cout<<(char)('a' + i)<<"\n";
            return ;
        }
    }
}

int main(void) {
    ios::sync_with_stdio(0); cin.tie(0);

    int t=1; // cin>>t;
    while(t--) solve();
    return 0;
}
```

I 题：xmy 的数学竞赛

Emmmmm.....很显然这题就是求 $\sum_{x=1}^n \sum_{i=1}^x i = \sum_{x=1}^n \frac{x(x+1)}{2} = \frac{n(n+1)(n+2)}{6}$

时间复杂度为 $O(1)$ 。

唯一要注意的就是要记得开 long long。

代码如下：

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
long long ans;
```

```
int main() {
```

```
    long long n;
```

```
    scanf("%lld",&n);
```

```
    ans=n*(n+1)*(n+2)/6;  
    printf("%lld",ans);  
    return 0;  
}
```