

全真模拟赛精讲稿 · Day 2

上午全真模拟 · 下午逐题精讲 (14:00–17:00)

配套主题：暴力枚举 · 搜索

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	程老师的负二进制	base	模拟 / 进制转换	红 ★☆	100	修正规则逐位转换
T2	程老师的消消乐	cancel	模拟 / 计数栈	橙 ★★	30	计数栈 $O(n)$
T3	程老师的浇水系统	water	差分 (二阶)	黄 ★★★	25	二阶差分 + 前缀和
T4	程老师的奖品座位	seat	线性 DP	黄 ★★★	30	dp + 前缀最大值

暴力分合计 185 分：T1 暴力即满分，T2 反复扫描消除 30 分，T3 逐坛累加 25 分，T4 $O(n^2)$ DP 30 分。本场的区分度不在算法储备，而在规则理解——T1 的修正规则、T2 的连锁消除、T3 的等差结构、T4 的间隔 d ，四道题的典型错解没有一种是“不会算法”，全都是“按模板、按直觉写”。读懂题，比背模板值钱。

T1 程老师的负二进制 (base.cpp / base.in / base.out)

考点	难度	暴力分	时限 / 内存
模拟 / 进制转换	红 ★ ☆	100	1000ms / 512MB

题意复述

把十进制整数 n （正、负、零都可能）转成负二进制：基数不是 2，而是 -2 。转换方法与普通进制一样——反复除以 -2 、记录余数、逆序拼接；但有一条修正规则：余数必须非负，若某步余数为负，就把商加 1、余数加 2，再继续。
 $n = 0$ 时直接输出 0。

输入输出与数据范围

输入一行一个整数 n ， $|n| \leq 10^9$ ；输出一行一个 01 串（负二进制表示，无前导零）。
先算复杂度： $|n| \leq 10^9$ 时负二进制最多约 30 位，逐位转换不过 30 次循环——这题暴力就是正解，区分度全在修正规则写没写对。

样例与解释

样例输入	样例输出
1	1
-3	1101
7	11011

样例 2 验算 $n = -3$ ：第一步 $-3 \div (-2)$ 商 1 余 -1 ，余数为负触发修正——商变 2、余数变 1；第二步 $2 \div (-2)$ 商 -1 余 0；第三步 $-1 \div (-2)$ 商 0 余 -1 ，再修正——商变 1、余数变 1；第四步 $1 \div (-2)$ 商 0 余 1，结束。余数逆序拼出 1101。
样例 3 验算 $n = 7$ ：7 → 商 -3 余 1； $-3 \rightarrow$ 修正后商 2 余 1；2 → 商 -1 余 0； $-1 \rightarrow$ 修正后商 1 余 1；1 → 商 0 余 1。逆序得 11011。

部分分策略（先保底，再冲刺）

按题面规则逐位转换：循环 while ($n \neq 0$)，每步 $r = n \% (-2)$ 、 $n /= (-2)$ ，若 $r < 0$ 则 $r += 2$ 、 $n += 1$ ，把字符 $0+r$ 追加到答案串，最后逆序输出。实测全部 20 个点通过，直接拿 100 分。
教学点：进制转换的框架（除基取余、逆序拼接）从不改变，变的只是基数和修正规则。考场上遇到“xx 进制变体”，先把普通框架写出来，再把题面的特殊规则嵌进循环体。

满分思路

正解与暴力完全一致，时间复杂度 $O(\log |n|)$ 。两个细节决定生死：一是 C++ 的 $\%$ 向零取整，负数取余可能得到负余数——这正是需要修正的情形，修正（商 +1、余数 +2）本质是把商向上凑整一格、让余数落回 0 或 1；二是 $n == 0$ 必须特判输出 0，否则循环不执行、输出空串。
参考代码（C++，NOI Linux 下 -O2 编译通过）

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

```
int main() {
    freopen("base.in", "r", stdin);
    freopen("base.out", "w", stdout);
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);

    int n;
    cin >> n;

    if (n == 0) {
        cout << "0\n";
        return 0;
    }

    string ans;
    while (n != 0) {
        int r = n % (-2);
        n /= (-2);
        if (r < 0) {
            r += 2;
            n += 1;
        }
        ans.push_back('0' + r);
    }

    reverse(ans.begin(), ans.end());
    cout << ans << "\n";

    return 0;
}
```

错解警示（均为实测得分）

典型错解	错在哪里	实测得分
按普通二进制转换	基数当成 2，负数取绝对值转换再补符号；两种表示只有 0 和 1 相同，其余全错	10
修正方向修反	负余数时商减 1、余数减 2（方向反了），越修越负，部分输入直接死循环	10

变式拓展

- 改成负三进制：余数为负时商 +1、余数 +3。修正量跟着基数走，框架一行不动。
- 为什么修正后的循环一定终止？——每两步 $|n|$ 至少减半。能讲清这一点，就真正理解了进制转换的正确性，而不只是背了模板。
- 思考题（提高组衔接，只点名不展开）：在负二进制下直接做加减法（不转回十进制），需要按位模拟一套新的进位规则。

T2 程老师的消消乐 (cancel.cpp / cancel.in / cancel.out)

考点	难度	暴力分	时限 / 内存
模拟 / 计数栈	橙 ★★	30	1000ms / 512MB

题意复述

黑板上一串小写字母。每次操作可以擦掉一段连续相同、恰好 k 个的字母；擦掉后左右拼接，拼接处若又形成连续 k 个相同字母，还要继续擦（连锁消除），直到不存在可擦的段。

求最终剩下的字符串；若全部擦光，输出 EMPTY。

输入输出与数据范围

第一行字符串 s ($|s| \leq 10^6$)，第二行整数 k ($2 \leq k \leq |s|$)。输出最终串或 EMPTY。

$|s|$ 最大 10^6 ，反复扫描消除的 $O(E \cdot n)$ 做法只能过 $|s| \leq 2000$ 的小档（30 分），满分需要 $O(n)$ 的一次扫描。

样例与解释

样例输入	样例输出
abba 2	EMPTY
aaabaaaa 3	ba
abccba 3	abccba

样例 1: abba 先擦中间两个 b，剩下的两个 a 拼到一起又凑够 $k = 2$ ，继续擦——黑板为空，输出 EMPTY。

样例 2: aaabaaaa 开头的 aaa 被擦，剩 baaaa；末尾 aaaa 每次擦 3 个，擦一次剩 1 个 a，最终剩 ba。

样例 3: abccba 中 c 只有连续 2 个，不够 $k = 3$ ，没有任何可擦的段，原样输出。

部分分策略（先保底，再冲刺）

暴力：反复扫描整串，每轮找到第一段连续相同字母数 $\geq k$ 的位置擦掉恰好 k 个，重新扫描，直到没有可擦段。每轮 $O(n)$ ，轮数等于消除次数 E ，总 $O(E \cdot n)$ 。实测过测试点 1-6（样例 + $|s| \leq 2000$ ），保底 30 分，第 7 点起超时。

教学点：string 的 erase 是 $O(n)$ ，嵌在循环里就是平方级。估算一下： $|s| = 10^6$ 、 $k = 2$ 时 E 可达 5×10^5 ， $E \cdot n$ 超 10^{11} ——写完暴力就该知道这个量级过不了，立刻想正解。

满分思路

正解是计数栈：栈里每个元素记（字符，连续出现次数）。从左到右扫描原串——当前字符与栈顶字符相同就把栈顶计数加一，否则压入新元素（字符，1）；每步检查栈顶计数，恰好等于 k 就把栈顶整段弹出。

连锁消除被这个结构天然解决：弹出栈顶后，新栈顶恰好是"拼接处左边"的连续段，它的计数继续与后续字符累加，无需任何额外判断。时间复杂度 $O(n)$ ，每个字符只进栈一次、出栈一次。

参考代码（C++，NOI Linux 下 -O2 编译通过）

```
// 正解：（字符，连续计数）栈， $O(n)$ 
// 栈顶与新字符合并计数，计数恰好到 k 即弹出
#include <bits/stdc++.h>
using namespace std;

const int N = 1e6 + 10;

struct Node {
    char c;
    int cnt;
};

Node st[N];
int tp;
string s;
int n, k;

int main() {
    freopen("cancel.in", "r", stdin);
    freopen("cancel.out", "w", stdout);
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    cin >> s >> k;
    n = s.size();
    for (int i = 0; i < n; i++) {
        if (tp > 0 && st[tp - 1].c == s[i]) {
            st[tp - 1].cnt++;
        } else {
            st[tp].c = s[i];
            st[tp].cnt = 1;
            tp++;
        }
        if (st[tp - 1].cnt == k) tp--;
    }
    if (tp == 0) {
        cout << "EMPTY\n";
    } else {
        string ans;
        for (int i = 0; i < tp; i++) ans.append(st[i].cnt, st[i].c);
        cout << ans << "\n";
    }
    return 0;
}
```

错解警示（均为实测得分）

典型错解	错在哪里	实测得分
只消一轮不处理连锁	游程压缩后每段留 $L \bmod k$ 个直接输出；拼接处新形成的连续段全部漏消	25
连续 $\geq k$ 个整段消	aaaa 在 $k = 3$ 时被整段消光——规则是"恰好 k 个"，应消 3 个剩 1 个	25

变式拓展

- 若规则改成"至少 k 个就能擦", 最终串还唯一吗? (不再唯一, 结果依赖消除顺序——体会本题"顺序无关"这一性质的巧妙。)
- 用 `string` 当栈 (尾部 `push_back` / `pop_back`) 重写, 代码量减半, 本质完全一样。
- 与 Day 1 T2 对照: 两道橙题都不是算法题, 区分度全在规则细节——"恰好 k 个"和"排序后相邻配对", 读错题就是 25 分与 100 分的差距。

T3 程老师的浇水系统 (water.cpp / water.in / water.out)

考点	难度	暴力分	时限 / 内存
差分（二阶）	黄 ★★★	25	1000ms / 512MB

题意复述

n 个花坛排成一排，初始水量都是 0。 q 次作业，每次给四个整数 l, r, a, d ：第 l 坛浇 a 升，之后每前进一坛多浇 d 升（等差递增），第 r 坛浇 $a + (r-l) \cdot d$ 升后收工。 $d = 0$ 时全程每坛都浇 a 升。作业范围可以任意重叠，水量直接累加。

q 次作业全部完成后，回答 m 次抽查：每次给一个编号 p ，报告第 p 坛的总水量。

输入输出与数据范围

第一行 n, q, m （均 $\leq 10^5$ ）；接下来 q 行作业、 m 行抽查。输出 m 行答案。

数值警示：满档数据单坛水量约 $q \cdot (a + n \cdot d) \approx 10^{13}$ ，远超 `int`——必须 `long long`，题面的数据范围提示是认真的。

样例与解释

样例输入	样例输出
5 2 5	2
1 4 2 3	5
3 5 1 2	9
1	14
2	5
3	
4	
5	
6 3 4	6
2 5 4 0	1
1 3 1 1	6
4 6 2 2	8
6	
1	
4	
5	

样例 1：作业 1 给 1~4 坛浇 2, 5, 8, 11；作业 2 给 3~5 坛浇 1, 3, 5。累加后五坛依次是 2, 5, 9, 14, 5，五次抽查恰好问遍全部花坛。

样例 2 的作业 1 中 $d = 0$ ：2~5 坛每坛都浇 4 升——等差退化成普通区间加。

部分分策略（先保底，再冲刺）

暴力：每次作业遍历 $[l, r]$ 逐坛累加 $a + (j-l) \cdot d$ ，总复杂度 $O(q \cdot n)$ 。 $n, q \leq 2000$ 时约 4×10^6 次，轻松通过；实测过测试点 1~5，保底 25 分，第 6 点起（ $n, q \geq 8 \times 10^4$ ）超时。

教学点：这 25 分是“诚实分”——按题意直译就能拿到。但 n, q 到 10^5 时 $O(qn)$ 约 10^{10} 次，必须换思路：多次区间修改 +

多次单点查询，是一阶差分的标准信号；只是本题的区间加不是常数，而是等差。

满分思路

关键一步是代数改写：花坛 j 的浇水量 $a + (j-1) \cdot d = (a - l \cdot d) + j \cdot d$ 。对一次作业而言， $(a - l \cdot d)$ 是与位置无关的常数项， d 是一次项系数——两部分各自都是“区间加常数”，可以分别差分。

维护两个差分数组：dc 管常数项 ($dc[l] += a - l \cdot d$, $dc[r+1] -= a - l \cdot d$)，ds 管公差 ($ds[l] += d$, $ds[r+1] -= d$)。全部作业做完后各做一次前缀和还原，第 p 坛的答案就是 $dc[p] + p \cdot ds[p]$ ，每次询问 $O(1)$ 。总复杂度 $O(n + q + m)$ 。

两个细节：修正点在 $r+1$ 而不是 r （第 r 坛仍要浇）；常数项与公差必须同时在 $r+1$ 处收掉——只收常数项会让 r 之后每坛残留一份随位置线性增长的水量。

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 dc[N], ds[N]; // dc: 常数项差分, ds: 斜率项差分

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

    int n, q, m;
    cin >> n >> q >> m;
    for (int i = 0; i < q; i++) {
        int l, r, a, d;
        cin >> l >> r >> a >> d;
        // 花坛 j (l<=j<=r) 浇 a+(j-l)*d = (a-l*d) + j*d
        dc[l] += a - l * d;
        dc[r + 1] -= a - l * d;
        ds[l] += d;
        ds[r + 1] -= d;
    }
    for (int j = 1; j <= n; j++) {
        dc[j] += dc[j - 1];
        ds[j] += ds[j - 1];
    }
    for (int i = 0; i < m; i++) {
        int p;
        cin >> p;
        cout << dc[p] + p * ds[p] << "\n";
    }
    return 0;
}
```

错解警示（均为实测得分）

典型错解	错在哪里	实测得分
等差当普通区间加	每坛只加 a 、无视递增量 d ；只有 d 全为 0 的性质 A 点碰巧正确	15
等差量均摊成常数	把 $a + (r-1) \cdot d/2$ 均摊到每坛；与逐点真实值都不同， d 越大偏得越远	15

变式拓展

- 抽查改成"问区间 $[x, y]$ 内所有花坛的总水量": 在还原出的水量数组上再套一层前缀和即可, 想清"差分还原"与"前缀和查询"的顺序。
- d 可以是负数 (等差递减) 时写法要不要改? ——完全不用, 代数改写对负 d 同样成立。
- 与 Day 1 T1 对照: 两题都是"规则照写", 但本题多了一步把规则拆成可差分形状的代数功夫——这是从模拟走向数据结构预处理的门槛。

T4 程老师的奖品座位 (seat.cpp / seat.in / seat.out)

考点	难度	暴力分	时限 / 内存
动态规划 / 线性 DP	黄 ★★★	30	1000ms / 512MB

题意复述

一排 n 个座位，第 i 个座位上放着价值 a_i 的奖品。可以挑选若干座位领奖，但有一条规矩：任意两个被选座位的编号之差必须大于 d （即至少隔开 d 个座位）。求能领到的最大总价值。

输入输出与数据范围

第一行 n, d ($1 \leq d \leq n \leq 10^5$)，第二行 n 个价值 a_i ($\leq 10^4$)。输出一行最大总价值。
特殊性质 A: $d = 1$ ，退化为"相邻不能同时选"的经典模型；性质 B: 所有价值相同。

样例与解释

样例输入	样例输出
5 1 2 7 3 5 4	12
5 2 2 7 3 5 4	11
4 3 1 2 3 4	4

样例 1: $d = 1$ ，选 2 号 (7) 和 4 号 (5)，编号差 $2 > 1$ ，总价值 12。

样例 2: $d = 2$ ，同样这排座位，2 号与 4 号编号差 2 不满足 > 2 ，改选 2 号 (7) 和 5 号 (4)，总价值 11—— d 变了，最优解跟着变。

样例 3: $d = 3$ ， $n = 4$ 时任意两座编号差都不超过 3，只能选一个，选最大的 4 号。

部分分策略（先保底，再冲刺）

$O(n^2)$ DP: $dp[i]$ 表示只考虑前 i 个座位的最大价值，转移 $dp[i] = \max(dp[i-1], \max\{dp[j] + a[i] \mid j \leq i-d-1\})$ ——不选 i ，或选 i 并枚举上一个被选座位 j 。 $n \leq 2000$ 可过，实测测试点 1-6 通过，保底 30 分，第 7 点起超时。

教学点：这个暴力本身就是正确的 DP，超时的原因只是"枚举前驱 j "这一步太蠢——所有 $j \leq i-d-1$ 里我们只需要最大的 $dp[j]$ 。优化目标非常明确。

满分思路

前缀最大值优化：再开数组 $pm[i] = \max(pm[i-1], dp[i])$ ，即 dp 的前缀最值。转移时"选 i "的最优前驱直接查 $pm[i-d-1]$ ， $O(1)$ 完成： $dp[i] = \max(dp[i-1], a[i] + pm[i-d-1])$ ($i-d-1 < 1$ 时视作 0)。总复杂度 $O(n)$ 。

$d = 1$ 时 $pm[i-2]$ 就是 $dp[i-2]$ ，式子退化为打家劫舍 $dp[i] = \max(dp[i-1], dp[i-2] + a[i])$ ——经典模型只是本题 $d = 1$

的特例。先把通用式推对，特例自然成立；反过来先背特例模板，换个 d 就全军覆没。

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;
int a[N], dp[N], pm[N];

void solve() {
    cin >> n >> d;
    for (int i = 1; i <= n; i++) cin >> a[i];
    dp[0] = 0;
    pm[0] = 0;
    for (int i = 1; i <= n; i++) {
        // 不选 i
        int v1 = dp[i - 1];
        // 选 i, 只能从 j <= i-d-1 转移
        int j = i - d - 1;
        int v2 = a[i] + (j >= 1 ? pm[j] : 0);
        dp[i] = max(v1, v2);
        pm[i] = max(pm[i - 1], dp[i]);
    }
    cout << dp[n] << "\n";
}

signed main() {
    freopen("seat.in", "r", stdin);
    freopen("seat.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
无视 d 硬套 d=1 模板	始终用 $dp[i] = \max(dp[i-1], dp[i-2] + a[i])$; $d = 1$ 的点碰巧正确, $d \geq 2$ 全错	15
按价值降序贪心	从大到小尝试选取、冲突跳过; 否决一个高价值座位可能浪费两侧整片可选空间, 无法保证全局最优	10

变式拓展

- 要求输出选了哪些座位: 从 $dp[n]$ 倒推——若 $dp[i] == dp[i-1]$ 则 i 没选, 否则选 i 并跳到 $i-d-1$ 之前继续。DP 记录方案的标准套路。
- 若价值可以为负数, "一个都不选"是否合法? (本题价值全正, 最优至少选一个; 允许负值时答案可能是 0, 转移初始化随之改变。)
- 与 Day 3 专题的衔接: 本题就是线性 DP 的第一课——状态、转移、前缀最值优化, 三样东西明天会反复出现。

本场得分策略小结

- 基本盘 185 分：T1 (100) 暴力即满分，T2 反复扫描 (30) + T3 逐坛累加 (25) + T4 $O(n^2)$ DP (30)。185 只是保底——本场暴力与正解差距很大，正解全对才进第一梯队。
 - 考场时间建议：T1 控制在 15 分钟内 (进制框架 + 修正规则，写完拿 1101 验算 -3)；T2 用 25 分钟写计数栈，先想清楚"恰好 k 个"再动手；T3 先写 $O(qn)$ 保底 25，再推二阶差分；T4 先写 $O(n^2)$ 保底 30，再加前缀最大值。
 - 检查清单：四个文件名 base/cancel/water/seat 与 freopen 拼写；T3 的 long long (单坛水量约 10^{13})；T2 的 EMPTY 输出 (全消光不是输出空串)；T1 的 $n = 0$ 特判。
 - 与下午专题的衔接：T4 从 $O(n^2)$ 到 $O(n)$
- 正是"枚举转移来源"被前缀最值替代的过程——枚举是起点不是终点；T2、T3 的暴力都是按题意直译，先写对、再写快，这就是下午「暴力枚举」专题要练的作答顺序。

本场题眼：T1 进制框架不变，修正规则跟着题目走；T2 连锁消除用计数栈，"恰好 k 个"不是"至少 k 个"；T3 等差量拆成常数项加一次项，各差分、各还原；T4 枚举前驱不如维护前缀最大值。