

CSP-J 第二轮国庆考前强化营 · 学生讲义

Day 2 暴力枚举 · 搜索 (DFS / BFS)

上午：全真模拟赛 2 | 下午：模拟赛精讲 + 本专题
极客少儿编程教研组 编 · 2026 年 10 月

一、今日学习目标

- 会用数据范围估算暴力复杂度，判断「枚举能不能过」；
- 熟练掌握 DFS 回溯框架（选 → 递归 → 撤销）与常见剪枝；
- 熟练掌握 BFS 队列框架，求无权最短路、连通块；
- 树立作答顺序意识：先写暴力拿保底分，再考虑优化。

二、考点梳理

2.1 暴力枚举：最容易拿的保底分

枚举不是「笨办法」，而是考场上的保底策略：满分思路想不全时，正确的暴力能拿到部分测试点的分。动手前先估算枚举量，与 1 秒约 10 次基本操作的量级对比：

枚举类型	规模上限 (1 秒内)	典型写法
单重 / 双重循环 $O(n^2)$	$n \leq 5000$ 左右	for 循环
子集枚举 $O(2^n)$	$n \leq 20 \sim 25$	二进制枚举或 DFS
组合枚举 $O(C(n,k))$	$C(n,k) \leq 10^6$ 左右	DFS 回溯
全排列 $O(n!)$	$n \leq 9 \sim 10$	DFS 或 next_permutation

模板：子集枚举与排列枚举

```
// 二进制枚举：枚举 n 个元素的所有子集 (n <= 20)
for (int s = 0; s < (1 << n); s++) {
    for (int i = 0; i < n; i++)
        if (s >> i & 1) { /* 第 i 个元素在子集中 */ }
}
```

```
// 排列枚举：STL 一行流 (要求序列先升序排序)
sort(a + 1, a + n + 1);
do { /* 使用 a[1..n] */ } while (next_permutation(a + 1, a + n + 1));
```

2.2 DFS 回溯：万能的三步框架

模板：DFS 回溯框架

```
bool vis[N]; int pth[N];           // pth[k] 记录第 k 步的选择
void dfs(int k) {                 // 正在做第 k 步决策
    if (k == n + 1) {             // 所有决策做完 → 得到一个完整方案
        输出或统计方案; return;
    }
    for (每个可选选项 i) {
        if (vis[i]) continue;    // 1. 选: 跳过不合法选项
        vis[i] = true; pth[k] = i;
        dfs(k + 1);              // 2. 递归: 做下一步决策
        vis[i] = false;          // 3. 撤销: 恢复现场 (回溯)
    }
}
```

- 剪枝：在递归前提前排除不可能的分支。两类：可行性剪枝（继续走也不可能合法）与最优性剪枝（继续走也不可能更优）。
- 回溯的本质：状态「先改后还」，vis 数组改了一定要改回来，否则后续分支全错。

2.3 BFS：无权最短路的标准武器

边权全为 1（如迷宫每步一格）时，BFS 按层扩展，第一次到达某点时的步数就是最少步数。关键纪律：入队时立即标记访问，否则同一节点会被重复入队，轻则变慢重则死循环。

模板：BFS 框架（网格版）

```
int dx[4] = {-1, 1, 0, 0}, dy[4] = {0, 0, -1, 1}; // 方向数组
int dis[N][N]; // -1 表示未访问
struct Node { int x, y; };
queue<Node> q;
dis[sx][sy] = 0; q.push({sx, sy});
while (!q.empty()) {
    Node u = q.front(); q.pop();
    for (int i = 0; i < 4; i++) {
        int tx = u.x + dx[i], ty = u.y + dy[i];
        if (越界 || 障碍 || dis[tx][ty] != -1) continue;
        dis[tx][ty] = dis[u.x][u.y] + 1; // 入队即标记
        q.push({tx, ty});
    }
}
```

三、例题精讲

【例题1】选数（组合枚举） 洛谷 P1036

题目描述：从 n 个互不相同的整数中选出 k 个，求使选出的数之和为素数的选法有多少种。 $n \leq 20$ 。

输入格式：第一行 n 、 k ；第二行 n 个整数。

输出格式：一个整数：合法选法数。

样例输入	样例输出
4 3 3 7 12 19	1

思路分析：枚举所有 $C(n,k)$ 种选法逐一检验。 $n = 20$ 时 $C(20,10) \approx 18$ 万，完全在暴力范围内。

用 DFS 回溯枚举组合：每步决定「选或不选下一个数」，或直接枚举「第 j 个数选谁」。素数判定用试除法到 $\sqrt{x}$ 。

样例验证：3+7+12=22（否）；3+7+19=29（是）；3+12+19=34（否）；7+12+19=38（否）→答案 1。

参考代码（C++）

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int N = 25;
int T = 1;

int n, k, a[N], cnt;

bool prime(int x) {
    if (x < 2) return false;
    for (int i = 2; i * i <= x; i++)
        if (x % i == 0) return false;
    return true;
}

void dfs(int p, int c, int s) { // 从 a[p] 起考虑，已选 c 个、和为 s
    if (c == k) { cnt += prime(s); return; }
    for (int i = p; i <= n; i++) // 从 p 起选，保证不重复
        dfs(i + 1, c + 1, s + a[i]);
}

void solve() {
    cin >> n >> k;
    for (int i = 1; i <= n; i++) cin >> a[i];
    dfs(1, 0, 0);
    cout << cnt << '\n';
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    while (T--) solve();
    return 0;
}
```

- 易错点：
- 组合枚举的关键：下一层从 $i+1$ 开始，避免同一组合被不同顺序重复统计；
 - 素数判定别忘 $x < 2$ 的特判；试除循环写成 $i*i \leq x$ ；
 - 本题无需回溯修改状态（参数传递天然隔离），体会与「vis 数组」式回溯的区别。

【例题2】全排列问题（回溯框架） 洛谷 P1706

题目描述：输出 $1\sim n$ 的全排列（ $n \leq 9$ ），按字典序从小到大，每行一个排列。

输入格式：一个整数 n 。

输出格式： $n!$ 行，每行 n 个整数（数之间用空格分隔）。

样例输入	样例输出
3	1 2 3 1 3 2 2 1 3 2 3 1 3 1 2 3 2 1

思路分析：套 § 2.2 回溯框架： $pth[k]$ 表示第 k 个位置放谁， vis 标记已用的数；按 $1..n$ 的顺序尝试，天然得到字典序。

也可以一行 `next_permutation` 解决，但考场上手写回溯更通用（可自由加限制条件），两种都要会。

参考代码 (C++)

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int N = 12;
int T = 1;

int n, pth[N];
bool vis[N];

void dfs(int k) {
    if (k == n + 1) {
        for (int i = 1; i <= n; i++)
            cout << pth[i] << " \n"[i == n];
        return;
    }
    for (int i = 1; i <= n; i++) {
        if (vis[i]) continue;
        vis[i] = true; pth[k] = i;
        dfs(k + 1);
        vis[i] = false;        // 撤销, 恢复现场
    }
}

void solve() {
    cin >> n;
    dfs(1);
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    while (T--) solve();
    return 0;
}
```

易错点:

- vis[i] 递归后必须恢复为 false, 漏掉是最常见的回溯错误;
- 按 1..n 顺序枚举候选, 输出天然字典序, 无需再排序;
- n=9 时输出量约 36 万行, 本身已是极限规模, n 再大必须剪枝或换思路。

【例题3】八皇后（回溯 + 剪枝） 洛谷 P1219

题目描述: 在 $n \times n$ 的棋盘上放 n

个皇后, 使任意两个皇后不在同一行、同一列、同一对角线上 ($n \leq 13$)。按字典序输出前 3 个放置方案, 最后一行输出方案总数。

输入格式: 一个整数 n 。

输出格式: 前 3 行每行 n 个整数 (第 i 个数表示第 i 行皇后所在列); 最后一行方案总数。

样例输入	样例输出
6	2 4 6 1 3 5 3 6 2 5 1 4 4 1 5 2 6 3 4

思路分析：每行恰好放一个皇后：dfs(r) 决定第 r 行的皇后放哪一列。

剪枝核心——三个标记数组：col[c] 标记列；d1[r+c] 标记副对角线（行列和相同）；d2[r-c+n] 标记主对角线（行列差相同，+n 防负下标）。

找到方案时只在前 3 个输出明细，但 cnt 要一直统计到搜索结束。

参考代码 (C++)

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int N = 15;
int T = 1;

int n, cnt, pth[N];
bool col[N], d1[2 * N], d2[2 * N]; // 列、副对角线、主对角线

void dfs(int r) {
    if (r == n + 1) {
        cnt++;
        if (cnt <= 3) {
            for (int i = 1; i <= n; i++)
                cout << pth[i] << " \n"[i == n];
        }
        return;
    }
    for (int c = 1; c <= n; c++) {
        if (col[c] || d1[r + c] || d2[r - c + n]) continue; // 剪枝
        col[c] = d1[r + c] = d2[r - c + n] = true;
        pth[r] = c;
        dfs(r + 1);
        col[c] = d1[r + c] = d2[r - c + n] = false; // 回溯
    }
}

void solve() {
    cin >> n;
    dfs(1);
    cout << cnt << '\n';
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    while (T--) solve();
    return 0;
}
```

易错点：

- 主对角线 r-c 范围是 $-(n-1) \dots (n-1)$ ，统一 +n 映射到正下标；
- 三个标记数组要「同改同还」，只还一个是经典翻车点；
- 先想清楚「和相同 / 差相同」对应哪条对角线，可以画 3×3 小棋盘验证。

【例题4】马的遍历 (BFS 求最少步数) 洛谷 P1443

题目描述： $n \times m$ 棋盘 ($n, m \leq 400$)，马走「日」字。从左下角起点 (x, y) 出发，求到达每个格子的最少步数，不可达输出 -1。

输入格式：一行四个整数 n 、 m 、 x 、 y 。

输出格式： n 行 m 列：到每个格子的最少步数，不可达为 -1 （起点为 0 ）。

样例输入	样例输出
3 3 1 1	0 3 2 3 -1 1 2 1 4

思路分析：每步代价相同 $\rightarrow$ BFS。dis 数组初始化 -1 兼作访问标记，起点 $dis=0$ 入队。

马有 8 个方向，方向数组写成两组 $\pm 1/\pm 2$ 的组合；出界即跳过。

按题意要求输出整个矩阵；每格步数就是 dis 值。

参考代码 (C++)

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int N = 405;
int T = 1;

int n, m, dis[N][N];
int dx[8] = {-2, -2, -1, -1, 1, 1, 2, 2};
int dy[8] = {-1, 1, -2, 2, -2, 2, -1, 1};

struct Node {
    int x, y;
};

void solve() {
    int sx, sy;
    cin >> n >> m >> sx >> sy;
    memset(dis, -1, sizeof(dis));
    queue<Node> q;
    dis[sx][sy] = 0;
    q.push({sx, sy});
    while (!q.empty()) {
        Node u = q.front(); q.pop();
        for (int i = 0; i < 8; i++) {
            int tx = u.x + dx[i], ty = u.y + dy[i];
            if (tx < 1 || tx > n || ty < 1 || ty > m) continue;
            if (dis[tx][ty] != -1) continue; // 已访问
            dis[tx][ty] = dis[u.x][u.y] + 1;
            q.push({tx, ty});
        }
    }
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) cout << dis[i][j] << ' ';
        cout << '\n';
    }
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    while (T--) solve();
    return 0;
}
```

易错点：

- 入队时立刻标记 dis，重复入队会让复杂度爆炸；
- 方向数组 8 组偏移别抄错，建议记「 $(\pm 1, \pm 2)$ 与 $(\pm 2, \pm 1)$ 」口诀；
- 起点步数为 0，不可达保持 -1；输出注意对齐（%-5d 场宽写法）。

四、今日易错点总结

- 回溯三件套「选、递归、撤销」缺一不可；vis 改了必须还。
- BFS 入队即标记；dis 数组用 memset(dis, -1, sizeof(dis)) 初始化。
- 枚举前先算规模：2、n!、C(n,k) 的数量级要形成直觉。
- 搜索题样例过了不代表对：自己造小数据手玩验证（如 2×2、n=1 的边界）。
- 实在不会满分：先交暴力保底，再回头优化——部分分也是分。

五、课后练习（今晚完成，标注★为必做）

题号	题名	考点	难度
★ P1162	填涂颜色	Flood Fill（BFS/DFS 连通块）	普及-
P1596	Lake Counting	Flood Fill 数连通块	普及-
P2392	考前临时抱佛脚	子集枚举 / 背包思想	普及-
P1101	单词方阵	DFS 多方向搜索	普及

做题后把题号、错因、正确思路写入错题本，明早抽查。