A.另一个爱与希望的故事

简单递推，对斐波那契数列略微修改，开一个数组记录当前台阶有没有损坏，循环的时候如果发现损坏就continue

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int maxn=100000+10;

ll x[maxn],ans[maxn];
ll mod=1000000007;


int main()
{
    //freopen("test0.in","r",stdin);
    //freopen("test0.out","w",stdout);
    int t; cin>>t;
    while(t--)
    {
        int n,k; cin>>n>>k;
        memset(x,0,sizeof(x));
        memset(ans,0,sizeof(ans));
        for(int i=0;i<k;i++)
        {
            int num; cin>>num;
            x[num]=1;
        }
        ans[0]=1;
        for(int i=1;i<=n;i++)
        {
            if(x[i]) continue;
            if(i==1) ans[i]=ans[i-1];
            else ans[i]=ans[i-1]+ans[i-2];
            ans[i]%=mod;
        }
        cout<<ans[n]<<"\n";
    }
    return 0;
}
```

B.BanGosu!

按照题目意思一步一步模拟即可，我们判断距离时可以使用不开根号的形式，这样就能减少误差

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
vector<pair<double,double> > circle;
double distence(double x1 , double y1 , double x2 , double y2) {
```

```
    return (x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2);
}
double getcombo(int x) {
    if(x >= 0 && x < 100)return 1.0;
    else if(x >= 100 && x < 200)return 1.01;
    else if(x >= 200 && x < 300)return 1.02;
    else if(x >= 300 && x < 400)return 1.03;
    else return 1.04;
}
int main() {
    int n;cin >> n;
    double r;cin >> r;
    for(int i = 0 ; i < n ; i ++) {
        double x,y;cin >> x >> y;
        circle.push_back(make_pair(x , y));
    }
    int combo = 0;
    double res = 0;
    for(int i = 0 ; i < n ; i ++) {
        double x,y;cin >> x >> y;
        double dis = distence(x , y , circle[i].first , circle[i].second);
        if(dis < 0.04 * r * r) {
            combo ++;
            double add = getcombo(combo);
            res += 300.0*add;
        }
        else if(dis < 0.25 * r * r) {
            combo ++;
            double add = getcombo(combo);
            res += 200.0*add;
        }
        else if(dis < r * r) {
            combo ++;
            double add = getcombo(combo);
            res += 100.0*add;
        }
        else {
            combo = 0;
        }
    }
    int ans = (int)res;
    cout << ans;
}
```

C.奇怪的引擎

对W(t)求导，得到$P(t) = \frac{1}{2}x^{\frac{3}{2}} + sin(x)$，不难发现这个函数在$(0, +\infty)$上单调递增，所以我们对着这个函数二分即可

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

double cal(double t) { return sqrt(t * t * t) / 2.0 + sin(t); }
double solve(double p)
{
```

```
    int cnt = 1000;
    double l = 0, r = 1e18;
    while (cnt--)
    {
        double mid = (l + r) / 2.0;
        if (cal(mid) < p)
            l = mid;
        else
            r = mid;
    }
    return l;
}

int main()
{
    int t;
    cin >> t;
    while (t--)
    {
        double p;
        cin >> p;
        double ans = solve(p);
        cout << fixed << setprecision(10) << ans<<"\n";
    }
    return 0;
}
```

D.Diana压缩算法

由于题目要求三个相邻的字符表示一个英文字母，并且要求字典序最小。那么我们定义一个字符变量 Ch = 'a' ，之后我们从前到后遍历一遍数组，如果发现某三个相邻字符没有对应的英文字母，那么就让 Ch 去对应它，然后输出Ch，并且令Ch = Ch + 1；反之，如果发现某三个相邻字符已经有了对应的英文字母，那么直接输出其对应的字母即可。这样就能保证字典序最小，而且我们最多只会使用 8 个英文字母。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int maxn=100000+10;

char ans[10];

int getnum(string s)
{
    int res=0;
    for(int i=2;i>=0;i--) res=res*2+(s[i]-'0');
    return res;
}

int main()
{
    for(int i=0;i<10;i++) ans[i]='#';
    int n; cin>>n;
    n*=3;
    string s; cin>>s;
```

```
    int alpha=0;
    for(int i=0;i<n;i+=3)
    {
        string ss;
        ss+=s[i],ss+=s[i+1],ss+=s[i+2];
        int num=getnum(ss);
        if(ans[num]=='#')
        {
            ans[num]='a'+alpha;
            alpha++;
        }
        cout<<ans[num];
    }
    cout<<"\n";
    return 0;
}
```

E.子矩形

我们要注意到最多只有10中字符，所以子矩形的大小不可能超过10，否则一定有重复元素；所以我们可以直接暴力求解，控制子矩形的大小，然后定位子矩形，判断其中是否有重复元素。我们可以使用标程的 6 重循环，也可以使用分类讨论求解

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 510;
int a[N][N],numUse[10],n,m;
//判断以(x,y)为左上角，长宽分别为lenx和leny的子矩形是否有重复元素
int getNum(int x,int y,int lenx,int leny) {
    memset(numUse , 0 , sizeof numUse);
    int flag = 0;
    for(int i = x ; i <= x + lenx - 1 ; i ++) {
        for(int j = y ; j <= y + leny - 1 ; j ++) {
            if(i <= n && j <= m && numUse[a[i][j]] == 0) {
                numUse[a[i][j]] = 1;
            }
            else {
                flag = 1;
                break;
            }
        }
        if(flag)break;
    }
    if(!flag)return 1;
    return 0;
}
/*分类讨论，只有
1*1、1*2……1*10
2*1、3*1……10*1
2*2、2*3、3*2、2*4、4*2、2*5、5*2
3*3
种子矩形
*/
int getDifferent(int x,int y){
    int res = 0;
```

```
    for(int i = 1 ; i <= 10 ; i ++) {
        res += getNum(x , y , 1 , i);
        res += getNum(x , y , i , 1);
    }
    res += getNum(x , y , 2 , 2);
    res += getNum(x , y , 2 , 3);
    res += getNum(x , y , 3 , 2);
    res += getNum(x , y , 2 , 4);
    res += getNum(x , y , 4 , 2);
    res += getNum(x , y , 2 , 5);
    res += getNum(x , y , 5 , 2);
    res += getNum(x , y , 3 , 3);
    return res;

}
int main() {
    cin >> n >> m;
    for(int i = 1 ; i <= n ; i ++)
        for(int j = 1 ; j <= m ; j ++)
            cin >> a[i][j];
    int res = 0;
    for(int i = 1 ; i <= n ; i ++) {
        for(int j = 1 ; j <= m ; j ++) {
            res += getDifferent(i , j);
        }
    }
    //前面重复算了n*m个大小为1*1的子矩形，所以减去
    cout << res - n * m << endl;
    return 0;
}
```

F.天天爱跑步

结构体存一下信息，然后按用时升序排序，若用时相同，则按照输入顺序排序即可。很多同学排序的时候没有考虑用时相同的时候按输入顺序排序，这个可以多加一个关键字进行排序，也可以直接使用稳定排序，题目里的n最大5000，其实就是方便大家冒泡排序的。

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 5e3+10;
struct INFO {
    string name;
    int time;
    bool operator > (const INFO &a) const {
        return time>a.time;
    }
} info[maxn];
int main() {
    int n; cin >> n;
    for (int i = 1; i<=n; ++i) {
        string name; cin >> name;
        int h, m, s; scanf("%d:%d:%d", &h, &m, &s);
        info[i] = {name, h*3600+m*60+s};
    }
    for (int i = 1; i<n; ++i)
        for (int j = 1; j<=n-i; ++j) {
```

```
            if (info[j]>info[j+1]) swap(info[j], info[j+1]);
        }
    for (int i = 1; i<=n; ++i) cout << info[i].name << endl;
    return 0;
}
```

G.link-cut-graph

题目的灵感来自于abc中的一道题。首先求出1到n的最短路径经过的边并标记起来，由于总共最多删除m条边，并且每条边被删除的概率是随机的，所以如果我们每次删掉被标记的边，就重新跑一遍最短路的话， 并且假设一个比较坏的情况，从1到n的最短路径要经过n条边的话，跑最短路的期望次数也不过$n+1$次，总的时间复杂度的期望是n^3。

```
#include <bits/stdc++.h>
using namespace std;
#define INF 0x3f3f3f3f
#define endl '\n'
typedef long long ll;
typedef pair<int,int> P;
const int maxn = 5e2+10;
const int maxm = maxn*maxn;
int g[maxn][maxn], mp[maxn][maxn];
struct INFO {
    int u, v, w;
} info[maxm];
bool vis[maxn], flag[maxm];
int n, m, s, d[maxn], p[maxn];
void dij() {
    memset(d, 0x3f, sizeof(d));
    memset(vis, 0, sizeof(vis));
    d[1] = 0;
    for (int i = 1; i<n; ++i) {
        int id = -1;
        for (int j = 1; j<=n; ++j)
            if (!vis[j] && (id==-1 || d[id]>d[j])) id = j;
        vis[id] = 1;
        for (int j = 1; j<=n; ++j)
            if (d[j]>d[id]+g[id][j]) {
                d[j] = d[id]+g[id][j];
                p[j] = id;
            }
    }
}
int main() {
    ios::sync_with_stdio(false);
    cin.tie();
    cout.tie();
    cin >> n >> m >> s;
    memset(g, 0x3f, sizeof(g));
    for (int i = 1, u, v, w; i<=m; ++i) {
        cin >> u >> v >> w;
        info[i] = {u, v, w};;
        mp[u][v] = mp[v][u] = i;
        g[u][v] = g[v][u] = w;
    }
    dij();
```

```
    int x = d[n];
    if (x==INF) x = -1;
    int ed = n;
    while(ed) {
        if (p[ed]) flag[mp[ed][p[ed]]] = 1;
        ed = p[ed];
    }
    for (int i = 1; i<=s; ++i) {
        vector<int> tmp;
        int num;
        cin >> num;
        int edge, f = 0;
        while(num--) {
            cin >> edge;
            if (flag[edge]) f = 1;
            INFO &t = info[edge];
            g[t.u][t.v] = g[t.v][t.u] = INF;
            tmp.push_back(edge);
        }
        if (!f) cout << x << endl;
        else {
            dij();
            if (d[n] == INF) d[n] = -1;
            cout << d[n] << endl;
        }
        for (auto v: tmp) {
            INFO &t = info[v];
            g[t.u][t.v] = g[t.v][t.u] = t.w;
        }
    }
    return 0;
}
```

H.奇怪的加法问题

xor可以理解为不进位的加法，而且本题求得是MOD2，所以考虑最后一位即可。

思路一：观察该公式，发现每个元素都出现了n - 1次，那么直接把所有元素加起来乘以n - 1然后再判奇偶即可

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int maxn=100000+10;

int main()
{
    ll n,ans=0; cin>>n;
    for(int i=0;i<n;i++)
    {
        ll num; cin>>num;
        ans+=num*(n-1);
        ans%=2;//在过程中MOD2等同于在最后MOD2
    }
    cout<<ans<<"\n";
```

```
    return 0;
}
```

思路二：计算有多少个$a_i + a_j$是奇数，因为只有奇数与奇数进行xor运算才会影响结果；假如有ans个$(ai + aj)$是奇数，如果ans为奇数，那么结果就是1，反之为0

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
int main() {
    int n;cin >> n;
    int sum = 0;
    for(int i = 1 ; i <= n ; i ++) {
        int x;cin >> x;
        if(x % 2)
            sum ++;//计算输入奇数的个数
    }
    ll ans = (n - sum) * sum;//计算ai + aj中奇数的个数
    if(ans % 2)cout << 1 << endl;
    else cout << 0 << endl;
}
```

I.奇怪的加法问题

可以枚举b的质因数x，然后根据x算出比a大的x的最小倍数。

```
# include "bits/stdc++.h"
using namespace std;
using ll = long long;
const int maxn = 1e5+10;
const int MOD = 1e9+7;
int dp[maxn], flag[maxn];
int main () {
    ios::sync_with_stdio(0);
    int __; cin >> __;
    while(__--) {
        ll a, b; cin >> a >> b;
        ll ans = (a+b-1)/b*b;
        for (ll i = 2; i*i<=b; ++i) {
            if (b%i==0) {
                ans = min(ans, (a+i-1)/i*i);
                ll j = b/i;
                ans = min(ans, (a+j-1)/j*j);
                //cout << i << ' ' << ans << endl;
            }
        }
        cout << ans-a << endl;
    }
    return 0;
}
```

J.扫描线

这题算是歪榜比较厉害的题了，其实只要直接枚举两种斜率的直线切割矩形的情况就行了。

解法一：出题人的解法，直接模拟，比较烂，建议直接看解法二

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1e3+10;
int n, g[maxn][maxn];
int calc0(int i, int j) {
    //cout << "-----1-----" << endl;
    int sum = 0;
    while(i<=n && j>=1) {
        sum += g[i][j];
        //cout << i << ' ' << j << endl;
        ++i, --j;
    }
    return sum;
}
int calc1(int i, int j) {
    //cout << "----2-------" << endl;
    int sum = 0;
    while(i<=n && j<=n) {
        sum += g[i][j];
        //cout << i << ' ' << j << endl;
        ++i, ++j;
    }
    return sum;
}
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    cin >> n;
    int sum0 = 0;
    for (int i = 1; i<=n; ++i)
        for (int j = 1; j<=n; ++j)
            cin >> g[i][j], sum0 += g[i][j];
    int ans = 1e9;
    int sum1 = 0;
    for (int i = 1; i<=n; ++i) {
        int x = calc0(1, i);
        sum1 += x;
        //cout << sum0-sum1 << ' ' << sum1-x << endl;
        ans = min(ans, abs(sum0-sum1*2+x));
    }
    for (int i = 2; i<=n; ++i) {
        int x = calc0(i, n);
        sum1 += x;
        //cout << sum0-sum1 << ' ' << sum1-x << endl;
        ans = min(ans, abs(sum0-sum1*2+x));
    }
    for (int i = 1; i<=n; ++i) reverse(g[i]+1, g[i]+n+1);
    sum1 = 0;
    for (int i = 1; i<=n; ++i) {
        int x = calc0(1, i);
        sum1 += x;
        //cout << sum0-sum1 << ' ' << sum1-x << endl;
        ans = min(ans, abs(sum0-sum1*2+x));
    }
```

```
    for (int i = 2; i<=n; ++i) {
        int x = calc0(i, n);
        sum1 += x;
        //cout << sum0-sum1 << ' ' << sum1-x << endl;
        ans = min(ans, abs(sum0-sum1*2+x));
    }
    cout << ans << endl;
    return 0;
}
```

解法二：优秀选手的做法，把矩阵分别左旋45°和右旋45°，然后用前缀和计算

```
#include <bits/stdc++.h>
using namespace std;
int n;
int a[1010][1010];
int l[2050],r[2050];
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            scanf("%d",&a[i][j]);
            l[i+j]+=a[i][j];
            r[j-i+n+1]+=a[i][j];
        }
    }
    for(int i=2;i<=2*n;i++)l[i]+=l[i-1];
    for(int i=2;i<=2*n;i++)r[i]+=r[i-1];
    int ans=0x7fffffff;
    for(int i=2;i<=2*n;i++)ans=min(ans,abs(l[i-1]-(l[2*n]-l[i])));
    for(int i=2;i<=2*n;i++)ans=min(ans,abs(r[i-1]-(r[2*n]-r[i])));
    printf("%d",ans);
    return 0;
}
```

K.黄金戟

语法题

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> P;
const int maxn=100000+10;

int main()
{
    //freopen("test0.in","r",stdin);
    //freopen("test0.out","w",stdout);
    int t; cin>>t;
    while(t--)
    {
        int a,b,c; cin>>a>>b>>c;
        if(a>=30&&b>=14&&c>=12) cout<<"Yes\n";
        else if(a>=20&&b>=14&&c>=12) cout<<"ulii\n";
        else cout<<"No\n";
```

```
    }
    return 0;
}
```