

# 全排列枚举+二进制枚举

## 人员

柳力玮、韩昱辰、田心一、刘宸熙、赵书梵、李瑞涵、李知朔、纪博涵、蒋叔璋、初锦阳 到课, 苑钊、高健桓 线上

## 上周作业检查

上周作业链接: <https://cppoj.kids123code.com/contest/1690>

📄 比赛概况

📋 题目列表

🏆 选择题列表

📝 提交记录

★ 实时榜单

★ 选择题排行榜

王向东老师周日三点半C++洪水填充搜索

刷新

#	用户名	姓名	编程分	时间	A	B	C	D	E	F	G
1	liuliwei	柳力玮	700	432	100	100	100	100	100	100	100
2	yuanzhao	苑钊	700	444	100	100	100	100	100	100	100
3	zhaoshufan	赵书梵	700	458	100	100	100	100	100	100	100
4	hanyuchen	韩昱辰	700	592	100	100	100	100	100	100	100
5	tianxinyi	田心一	700	675	100	100	100	100	100	100	100
6	liiruihan	李瑞涵	700	724	100	100	100	100	100	100	100
7	gaojianhuan	高健桓	680	224	100	100	100	100	100	100	80
8	lizhishuo	李知朔	600	223	100	100	100	100	100	100	
9	wenhaodong	温郝冬	500	181	100	100	100	100	100		
10	jibohan	纪博涵	483	160	100	100	100	83	100		
11	liuchenxi	刘宸熙	400	160	100	100	100		100		
12	jiangshuzhang	蒋叔璋	300	116	100	100	100				
13	chujinyang	初锦阳	140	40	100	40					

## 本周作业

<https://cppoj.kids123code.com/contest/1790> (课上讲了 A ~ F 题, 课后作业是 G 题)

## 课堂表现

这节课讲了 next\_permutation 和 二进制枚举 这两个内容, 这两个内容是不难的, 同学们课上基本掌握的都比较好。

## 课堂内容

### [NOI Online #3 入门组] 观星 (上周作业)

洪水填充算法 找每个联通块, 并需要在搜索过程中, 额外记录每个联通块的大小

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

using namespace std;

const int maxn = 1500 + 5;
char s[maxn][maxn];
```

```

bool st[maxn][maxn];
int dx[] = {-1, -1, -1, 0, 0, 1, 1, 1};
int dy[] = {-1, 0, 1, -1, 1, -1, 0, 1};
int cnt;
int n, m;

void dfs(int x, int y) {
    if (st[x][y]) return;
    st[x][y] = true;
    cnt++;

    for (int i = 0; i < 8; ++i) {
        int nx = x+dx[i], ny = y+dy[i];
        if (nx>=1 && nx<=n && ny>=1 && ny<=m && s[nx][ny]=='*') dfs(nx,ny);
    }
}

int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> (s[i]+1);

    map<int,int> mp;
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            if (s[i][j]=='*' && !st[i][j]) {
                cnt = 0;
                dfs(i,j);
                mp[cnt]++;
            }
        }
    }

    int res = 0;
    for (auto it : mp) res = max(res, it.first*it.second);
    cout << mp.size() << " " << res << endl;
    return 0;
}

```

## 全排列问题

next\_permutation 使用模板

```

#include <bits/stdc++.h>

using namespace std;

const int maxn = 10 + 5;
int w[maxn];

int main()

```

```

{
    int n; cin >> n;
    for (int i = 1; i <= n; ++i) w[i] = i;
    do {
        for (int i = 1; i <= n; ++i) printf("%5d", w[i]);
        cout << endl;
    } while (next_permutation(w+1, w+n+1));
    return 0;
}

```

## 枚举子集（递归实现指数型枚举）

### 二进制枚举

```

#include <bits/stdc++.h>

using namespace std;

int main()
{
    int n; cin >> n;
    for (int i = 0; i < (1<<n); ++i) {
        for (int j = n-1; j >= 0; --j) {
            if ((i>>j)%2 == 1) cout << "Y";
            else cout << "N";
        }
        cout << endl;
    }
    return 0;
}

```

## [NOIP 2005 普及组] 采药

这道题同学们用 二进制枚举 拿到 30 分即可

```

#include <bits/stdc++.h>

using namespace std;

const int maxn = 100 + 5;
int t[maxn], val[maxn];

int main()
{
    int T, n; cin >> T >> n;
    for (int i = 1; i <= n; ++i) {
        cin >> t[i] >> val[i];
    }
}

```

```

int res = 0;
for (int i = 0; i < (1<<n); ++i) {
    int sumt = 0, sumv = 0;
    for (int j = 0; j < n; ++j) {
        if ((i>>j) & 1) {
            sumt += t[j+1], sumv += val[j+1];
        }
    }
    if (sumt <= T) res = max(res, sumv);
}
cout << res << endl;
return 0;
}

```

## 众数

记录每个二进制位出现的次数, 看哪些二进制位出现的次数  $\geq$  一半

```

#include <bits/stdc++.h>

using namespace std;

typedef long long LL;
const int n = 999999;
int st[70];

int main()
{
    for (int i = 1; i <= n; ++i) {
        LL x; cin >> x;
        for (int j = 0; j <= 62; ++j) {
            if ((x>>j) % 2 == 1) st[j]++;
        }
    }

    LL res = 0;
    for (int i = 0; i <= 62; ++i) {
        if (st[i] > n/2) res += (1LL<<i);
    }
    cout << res << endl;
    return 0;
}

```

## [蓝桥杯 2025 省 A] 红黑树

搜索, 第  $n$  行第  $k$  列的颜色, 是由第  $n-1$  行第  $(k+1)/2$  列的颜色决定的

```

#include <bits/stdc++.h>

```

```
using namespace std;

int dfs(int n, int k) { // 求第 n 行的第 k 列
    if (n==1 && k==1) return 1;
    if (n == 2) {
        if (k == 1) return 1;
        else return 0;
    }

    if (k%2 == 0) return 1 - dfs(n-1, (k+1)/2);
    else return dfs(n-1, (k+1)/2);
}

void solve() {
    int n, k; cin >> n >> k;
    int value = dfs(n, k);
    if (value == 0) cout << "BLACK" << endl;
    else cout << "RED" << endl;
}

int main()
{
    int T; cin >> T;
    while (T -- ) solve();
    return 0;
}
```