1 pA

括胡匹配給定一串左右括胡問要修改多少個括胡才能使其匹配

策略遇到左括胡 +1, 右括胡 -1 如果當前的 value 低於 0 代表左括胡數量不足則反轉此括胡最後若 value 不為 0 代表右括胡不足反轉 value/2 個括胡

```cpp
#include <bits/stdc++.h>
using namespace std;
char str[128];
int main(){
    int t;
    scanf("%d", &t);
    while(t--){
        int n;
        scanf("%d%s", &n, str);
        int now = 0, ans = 0;
        for(int i = 0 ; i < n ; i++){
            if(str[i] == '(') now++;
            else{
                now--;
                if(now < 0)
                    now += 2, ans++;
            }
        }
        printf("%d\n", ans + now / 2);
    }
}
```

2 pB

將給定的 matrix 扣掉單位矩陣後
做高斯消去如果斜對角有一個數為 0 則答案為 0 否則為 1

3 pC

給定一張圖問有幾個點是關鍵點因為 n 很小 (n<=100) 所以可以直接枚舉並直接走完

```cpp
#include <bits/stdc++.h>
using namespace std;
#define f first
#define s second
typedef pair<int, int> PII;
namespace DS{
    int v[128];
    int sz[128];
    void init(int n){
        for(int i = 1 ; i <= n ; i++){
            v[i] = i;
            sz[i] = 1;
        }
    }
    int find(int x){
        return x == v[x] ? v[x] : v[x] = find(v[x]);
    }
    void merge(int x, int y){
        x = find(x), y = find(y);
        if(x == y) return;
        v[x] = y;
        sz[y] += sz[x];
        sz[x] = 0;
    }
};
void Solve(int n){
    int s, m, x, y;
    scanf("%d%d", &s, &m);
    vector<PII> ed;
    for(int i = 0 ; i < m ; i++){
        scanf("%d%d", &x, &y);
        ed.push_back(PII(x, y));
    }
    int ans, ans_num = 0x3f3f3f3f;
    for(int i = 1 ; i <= n ; i++){
        if(i == s) continue;
        DS::init(n);
        for(int j = 0 ; j < m ; j++){
            int u = ed[j].f, v = ed[j].s;
            if(u == i || v == i) continue;
            DS::merge(u, v);
        }
        int num = DS::sz[DS::find(s)];
        if(num < ans_num){
            ans_num = num;
            ans = i;
        }
    }
    printf("%d\n", ans);
}

int main(){
    int n;
    while(scanf("%d", &n), n)
        Solve(n);
    return 0;
}
```

4 pD

給定一質數 p 為模數對於每筆測資 a, b 詢問是否存在 x 使得 $a^x = b$ 若有輸出 x 否則輸出 0

策略: 因為 mod 一數根據鴿籠原理枚舉到第 p+1 次一定會重複故只需枚舉到 p 次就可 p 次內若無出現 b 則無解否則輸出第一次出現 b 的時候

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

int main(){
    int p;
    int a, b;
    scanf("%d", &p);
    while(scanf("%d", &a), a != 0){
        scanf("%d", &b);
        int now = 1, flag = 0;
        a %= p;
        b %= p;
        for(int i = 1 ; i < p ; i++){
            now *= a;
            now %= p;
            if(now == b){
                printf("%d\n", i);
                flag = 1;
                break;
            }
        }
        if(!flag)
            printf("0\n");
    }

}
```

5 pG

給定一二元樹和一起點每次挑選 cost 最小的一個點加到集合裡若 cost 一樣則挑選 id 較小的問挑選點的順序

策略: 開一個 priority_queue 來挑選下一個要加入的點開一張 table 來記錄這個點是否進過 priority_queue 了每加入一個點檢視其 parent, left child, right child 是否在範圍內如果在範圍內且不在 priority_queue 裡則加入 priority_queue, 並更新 table

```cpp
#include <bits/stdc++.h>
using namespace std;
struct S{
    int id, value;
    S(){}
    S(int _id, int _value) : id(_id), value(_value){}
    bool operator<(const S &b) const {
        return value == b.value ? id > b.id : value < b.value;
    }
};
int n, s, m;
priority_queue<S> PQ;
bool use[501234];

void add(int x, int y){
    if(x == 0 || x > n || use[x]) return;
    use[x] = 1;
    PQ.push(S(x, abs(x - y) % m));
}

void Solve(){
    memset(use, 0, sizeof(use));
    scanf("%d%d%d", &n, &s, &m);
    PQ.push(S(s, 0));
    use[s] = 1;
    while(PQ.size()){
        printf("%d ", PQ.top().id);
        int u = PQ.top().id;
        PQ.pop();
        add(u/2, u);
        add(u*2, u);
        add(u*2+1, u);
    }
    printf("\n");
}

int main(){
    int t;
    scanf("%d", &t);
    while(t--)
        Solve();
    return 0;
}
```

6 pH

平面上給定 n(n<=250) 個點點有兩種問存不存在一條線可以將兩種點分開

策略: 枚舉同一種點的任兩點做一直線若此線兩邊的任一邊存在兩種點代表此線不可能為答案故不考慮以下 case 若此線上只有與枚舉點相同則此線可成功使點分開若此線上存在兩種點則檢查此線上兩種點分布是否分開若分開 (如 ooooxxxx) 則可以在交換處附近找到斜率接近的線將之分開若交錯出現 (如 ooxoxxx) 則此線不可能為答案

7 pI

給定一圖的關係 (U,J) 現在要在點上面編號規定對於編號相同的點則其路徑上至少要有一點編號大於他們

策略: 點的數量 10000 個不可能把圖建起來觀察可以發現可以維護一個 (num, size) 代表這張圖至少要編到幾號以及圖上點的數量對於 U 操作可以得到 (max(num1, num2), size1+size2) 對於 J 操作可以得到 (min(num1+size2, num2+size1), size1+size2)

```cpp
#include <bits/stdc++.h>
using namespace std;
#define f first
#define s second
typedef pair<int, int> PII;
int p, len;
char str[1048576];

PII dfs(){
    if(str[p] == '('){
        p++;
        PII a = dfs();
        char op = str[p];
        p++;
        p++;
        PII b= dfs();
        p++;
        if(op == 'U'){
            return PII(max(a.f, b.f), a.s+b.s);
        } else {
            return PII(min(a.f+b.s, a.s+b.f), a.s+b.s);
        }
    } else {
        while(isdigit(str[p]))
            p++;
        p++;
        return PII(1, 1);
    }
}

void Solve(){
    p = 0;
    int n;
    scanf("%d", &n);
    scanf("%s", str);
    len = strlen(str);
    printf("%d\n", dfs().f);
}

int main(){
    int n;
    scanf("%d", &n);
    while(n--)
        Solve();
    return 0;
}
```