# 2024 CSP-J 复赛完整版解析（含标程）云斗学院版

## T1.扑克牌

性质A：不重复，输出 $52-n$ 即可。

性质B：相同牌会一起给出，判断与前一张牌是否相同即可去重。

正解：使用哈希/unordered_map/先排序进行去重。答案为 $52$ 减去去重后的牌数。

```
#include<bits/stdc++.h>
using namespace std;
unordered_map<string, int> ma;
int main()
{
    int n; cin >> n;
    int ans = 52;
    for(int i = 1; i <= n; i ++)
    {
        string s; cin >> s;
        if(!ma.count(s)) ans --;
        ma[s] = 1;
    }
    cout << ans << "\n";
    return 0;
}
```

## T2.地图探险

$k=1$ ：判断下一步能否走得动即可。

均为空地：处理好边界处转弯即可。

正解：首先要判断下一步能否走动，记录 d = 0/1/2/3 对应的方向数组 dx, dy，如果下一步是空地，那么进行移动，否则进行转向。同时开一个数组记录所有走过的位置，最后统计即可。记得多测清空。

```
#include<bits/stdc++.h>
using namespace std;
int dx[5] = {0,1,0,-1};
int dy[5] = {1,0,-1,0};
int mp[1020][1020];
int vis[1020][1020];
int main()
{
    int t; cin >> t;
    while(t --)
    {
        int n, m, k; cin >> n >> m >> k;
```

```
        int x, y, d; cin >> x >> y >> d;
        for(int i = 1; i <= n; i ++)
        {
            string s; cin >> s;
            for(int j = 0; j < m; j ++)
            {
                mp[i][j + 1] = (s[j] == 'x');
                vis[i][j + 1] = 0;
            }
        }
        int cnt = 1;
        vis[x][y] = 1;
        while(k --)
        {
            int tx = x + dx[d];
            int ty = y + dy[d];
            if(tx > 0 && tx <= n && ty > 0 && ty <= m && !mp[tx][ty])
            {
                x = tx, y = ty;
                if(!vis[x][y])
                {
                    vis[x][y] = 1;
                    cnt ++;
                }
            }
            else
            {
                d ++;
                if(d == 4) d = 0;
            }
        }
        cout << cnt << "\n";
    }
    return 0;
}
```

## T3.小木棍

$n < 50$：手玩 / 暴搜 / 打表的分。

性质A：我们首先希望位数尽量的少，显然如果全是7的倍数，放一堆 8 是位数最小的。任何一种其他的方式位数都会更多。

性质B：我们发现最少的位数只能是 $k+1$，但是单独出来的一根无法处理，我们可以把最高位的 8 变为 0，从而省下一根来凑 1放在最高位。

正解: 性质AB提示了我们要对余数进行讨论。类似的，我们可以将 8 变为 0 来凑出首位。(具体凑法见std中注释)。

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

```cpp
using namespace std;
int a[10] = {6,2,5,5,4,5,6,3,7,6};
/*
0 6
1 2 剩 1 / 2
2 5 剩 3 / 4 / 5
3 5 不如2
4 4 不如2
5 5 不如2
6 6 剩 6
7 3
8 7
9 6 不如6
*/

int main()
{
    int t;
    cin >> t;
    while(t --)
    {
        int n;
        cin >> n;
        int res = n % 7;
        int num = n / 7;
        if(n == 1)
        {
            cout << -1 << "\n";
            continue;
        }
        if(n == 2)
        {
            cout << 1 << "\n";
            continue;
        }
        if(n == 3)
        {
            cout << 7 << "\n";
            continue;
        }
        if(n == 4)
        {
            cout << 4 << "\n";
            continue;
        }
        if(n == 5)
        {
            cout << 2 << "\n";
            continue;
        }
        if(n == 6)
        {
            cout << 6 << "\n";
            continue;
```

```
        }
        if(n == 10)
        {
            cout << 22 << "\n";
            continue;
        }
        if(res == 0)
        {
            for(int i = 1; i <= num; i ++) cout << 8;
            puts("");
        }
        else if(res == 1)
        {
            cout << 10;
            for(int i = 1; i <= num - 1; i ++) cout << 8;
            puts("");
        }
        else if(res == 2)
        {
            cout << 1;
            for(int i = 1; i <= num; i ++) cout << 8;
            puts("");
        }
        else if(res == 3)
        {
            cout << 200;
            for(int i = 1; i <= num - 2; i ++) cout << 8;
            puts("");
        }
        else if(res == 4)
        {
            cout << 20;
            for(int i = 1; i <= num - 1; i ++) cout << 8;
            puts("");
        }
        else if(res == 5)
        {
            cout << 2;
            for(int i = 1; i <= num; i ++) cout << 8;
            puts("");
        }
        else if(res == 6)
        {
            cout << 6;
            for(int i = 1; i <= num; i ++) cout << 8;
            puts("");
        }
    }
    return 0;
}
```

# T4.接龙

正解：观察数据范围，我们发现 $r$ 也就是比赛进行的轮数的范围很小。

新的一轮的接龙仅与上一轮接龙的末字符、上一轮接龙的人是否为自己有关，而与具体的人无关。

我们从上一轮信息转移，处理出当前轮次能否以字符 $c$ 结尾，并把对应的人的编号存在数组 a 中。如果有大于一人可以转移到字符 $c$，那么 a 的值记为 $-1$，因为下一轮一定能够被接龙。我们顺便更新 vis 数组，询问时直接查询即可。

问题在于如何处理当前轮次能否以字符 $c$ 结尾，由于有长度限制，我们需要判断当前字符能否以位置 $j$ 开始。而一个位置能作为起点，当且仅当上一轮中这个位置对应的字符能作为终点，且对应编号的人不是他。我们可以每轮结束后将这个信息更新在 tag 数组中。

a[c] 表示当前轮次能否以字符 $c$ 结尾。若能，储存对应人的编号；若大于一人，记为 $-1$；

vis[i][c] 表示第 $i$ 时刻能否以字符 $c$ 结尾。

tag[i][j]表示第下一时刻能否以第 $i$ 个人的第 $j$ 的位置作为起点.

具体实现细节见std。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 2.01e5;
int T, n, k, q;
int len[N], a[N];
vector<int> s[N], tag[N];
bitset<N> vis[101];
int read() {
    int x = 0;
    char c = getchar();
    while (c < '0' || c > '9')
        c = getchar();
    while ('0' <= c && c <= '9')
        x = x * 10 + c - '0', c = getchar();
    return x;
}
int main() {
    T = read();
    while (T--)
    {
        n = read(), k = read(), q = read();
        int sumL = 0;
        for (int i = 1; i <= n; i +)
        {
            len[i] = read();
            s[i].resize(len[i] + 1);
            tag[i].resize(len[i] + 1);
            for (int j = 1; j <= len[i]; j ++)
            {
                s[i][j] = read();
                if (s[i][j] == 1)
```

```
                    tag[i][j] = 1;
                else
                    tag[i][j] = 0;
            }
            sumL += len[i];
        }
//      printf("n=%d sumL=%d\n", n, sumL);
        for (int t = 1; t <= 100; t ++)
        {
            for (int i = 1; i < N; i ++)
                a[i] = 0, vis[t][i] = 0;
            for (int i = 1; i <= n; i ++)
            {
                int pre = -k;
                for (int j = 1; j <= len[i]; j ++)
                {
                    if (j - pre + 1 <= k)
                    {
                        if (!a[s[i][j]])
                            a[s[i][j]] = i;
                        else if (a[s[i][j]] != i)
                            a[s[i][j]] = -1;
                        vis[t][s[i][j]] = 1;
                    }
                    if (tag[i][j])
                        pre = j;
                }
            }
            for (int i = 1; i <= n; i++)
            {
                for (int j = 1; j <= len[i]; j++)
                {
                    if (a[s[i][j]] != 0 && a[s[i][j]] != i)
                        tag[i][j] = 1;
                    else
                        tag[i][j] = 0;
                }
            }
        }
        for (int i = 1, r, c; i <= q; i++)
        {
            r = read(), c = read();
            if (vis[r][c])
                putchar('1');
            else
                putchar('0');
            putchar('\n');
        }
    }
    return 0;
}
```