哈尔滨理工大学 2021 级新生赛题解

A 考试周破防

题解：

用字符串数组保存对应周一到周五对应的课程安排，输出时特判为 5 的情况即可。

标程：

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


string s[10] ;
void Init_s()
{
    s[1]="Higher mathematics";
    s[2]="Linear algebra";
    s[3]="Principle of computer composition";
    s[4]="Database system concept";
    s[5]="Data structures and algorithms";
}
int main()
{
    Init_s();
    int n;
    cin>>n;
    if(n == 5)
    {
        printf("You were playing games!\n");
        return 0;
    }
    printf("You were studying ");
    cout << s[n];
    printf("!\n");
}
```

B 咖啡店

题解：

贪心地选取 5 块钱，如果 5 块钱不够或者需要用 1 元补充剩下金额的情况，则支付若干数量 1 块钱

标程：

```
#include <bits/stdc++.h>
typedef long long LL;
using namespace std;
int main()
{
    LL a, n;
    scanf("%lld%lld", &n, &a);
    LL cnt5 = 0, cnt1 = 0;
    cnt5 = min(a / 5, n);
    cnt1 = a - cnt5 * 5;
    printf("%lld %lld\n", cnt5, cnt1);
}
```

C kiki 和 bob 玩取石子

题解：

当石子数为 1、2、3 时，全部拿走取胜，
当石子数为 4 时，无论拿走多少，下一个人都必定能拿走剩下的所有而取胜，
当石子数为 5、6、7 时，分别拿走 1、2、3 颗石子可以让下一个人面对剩余 4 颗石子的局面
总结规律发现当 n 是 4 的倍数是先手必败，否则先手必胜

标程：

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    scanf("%d",&n);
    if(n%4) printf("kiki");
```

```
    else printf("bob");
}
```

D 猴王 kiki 分桃

题解：

就是求 L 到 R 之间对 n 求余最大的余数是多少，可以循环判断，也可以直接判断。

标程：

```
#include <iostream>
using namespace std;
int main()
{
    int k, L, r, n;
    while(cin >> n >> L >> r)
    {
        if( L / n == r / n)
        {
            cout <<   ( r % n ) << endl;
        }
        else
        {
            cout << ( n - 1) << endl;
        }
    }
    return 0;
}
```

E 很二的拆分

题解：

显然奇数不能被拆分，把偶数转成二进制即可

标程：

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n;
    cin >> n;
```

```
    if (n & 1) {
        cout << -1 << '\n';
        return 0;
    }
    for (int i = 30; i >= 1; --i) {
        if (n & (1 << i)) cout << (1 << i) << ' ';
    }
    return 0;
}
```

F 构造字符串

题解：

由字典序的性质来看，我们希望尽早使用更小的字符，所以可以如下贪心：
按照字典序比较 S 和将 S 反转后的字符串 S′
如果 S 较小，就从 S 的开头取出一个字符，加入 T 的末尾
如果 S′较小，就从 S 的末尾取出一个字符，加入 T 的末尾
如果相同则取哪个都行

标程：

```
#include<bits/stdc++.h>
using namespace std;
char s[2005];
int main()
{
    int n;
    scanf("%d",&n);
    cin>>s;
    int a=0,b=n-1;
    while(a<=b)
    {
        bool left=false;
        for(int i=0;a+i<=b;++i)
        {
            if(s[a+i]<s[b-i])
            {
                left=true;
                break;
            }
```

```
            else if(s[a+i]>s[b-i])
            {
                left=false;
                break;
            }
        }
        if(left) putchar(s[a++]);
        else putchar(s[b--]);
    }
}
```

G 信号之旅

题解：

分情况讨论
如果起点和终点的横坐标和纵坐标都不相同，那么是否绕过 bug 点对距离没有影响；
如果起点和终点的横坐标或纵坐标由相同，只需要判断 bug 点是否在由起点和终点形成的线段上，如果在，需要多 2 步将其绕开，否则没有影响。

标程：

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int t, x1, x2, x3, y1, y2, y3;
     cin >> t;
     while (t -- )
     {
         cin >> x1 >> y1;
         cin >> x2 >> y2;
         cin >> x3 >> y3;
         if (x1 != x2 && y1 != y2) cout << abs(x1 - x2) + abs(y1 - y2) << endl;
         else if (x1 == x2)
         {
             if (x3 == x2 && y3 < max(y1, y2) && y3 > min(y1, y2)) cout << 2 + abs(y1 - y2) <<
endl;
             else cout << abs(y1 - y2) << endl;
         }
         else if (y1 == y2)
```

```
        {
            if (y3 == y1 && x3 < max(x1, x2) && x3 > min(x1, x2)) cout << 2 + abs(x1 - x2) <<
endl;
            else cout << abs(x1 - x2) << endl;
        }
    }
    return 0;
}
```

H 小球滚动

题解：

由于小球之间没有区别，所以可以将小球相遇后返回当作是保持原样交叉通过。
于是每只小球可以看作是独立行动的，互相之间不产生影响。
所以要求最长/最短时间，只要求小球到滑槽的最大/最小距离即可。

标程：

```
#include<bits/stdc++.h>
using namespace std;
int a[1000005];
int main()
{
    int l,n;
    scanf("%d %d",&l,&n);
    for(int i=1;i<=n;++i)
        scanf("%d",&a[i]);
    int minT=0,maxT=0;
    for(int i=1;i<=n;++i)
    {
        minT=max(minT,min(a[i],l-a[i]));
        maxT=max(maxT,max(a[i],l-a[i]));
    }
    printf("%d %d\n",minT,maxT);
}
```

I kiki 看球赛

题解：

第一次记录时小的数+1 就是当时的最多平分次数，因为 0：0 也算是一次平分。
之后每组数据，与前面一组比较可能的平分次数，

注意不要疏漏前面一组低分追平的状况。

标程：

```
#include<bits/stdc++.h>
using namespace std;
int a[10005],b[10005];
int main()
{
	int n;
	cin>>n;
	for(int i=1;i<=n;++i)
	{
		cin>>a[i]>>b[i];
	}
	int A=0,B=0,ans=1;
	for(int i=1;i<=n;i++)
	{
		A=max(a[i-1],b[i-1]);
		B=min(a[i],b[i]);
		if(B>=A)
			ans+=B-A+(a[i-1]!=b[i-1]);
	cout<<ans<<endl;
	}
	return 0;
}
```

J　跳一跳

题解：

每到达一块瓷砖更新当前可到达瓷砖的最大值。

标程：

```
#include <bits/stdc++.h>
typedef long long LL;
const LL maxn = 1e5 + 5;
using namespace std;


LL a[maxn];
```

```
LL n;


int main()
{
    LL i,j;


    LL ans = 1;


    cin >> n;
    for(i=1;i<=n;i++)
        scanf("%lld", &a[i]);


    for(i = 1 ; i <= min(n, ans) ; i ++)
    {
        ans = max(ans, i + a[i]);
        if(ans >= n)    break;
    }


    ans = min(n, ans);


    printf("%lld\n", ans);
}
```

K Jay 的小迷弟

题解：

如果是一个连续的 Jay 字符串，它既可以被用来使碎片数量*2，也可以被用来使碎片数量+1。因此对于出现的连续的 Jay 字符串，我们单独用一个值来保存其出现次数，而对于字符 J,a,y 的个数，只要取三者的最小值即是未计算连续的 Jay 字符串时碎片的数量。如果此时碎片数量为 0 且连续的 Jay 字符串数量不为 0，则我们先用一个连续的 Jay 字符串给碎片数量

+1，剩下的只要*2 即可。

标程：

```
#include <bits/stdc++.h>
typedef long long LL;
const LL mod = 1e9+7;
const LL maxn = 1e8+5;
using namespace std;
char ch[maxn];
LL n;
LL cnt_Jay,cnt_J,cnt_a,cnt_y;
LL fpow(LL a,LL b)
{
    LL res=1;
    a%=mod;
    while(b)
    {
        if(b&1) res=res*a%mod;
        b>>=1;
        a=a*a%mod;
    }
    return res;
}


int main(){
     LL i,j;
     scanf("%s",ch+1);
     n=strlen(ch+1);
     for(i=1;i<=n;i++)
     {
         if(i<=n-2 && ch[i]=='J' && ch[i+1]=='a' && ch[i+2] =='y')
         {
             cnt_Jay++;
             i+=2;
         }
         else if(ch[i]=='J')
```

```
            cnt_J++;
        else if(ch[i]=='a')
            cnt_a++;
        else if(ch[i]=='y')
            cnt_y++;
    }
    LL ans=0;
    ans=min(cnt_J,min(cnt_a,cnt_y));
    if(ans==0&&cnt_Jay>0)
    {
        ans=1;
        cnt_Jay--;
    }
    ans%=mod;
    ans=ans*fpow(2,cnt_Jay)%mod;
    printf("%lld\n",ans);
}
```

L 翻转卡片

题解：

分数取模需要提前了解费马小定理 + 快速幂

考虑动态规划，此题状态定义的方式有很多种。我们定义 dp[i][j][k]代表：i 张卡片，已经翻转了 k 张的方法数，j == 0 表示最后一张卡片未翻转，j == 1 表示最后一张卡片已翻转。显然 dp[i][0][k] = dp[i - 1][1][k]，dp[i][1][k] = dp[i - 1][0][k - 1] + dp[i - 2][0][k - 1]。那么 i 张卡片翻转 k 张的总方法数就等于(dp[i][0][k] + dp[i][1][k]) * fact(k)，其中 fact(k)表示 k 的阶乘，接下来计算期望即可。

标程：

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

const int mod = (int) 1e9 + 7;

long long mypow(long long a, long long n) {
```

```
    long long ans = 1;
    while (n) {
        if (n & 1) ans = ans * a % mod;
        a = a * a % mod;
        n >>= 1;
    }
    return ans;
}


long long dp[10000][2][5000];
long long cnt[10000] = {0, 1, 2};
long long expect[10000];
long long fact[10000];
int main() {
    fact[0] = fact[1] = 1;
    for (int i = 2; i <= 1000; ++i) {
        fact[i] = fact[i - 1] * i % mod;
    }
    dp[1][1][1] = 1;
    dp[2][1][1] = 1;
    dp[2][0][1] = 1;
    for (int i = 3; i <= 1000; ++i) {
        for (int j = 1; j <= (i + 1) / 2; ++j) {
            dp[i][0][j] += dp[i - 1][1][j];
            dp[i][1][j] += dp[i - 1][0][j - 1];
            dp[i][1][j] += dp[i - 2][0][j - 1];
            dp[i][0][j] %= mod;
            dp[i][1][j] %= mod;
            cnt[i] += ((dp[i][0][j] + dp[i][1][j]) % mod * fact[j] % mod);
            cnt[i] %= mod;
        }
    }
    for (int i = 1; i <= 1000; ++i) {
        for (int j = 0; j <= (i + 1) / 2; ++j) {
            expect[i] += ((dp[i][0][j] + dp[i][1][j]) % mod * fact[j] % mod) * j % mod;
            expect[i] %= mod;
```

```
        }
        expect[i] = expect[i] * mypow(cnt[i], mod - 2) % mod;
    }
    int tt;
    cin >> tt;
    while (tt--) {
        int n;
        cin >> n;
        cout << expect[n] << '\n';
    }
    return 0;
}
```