

全真模拟赛精讲稿 · Day 1

上午全真模拟 · 下午逐题精讲 (14:00–17:00)

配套主题：基础算法 · 二分 · 贪心
极客少儿编程教研组 · 2026 年 10 月

本场概览

题号	题目	文件名	考点	难度	暴力分	满分算法
T1	程老师的报数游戏	count	枚举 + 数位判断	红 ★☆	100	直接枚举
T2	程老师的划船比赛	boat	贪心 (排序辅助)	橙 ★★	100	排序后相邻贪心配对
T3	程老师的搬运车	rover	BFS / 状态设计	黄 ★★★	35	状态 (格子 × 冷却) BFS
T4	程老师的补给站	supply	DP / 二维容量背包	黄 ★★★	25	双背包 $dp[a][b]$

暴力分合计 260 分：T1、T2 暴力即满分，T3 限深搜索 35 分，T4 三去向枚举 25 分。本场的基本盘很厚——哪怕正解一道不会，诚实写好暴力也能拿到 260 分。

T1 程老师的报数游戏 (count.cpp / count.in / count.out)

考点	难度	暴力分	时限 / 内存
枚举 / 数位处理	红题 ★☆	100	1000ms / 512MB

题意复述

全班玩「逢 7 过」：依次报数 $1, 1+1, \dots, r$ ，遇到一个数含有数字 7 或是 7 的倍数就要拍手。两个条件同时满足也只拍一次手。求从 1 报到 r 一共拍手多少次。

一句话模型：统计区间 $[l, r]$ 内满足 " $x \% 7 == 0$ 或 x 的十进制写法含 7" 的整数个数。

输入输出与数据范围

输入一行两个整数 l, r ；输出一行一个整数。数据范围 $1 \leq l \leq r \leq 10^7$ 。

先算复杂度：枚举每个数并逐位检查，单个数最多 8 位，总操作约 8×10^7 次，远在 1 秒时限之内——这题暴力就是正解，不需要任何优化。

样例与解释

样例输入	样例输出
1 20	3
46 50	2

样例 1：1 到 20 中要拍手的是 7（含 7 且是 7 的倍数）、14（7 的倍数）、17（含 7），共 3 次。

样例 3：47 含数字 7，49 是 7 的倍数，46、48、50 都不沾边，共 2 次。

部分分策略（先保底，再冲刺）

枚举 $[l, r]$ 内每个数 x ：用 $x \% 7 == 0$ 判断倍数；写一个 `has7(x)` 函数反复取 $x \% 10$ 、 $x /= 10$ 判断是否含数字 7。两个条件用或连接，满足即计数。实测最慢点 56.8ms，直接拿 100 分。

教学点：拿到题先估算规模—— $r \leq 10^7$ 、每数 8 位， 10^8 次简单运算在 C++ 里约 0.1 秒。确认暴力可过就果断写暴力，把省下的时间留给 T3、T4。考场上"敢写暴力"是一种需要训练的判断力。

满分思路

正解与暴力完全一致：枚举 + 数位判断，时间复杂度 $O((r-l+1) \cdot \log r)$ 。

写 `has7` 时注意循环条件是 `while (x)`，逐位取余；主循环里两个条件只要满足其一就 `ans++`，天然保证同一数不会重复计数。

参考代码（C++，NOI Linux 下 -O2 编译通过）

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

bool has7(int x) {
    while (x) {
        if (x % 10 == 7) return true;
        x /= 10;
    }
}
```

```
        return false;
    }

    void solve() {
        int l, r;
        cin >> l >> r;
        int ans = 0;
        for (int i = l; i <= r; i++)
            if (i % 7 == 0 || has7(i)) ans++;
        cout << ans << '\n';
    }

    signed main() {
        freopen("count.in", "r", stdin);
        freopen("count.out", "w", stdout);
        ios::sync_with_stdio(false);
        cin.tie(0); cout.tie(0);
        while (T--) solve();
        return 0;
    }
```

错解警示（均为实测得分）

典型错解	错在哪里	实测得分
只算 7 的倍数	漏掉 17、70、172 等"含 7 但不是 7 的倍数"的数	20
只算含数字 7	漏掉 14、28、49 等"是 7 的倍数但不含 7"的数	25
两条件分别计数再相加	7、70、707 这类两条件同时满足的数被算了两次，答案偏大	15

变式拓展

- 把规则改成"含数字 d 或是 d 的倍数"（d 由输入给出）：has7 参数化即可，体会"magic number 写成变量"的好习惯。
- 把规则改成"两个条件同时满足才拍手"：或改成且，答案会大幅缩水——估算一下 $[1, 10^7]$ 内还剩多少。
- 思考题（提高组衔接，只点名不展开）：若 $r \leq 10^{18}$ ，枚举不可行，需要用数位 DP 或按位分类计数——同一道题，数据范围决定算法档次。

T2 程老师的划船比赛 (boat.cpp / boat.in / boat.out)

考点	难度	暴力分	时限 / 内存
贪心（排序辅助）	橙题 ★★	100	1000ms / 512MB

题意复述

n 名学生，第 i 名体重 w_i 。每艘船恰好坐 2 人，且同船两人体重差不能超过 d 。每名同学最多上一条船，也可以不上船。求最多能组成多少对（多少艘船同时出发）。

输入输出与数据范围

第一行 n, d ；第二行 n 个体重。 $1 \leq n \leq 10^5$, $w_i \leq 10^9$ 。

排序 $O(n \log n)$ 完全可行——本题的算法骨架就是“排序 + 一遍扫描”。

样例与解释

样例输入	样例输出
4 1 1 2 3 10	1
5 0 5 5 5 3 3	2

样例 1：排序后 $[1, 2, 3, 10]$ 。配 1 和 2（差 $1 \leq 1$ ），剩 3 和 10 配不上，共 1 对；也可以跳过 1 配 2 和 3，同样只有 1 对。

样例 2： $d = 0$ 时只能同体重配对， $[3, 3, 5, 5, 5]$ 配成 2 对。

部分分策略（先保底，再冲刺）

排序后从左往右扫：若相邻两人 $w[i+1] - w[i] \leq d$ 就配对（ i 加 2），否则跳过当前人（ i 加 1）。一遍扫描 $O(n)$ ，实测最慢 11.5ms，直接 100 分。

教学点：本题“暴力即满分”，但前提是贪心策略想对了。贪心题真正的成本不在代码，在正确性论证。

满分思路

排序之后，“相邻配对”的正确性用交换论证说明：

① 若 $w[i]$ 连离它最近的 $w[i+1]$

都配不上，那它与后面任何人差距更大，永远配不上，直接放弃不亏；

② 若 $w[i]$ 与 $w[i+1]$ 配得上，就把它们配成一对——假设某最优方案里 $w[i]$ 配的是后面的 $w[j]$ （ $j > i+1$ ），把 $w[i+1]$ 换进来与 $w[i]$ 配对，体重差只会更小仍然合法，而 $w[j]$ 被释放出来不会减少任何配对机会，总数不会变少。所以“能配就配最近的那个”绝不劣于任何方案。

代码上注意 i 的两种步进：配对成功 $i += 2$ ，失败 $i += 1$ 。

参考代码（C++，NOI Linux 下 -O2 编译通过）

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int N = 1e5 + 10;
int T = 1;
```

```
int n, d, w[N];

bool cmp(int a, int b) { return a < b; }

void solve() {
    cin >> n >> d;
    for (int i = 0; i < n; i++) cin >> w[i];
    sort(w, w + n, cmp);
    int ans = 0;
    for (int i = 0; i + 1 < n; ) {
        if (w[i + 1] - w[i] <= d) {
            ans++;
            i += 2;
        } else {
            i++;
        }
    }
    cout << ans << "\n";
}

signed main() {
    freopen("boat.in", "r", stdin);
    freopen("boat.out", "w", stdout);
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    while (T--) solve();
    return 0;
}
```

错解警示（均为实测得分）

典型错解	错在哪里	实测得分
固定相邻配对 (1,2)(3,4)...	某对配不上时本应只跳过一人，此写法机械跳过两人，错过后续机会	25
首尾双指针配对	排序后拿最轻与最重硬凑，浪费中间"桥梁"人选，策略本身不成立	15

变式拓展

- 对照题 P1094 纪念品分组：每组 2 人、价格之和不超 过 w——"和"型约束的正确贪心是排序后首尾双指针，与本题"差"型约束的相邻扫描恰好互换。放在一起讲：贪心策略的形态由约束形状决定，不能背套路。
- 追问：若每艘船可以坐 3 人且任意两人差都 $\leq d$ ，贪心还成立吗？（不再成立，需要搜索或 DP——体会贪心的适用边界：问题必须有"交换不亏"的结构。）

T3 程老师的搬运车 (rover.cpp / rover.in / rover.out)

考点	难度	暴力分	时限 / 内存
BFS / 状态设计	黄题 ★★★	35	1000ms / 512MB

题意复述

n 行 m 列的仓库，每格印着 >、<、^、v 四种箭头之一。搬运车从 (1,1) 出发到 (n, m)，每步两种驾驶方式：自动导航——沿当前格箭头走一格（箭头指向仓库外则此步自动不可用）；手动驾驶——向上下左右任意相邻格走一步，但一次手动之后接下来 2 步必须自动，第 3 步起恢复手动资格。出发时处于可手动状态。求最少步数，无法送达输出 -1。

读题关键："手动之后恰好 2 步强制自动"——车"现在能不能手动"取决于历史，这正是需要给状态加维的信号。

输入输出与数据范围

第一行 n, m (≤ 1000)，随后 n 行箭头网格。输出最少步数或 -1。
状态数若只按格子算是 10⁶，加一维冷却 0/1/2 后 3×10⁶，BFS 仍可轻松胜任（实测 115.7ms）。

样例与解释

样例输入	样例输出
3 3 >v> ^>< ^<<	4
2 3 >>v ^<<	3

样例 1：第 1 步手动从 (1,1) 向右到 (1,2)——此后 2 步被锁定为自动；第 2 步沿 (1,2) 的 v 自动到 (2,2)；第 3 步沿 (2,2) 的 > 自动到 (2,3)，锁定解除；第 4 步手动向下到 (3,3)。共 4 步。

样例 2：全程自动 (1,1)→(1,2)→(1,3)→(2,3)，3 步直达——自动导航不触发冷却，能自动时未必需要手动。

部分分策略（先保底，再冲刺）

DFS 枚举驾驶序列，深度上限设为 10：欠 0 步时枚举 4 个手动方向 + 1 个自动方向，欠 1~2 步时只能走自动。由于手动后必接 2 步强制自动，搜索树实际只有几千个节点，毫秒级跑完。
实测：测试点 1~7 的答案都不超过 10，全部算对；第 8 点起答案跳到 98
以上，深度内搜不到终点，一律输出 -1（是放弃，不是超时）。合计 35 分。
教学点：搜索题的限深枚举是最常见的部分分手段；事先无法知道答案深度时，"限深 + 搜不到输出 -1"是诚实且安全的写法。

满分思路

把局面抽象成状态：（所在格子，还欠几步强制自动），欠步数只有 0/1/2 三档，状态总数 3nm。每个状态的转移：

① 欠 0 步：可以手动走向 4 个相邻格（新状态欠 2 步）；也可以自动（沿箭头走，新状态仍欠 0 步）；

② 欠 $c > 0$ 步：只能自动，新状态欠 $c-1$ 步；箭头朝外时该状态没有出边。

每步代价都是 1，"最少步数"就是这张状态图上的无权最短路——BFS。终点 (n, m) 的三个冷却档位中取最小距离；三档都不可达输出 -1。

实现技巧：状态编号 = 格子编号 $\times 3 +$

欠步数，用一维数组当队列和距离表，省去结构体。每个状态至多入队一次，总复杂度 $O(nm)$ 。

与 Day 4 的呼应：这就是"状态即点、转移即边"的隐式图——图不一定以 n 个点 m 条边的形式给出，能用 (格子, 冷却) 唯一描述一个局面，它就是图上的一个顶点。

参考代码 (C++, NOI Linux 下 -O2 编译通过)

```
#include <bits/stdc++.h>
using namespace std;
#define int long long

int n, m, nm;
string g;
vector<int> dst, qu;
int dx[4] = {-1, 1, 0, 0};
int dy[4] = {0, 0, -1, 1};
int T = 1;

bool ins(int r, int c) {
    return 0 <= r && r < n && 0 <= c && c < m;
}

// 当前格箭头指向的格子编号，指向网格外返回 -1
int nxt(int r, int c) {
    char ch = g[r * m + c];
    int nr = r, nc = c;
    if (ch == '^') nr--;
    else if (ch == 'v') nr++;
    else if (ch == '<') nc--;
    else nc++;
    if (!ins(nr, nc)) return -1;
    return nr * m + nc;
}

// 状态编号 = 格子编号 * 3 + 还需强制自动的步数(0/1/2)
void solve() {
    cin >> n >> m;
    g.assign(n * m, '?');
    for (int i = 0; i < n; i++) {
        string s;
        cin >> s;
        for (int j = 0; j < m; j++) g[i * m + j] = s[j];
    }
    nm = n * m;
    dst.assign(3 * nm, -1);
    qu.assign(3 * nm, 0);
    int hd = 0, tl = 0;
    dst[0] = 0;
    qu[tl++] = 0;
    while (hd < tl) {
        int u = qu[hd++];
```

```

int cd = u % 3, p = u / 3;
int d = dst[u];
int r = p / m, c = p % m;
if (cd == 0) {
    for (int k = 0; k < 4; k++) {
        int nr = r + dx[k], nc = c + dy[k];
        if (!ins(nr, nc)) continue;
        int v = (nr * m + nc) * 3 + 2;
        if (dst[v] == -1) {
            dst[v] = d + 1;
            qu[tl++] = v;
        }
    }
    int t = nxt(r, c);
    if (t != -1 && dst[t * 3] == -1) {
        dst[t * 3] = d + 1;
        qu[tl++] = t * 3;
    }
} else {
    int t = nxt(r, c);
    if (t != -1 && dst[t * 3 + cd - 1] == -1) {
        dst[t * 3 + cd - 1] = d + 1;
        qu[tl++] = t * 3 + cd - 1;
    }
}
}
int tg = (nm - 1) * 3;
int ans = -1;
for (int k = 0; k < 3; k++) {
    int d = dst[tg + k];
    if (d != -1 && (ans == -1 || d < ans)) ans = d;
}
cout << ans << '\n';
}

signed main() {
    freopen("rover.in", "r", stdin);
    freopen("rover.out", "w", stdout);
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    while (T--) solve();
    return 0;
}

```

错解警示（均为实测得分）

典型错解	错在哪里	实测得分
只按格子判重	状态转移按冷却展开，但判重只记"格子来过没有"——先到者若是欠 2 步进来的，后面"欠 0 步"的回归被整体跳过，而能继续手动的恰恰是欠 0 步	15
手动驾驶无冷却	把冷却一维整个丢掉，每格随时可手动，答案只会偏小不错大——但偏小也是错	20
只会自动导航	从不手动，沿箭头一路走，随机数据下很快进环或被边界卡死	15

变式拓展

- 若冷却从 2 步改成 k 步：状态变成 $(k+1)$ 档，写法完全不变——状态维数跟着规则走。
- 对比 P1443 马的遍历（裸 BFS）：什么时候要加状态维？——当“下一步能怎么动”不仅取决于位置、还取决于历史时，就把这个历史量加进状态。
- 思考题（提高组衔接，只点名）：若手动一步代价 2、自动代价 1，边权不再相等，BFS 要换成 Dijkstra。

T4 程老师的补给站 (supply.cpp / supply.in / supply.out)

考点	难度	暴力分	时限 / 内存
DP / 二维容量背包	黄题 ★★★	25	1000ms / 512MB

题意复述

n 件装备，第 i 件重 w_i 公斤、重要度 v_i ，每件只有一件。程老师与向导各背一个包，容量分别为 W_1 、 W_2 ，两个包互不借用。每件装备三选一：进程老师的包、进向导的包、不带。求两包都不超重时，带走装备的重要度总和最大值。

与经典 0/1

背包的唯一差别：容量约束从一维变成二维——一件装备可以放进两个包中的任意一个。

输入输出与数据范围

第一行 n, W_1, W_2 ；随后 n 行 w_i, v_i 。 $n \leq 100$, $W_1, W_2 \leq 200$, $v_i \leq 10^6$ 。

答案上界 $\sum v_i \leq 100 \times 10^6 = 10^8$ ，必须开 long long（本题代码统一 `#define int long long`）。

样例与解释

样例输入	样例输出
3 5 5 3 10 3 9 3 8	19
3 4 4 3 11 3 9 2 8	20

样例 1：三件都重 3 公斤，每个包只能装一件，最多带走两件，取重要度最大的 10 和 9，合计 19。

样例 2：第一件 (3, 11) 进程老师的包，第三件 (2, 8) 进向导的包，合计 19？不对——最优是第一件与第二件各进一包：11 + 9 = 20。注意 (3, 9) 这件在两个样例里的取舍不同。

部分分策略（先保底，再冲刺）

枚举每件装备的三个去向（程包 / 向导包 / 不带），检查两包容量，取合法方案的最大值。共 3^n 种方案， $n \leq 20$ 时可过。实测通过测试点 1~5 ($n \leq 20$)，第 6 点起 $n = 100$ ， 3^{100} 必然超时，合计 25 分。

教学点：背包问题的暴力形态就是“每件物品的若干去向全枚举”，指数级但写得快，先把 25 分装进口袋，再回头推 DP。

满分思路

把 0/1 背包的容量维直接复制一份：设 $dp[a][b]$ = 已处理的装备中，程老师的包总重不超过 a 、向导的包总重不超过 b 时的最大重要度。对每件装备 (w, v) 做三种转移：

- ① 进程老师的包： $dp[a][b] = \max(dp[a][b], dp[a-w][b] + v)$ （要求 $a \geq w$ ）；
- ② 进向导的包： $dp[a][b] = \max(dp[a][b], dp[a][b-w] + v)$ （要求 $b \geq w$ ）；

③ 不带: $dp[a][b]$ 保持原值——不用写代码, 数组不动即是"不带"。

a、b 都必须倒序枚举, 保证 $dp[a-w][b]$ 与 $dp[a][b-w]$

仍是上一件装备的结果, 每件装备至多使用一次。总复杂度 $O(n \cdot W_1 \cdot W_2) = 100 \times 200 \times 200 = 4 \times 10^6$, 实测最慢 7.6ms。

与 Day 3 的衔接: 背包问题的状态维数 =

容量约束的个数。下午讲背包时, 从"一维容量"到本题"二维容量"再到"件数上限" (Day 4 T3), 是同一条增维主线。

参考代码 (C++, NOI Linux 下 -O2 编译通过)

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int W = 205;

int n, w1, w2;
// dp[a][b]: 已处理的装备中, 程老师的包总重不超过 a、向导的包总重不超过 b 时的最大重要度
int dp[W][W];

void solve() {
    cin >> n >> w1 >> w2;
    memset(dp, 0, sizeof(dp));
    for (int i = 1; i <= n; i++) {
        int w, v;
        cin >> w >> v;
        // a、b 均倒序, 保证 dp[a-w][b] 与 dp[a][b-w] 仍是上一件装备的结果
        for (int a = w1; a >= 0; a--)
            for (int b = w2; b >= 0; b--) {
                if (a >= w) dp[a][b] = max(dp[a][b], dp[a - w][b] + v); // 放进程老师的包
                if (b >= w) dp[a][b] = max(dp[a][b], dp[a][b - w] + v); // 放进向导的包
            }
        // 不带: dp[a][b] 保持原值
    }
    cout << dp[w1][w2] << "\n";
}

signed main() {
    freopen("supply.in", "r", stdin);
    freopen("supply.out", "w", stdout);
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    solve();
    return 0;
}
```

错解警示 (均为实测得分)

典型错解	错在哪里	实测得分
两包合并成容量 W_1+W_2 的单背包	"总容量够"不代表"分得进两个包"——大件可能谁也单独装不下; 仅在 $W_2=0$ 或全部 $w_i=1$ 的性质点上与正解结构性等价	30
按性价比 v/w 降序贪心	0/1 背包不具备贪心选择性质, 性价比高不等于该选; 仅在全部 $w_i=1$ 的性质点上碰巧正确	20

变式拓展

- 若变成三个背包: $dp[a][b][c]$ 三维状态, 容量乘积若在 10^7 内仍可做——体会"维数 = 约束个数"以及维数灾难。
- 对照题 P1855 (二维费用背包: 费用同时消耗金钱与时间): 与本题"双容量"同构, 都是 0/1 背包加一维约束。
- 追问: 转移里为什么没有"不带"这一行代码? —— $dp[a][b]$ 保持原值就是"不带", 背包 DP 里"不选"永远是默认分支。

本场得分策略小结

- 基本盘 260 分：T1 (100) + T2 (100) 暴力即满分，T3 限深搜索 (35) + T4 三去向枚举 (25)。哪怕正解全部不会，诚实写好四个暴力也远超预期线。
- 考场时间建议：T1 控制在 15 分钟内（暴力 + 样例验证）；T2 用 20 分钟写出排序贪心；T3 先写限深 DFS 保底 35，再攻状态 BFS；T4 先写 3^n 枚举保底 25，再推双背包 DP。
- 检查清单：四个文件名 count/boat/rover/supply 与 freopen 拼写；T4 的 long long；T3 的 -1 输出（不可达不是 0）；T2 排序后再扫描。
- 与下午专题的衔接：T2 贪心的交换论证 → 下午「贪心证明套路」；T4 背包增维 → Day 3 动态规划主线预告；T3 状态即点 → Day 4 图论「隐式图」预告。

本场题眼：T1 读范围定算法；T2 约束形状决定贪心形态；T3 历史影响决策就要加状态维；T4 约束有几个容量，DP 就开几维。