牛客小白月赛43

A 满意的数字

第m个因子就为本身,所以本身一定能被任何一个因子整除

```
//
int main(){
    int t,n;
    scanf("%d",&t);
    for(int i = 1;i <= t;i++){
        scanf("%d",&n);
        printf("%d\n",n);
    }

    return 0;
}
```

B 牛牛变魔术

首先处理 a == tar || b == tar ,每次操作后a b翻倍后必为偶数
所以tar为奇数时,a b通过操作必不可得
当tar为偶数时,通过n次操作使**a+b** > **tar**过程中,一定能通过操作获得**tar**含量.

```cpp
using ll = long long;
ll a,b,tar,temp;
int T,cnt;

int main(){
    scanf("%d",&T);
    while(T-- > 0){
        scanf("%lld %lld %lld",&a,&b,&tar);
        //
        if(a == tar || b == tar){
            printf("0\n");
            continue;
        }
        //
        if(tar&1){
            printf("-1\n");
            continue;
        }
        temp = (a + b)<<1;
        cnt = 1;
        while(temp < tar){
            cnt++;
            temp <<= 1;
        }

        printf("%d\n",cnt);
    }

    return 0;
}
```

C 木棍游戏

因为3 <= n <= 8
数据量不大,可以考虑暴力搜索
每根木棍有4个选择: 1.给a 2.给b 3.给c 4.不选

```cpp
int n,flag = 1;
double ans = INT32_MIN;
int len[10] = {0};

void get_ans(double a,double b,double c){
    double p = (a+b+c)/2;
    double S = sqrt(p*(p-a)*(p-b)*(p-c));
    ans = std::max(ans,S);
}

void dfs(int a,int b,int c,int i){
    if(a+b > c && a+c > b && b+c > a){
        get_ans(a,b,c);
    }
    if(i > n+1)
        return;
    dfs(a+len[i],b,c,i+1);
    dfs(a,b+len[i],c,i+1);
    dfs(a,b,c+len[i],i+1);
    dfs(a,b,c,i+1);
}

int main(){
    int temp;
    scanf("%d",&n);
    for(int i = 1;i <= n;i++)
        scanf("%d",&len[i]);
    dfs(0,0,0,1);
    if(ans <= 0)
        printf("-1\n");
    else
        printf("%.1f\n",ans);
    return 0;
}
```

D 有趣的区间

从题意看,区间只要包含奇数即为"有趣的区间"
因此从左向右依次求出区间数量:
奇数: n-i+1;
偶数: n-(右边最近的奇数位置)i+1;

```
const int max_N = 5e5+10;

int array[max_N],next_odd[max_N];
long long ans = 0;

int main(){
    int n;
    scanf(" %d",&n);
    for(int i = 1;i <= n;i++){
        scanf(" %d",&array[i]);
        array[i] = array[i] & 1;
    }
    int pos_odd = n+1;
    for(int i = n;i >= 1;i--){
        if(array[i])
            pos_odd = i;
        next_odd[i] = pos_odd;
    }
    for(int i = 1;i <= n;i++){
        if(array[i])
            ans += n-i+1;
        else
            ans += n-next_odd[i]+1;
    }
    printf("%lld\n",ans);
    return 0;
}
```

E 满意的集合

从单种数字看:
取余为 1 的集合取法有:

```
mod_1 = (cnt[i] + 2) / 3;
mod_2 = (cnt[i] + 1) / 3;
mod_0 = cnt[i] / 3;
```

取余为 2 的集合取法有:

```
mod_1 = (cnt[i] + 1) / 3;
mod_2 = (cnt[i] + 2) / 3;
mod_0 = cnt[i] / 3;
```

取余为 0 的集合取法有:

```
mod_1 = 0;
mod_2 = 0;
mod_0 = cnt[i];
```

用num_mod[i][mod] 表示前i个数字组成的余数为**mod**的集合个数
可以得到递推公式

```
num_mod[i][0] = num_mod[i-1][0] * (mod_0+1) + num_mod[i-1][1] * (mod_2) + num_mod[i-1][2] * mod_
num_mod[i][1] = num_mod[i-1][1] + num_mod[i-1][0] * mod_1 + num_mod[i-1][1] * mod_0 + num_mod[i-
num_mod[i][2] = num_mod[i-1][2] + num_mod[i-1][0] * mod_2 + num_mod[i-1][1] * mod_1 + num_mod[i-
```

就可以找出"满意的集合"个数

```
using ll = long long;

const int n = 10, _mod = 1e9 + 7;

ll cnt[n];
ll num_mod[n][3];

int main(){
    for(int i = 1;i <= 9;i++)
        scanf(" %d",&cnt[i]);

    for(int i = 0;i <= 9;i++)
        num_mod[i][0] = 1;

    for(int i = 1;i <= 9;i++){
        int mod_0, mod_1, mod_2;
        if(i % 3 == 1){
            mod_1 = (cnt[i] + 2) / 3;
            mod_2 = (cnt[i] + 1) / 3;
            mod_0 = cnt[i] / 3;
        }
        else if(i % 3 == 2){
            mod_1 = (cnt[i] + 1) / 3;
            mod_2 = (cnt[i] + 2) / 3;
            mod_0 = cnt[i] / 3;
        }
        else{
            mod_1 = 0;
            mod_2 = 0;
            mod_0 = cnt[i];
        }

        num_mod[i][0] = num_mod[i-1][0] * (mod_0+1) + num_mod[i-1][1] * (mod_2) + num_mod[i-1][2
        num_mod[i][1] = num_mod[i-1][1] + num_mod[i-1][0] * mod_1 + num_mod[i-1][1] * mod_0 + nu
        num_mod[i][2] = num_mod[i-1][2] + num_mod[i-1][0] * mod_2 + num_mod[i-1][1] * mod_1 + nu

        num_mod[i][0] %= _mod;
        num_mod[i][1] %= _mod;
        num_mod[i][2] %= _mod;
    }

    printf("%lld\n",num_mod[9][0]);
    return 0;
}
```

F 全体集合

如果该无向图不构成二分图,则一定存在一个奇数点位能让所有人同时到达
如果构成二分图:
1.若所有人都在一边中,则可以集中在一点
2.若所有人不都在一边,则不可能集中至一点

```cpp
const int max_n = 2e5+10;

std::vector<int> edge[max_n];

int color[max_n];
int pos_[max_n];
int n,m,k,flag = 0;

void graph_judge(int node,int group){
    if(flag)
        return;
    color[node] = group;
    for(int i = 0;i < edge[node].size();i++){
        int node2 = edge[node][i];
        if(color[node2] != 0 && color[node2] == group){   //非二分图
            flag = 1;
            return;
        }
        if(color[node2])
            continue;
        graph_judge(node2,3 - group);
    }
}

int main(){
    int node1,node2;
    scanf(" %d %d %d",&n,&m,&k);
    for(int i = 1;i <= m;i++){
        scanf(" %d %d",&node1,&node2);
        edge[node1].push_back(node2);
        edge[node2].push_back(node1);
    }
    for(int i = 1;i <= k;i++)
        scanf(" %d",&pos_[i]);
    graph_judge(1,1);
    if(flag)
        printf("YES\n");
    else{
        int group = color[pos_[1]];
        for(int i = 2;i <= k;i++){
            if(color[pos_[i]] != group){
                printf("NO\n");
                return 0;
            }
        }
        printf("YES\n");
    }
    return 0;
}
```