

全真模拟赛精讲稿 · Day 4

上午全真模拟 · 下午逐题精讲 (14:00–17:00)

配套主题：图论 (J 组大纲：图的存储 + DFS/BFS 遍历应用)

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	程老师的借阅登记机	borrow	模拟 / 过程处理	红 ★☆	100	状态数组直接模拟
T2	程老师的工牌校验	badge	字符串 / 校验规则	红 ★☆	100	逐位加权和 $O(q)$
T3	程老师的周转交接	stock	前缀和 + 后缀最值	黄 ★★★	30	后缀最小值 $O(n)$
T4	程老师的跳格训练	jump	DP / 状态设计	绿 ★★★	10	$dp[i][l]$ + 滚动数组

暴力分合计 240 分：T1、T2 两道签到题暴力即满分，T3 按定义 $O(n^2)$ 模拟 30 分，T4 指数 DFS 10 分。本场前两题考“规则逐条落实”的基本功，后两题考“倒着想”与“状态设计”——区分度全在思维，不在模板。

T1 程老师的借阅登记机 (borrow.cpp / borrow.in / borrow.out)

考点	难度	暴力分	时限 / 内存
模拟 / 过程处理	红 ★☆	100	1000ms / 512MB

题意复述

图书角的登记机依次处理 n 条记录：`B x` 借书、`R x` 还书。书 x 已在借出状态时再借被拒绝；不在借出状态时还书被忽略。处理完全部记录，输出成功执行的条数，以及最终仍借出在外的书号（升序，没有则为空行）。

输入输出与数据范围

$n \leq 10^5$ ，书号 $x \leq 10^4$ 。书号范围只有一万——一个布尔数组就能装下全部状态，这题暴力就是正解，区分度全在细节。

样例与解释

样例输入	样例输出
6 B 1001 R 1001 B 1001 B 1001 R 1002 B 1003	4 1001 1003
3 B 7 B 7 R 8	1 7

样例 1：B 1001 成功；R 1001 成功；B 1001 成功；B 1001 被拒绝（1001 已在借出）；R 1002 被忽略（1002 不在借出）；B 1003 成功。成功 4 条，最后 1001、1003 未还，升序输出。

部分分策略（先保底，再冲刺）

按题意逐条模拟：o[x] 记录书 x 是否借出，B 时未借出才成功并置 1，R 时已借出才成功并置 0。O(n) 一遍扫完，实测 20 个点全过，直接拿 100 分。

教学点：签到题丢分从不在算法，而在细节：第二行要求"没有书借出时输出空行"——不管有没有书，这个换行都必须输出；书号按从小到大输出——从 1 扫到 10^4 天然有序，不用再排序。

满分思路

正解与暴力完全一致，时间复杂度 O(n)。两个细节决定生死：一是成功次数只数"真正改变状态"的记录；二是第二行的空行——很多人前面的分都拿到了，就丢在这一行。

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 = 1e4 + 10;

int n, cnt;
int o[N];
```

```

void solve() {
    cin >> n;
    for (int i = 1; i <= n; i++) {
        string op;
        int x;
        cin >> op >> x;
        if (op == "B") {
            if (!o[x]) {
                o[x] = 1;
                cnt++;
            }
        } else if (o[x]) {
            o[x] = 0;
            cnt++;
        }
    }
    cout << cnt << "\n";
    int f = 1;
    for (int x = 1; x <= 10000; x++) {
        if (o[x]) {
            if (!f) cout << " ";
            f = 0;
            cout << x;
        }
    }
    cout << "\n";
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    freopen("borrow.in", "r", stdin);
    freopen("borrow.out", "w", stdout);
    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
拒绝 / 忽略的记录也计成功	没查状态直接计数；只有全程无拒绝的点（性质 A 与 3 个“借一批还一批”点）碰巧正确	20
第二行按借出先后输出	不排序、按记录顺序输出；只有空清单与碰巧有序的 3 个点得分	15

变式拓展

- 若书号范围到 10^9 ，布尔数组开不下——用 set 维护借出集合，或排序去重；想清楚每种结构的代价。
- 若要求第二行按“借出时间先后”输出呢？——扫描数组做不到，得在每次成功借出时把书号追加进 vector。
- 与 Day 3 T1 加密器对照：两道签到题都是“规则给你、逐条落实”，先把样例手推一遍再动手，比任何模板都稳。

T2 程老师的工牌校验 (badge.cpp / badge.in / badge.out)

考点	难度	暴力分	时限 / 内存
字符串 / 校验规则	红 ★ ☆	100	1000ms / 512MB

题意复述

工牌号 = 6 位数字本体 + 1 位校验符。校验符算法： $v = (2d_1 + 3d_2 + 5d_3 + 7d_4 + 11d_5 + 13d_6) \bmod 13$ ； v 为 0~9 时写作数字，10、11、12 分别写作 A、B、C。本体可能含前导零——前导零照样参与计算、照样原样输出。给 q 个本体，逐个输出完整工牌号。

输入输出与数据范围

$q \leq 10^5$ ，本体是长度恰好为 6 的数字串。先想一个反例：本体 000010 用整数读入会变成 10——必须按字符串读入。

样例与解释

样例输入	样例输出
3	0012340
001234	000010B
000010	1357912
135791	
2	9999995
999999	1000002
100000	

样例 1：000010 的加权和 = $11 \times 1 = 11$ ， $\bmod 13 = 11 \rightarrow B$ ，工牌号 000010B——前导零一个都不能丢；135791 的加权和 = $2 + 9 + 25 + 49 + 99 + 13 = 197$ ， $\bmod 13 = 2$ 。
样例 2：999999 的加权和 = $9 \times (2 + 3 + 5 + 7 + 11 + 13) = 369$ ， $\bmod 13 = 5$ ；100000 只有首位非零，加权和 = 2，校验符 2。

部分分策略（先保底，再冲刺）

逐位展开求和即可：string 读入本体， $v = \sum w_i \times (s[i] - '0')$ （ w 依次是 2, 3, 5, 7, 11, 13）， $v \% 13$ ，按规则映射字符，拼接输出。 $O(q)$ ，实测 20 个点全过，直接拿 100 分。

教学点：凡是"编号 / 号码 / 串"类输入，先问三个问题：会不会有前导零？长度固定吗？有没有字母？三个里有一个"是"，就用 string 读，别用 int。

满分思路

正解与暴力一致。映射细节： $v < 10$ 时输出字符 '0' + v ，否则 'A' + ($v - 10$)—— $10 \rightarrow A$ 、 $11 \rightarrow B$ 、 $12 \rightarrow C$ ，正好连续，一个表达式写完。

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 q;
string s;

char chk(string t) {
    int w[6] = {2, 3, 5, 7, 11, 13};
    int v = 0;
    for (int i = 0; i < 6; i++) v += w[i] * (t[i] - '0');
    v %= 13;
    if (v < 10) return char('0' + v);
    return char('A' + v - 10);
}

void solve() {
    cin >> q;
    for (int i = 1; i <= q; i++) {
        cin >> s;
        cout << s << chk(s) << "\n";
    }
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    freopen("badge.in", "r", stdin);
    freopen("badge.out", "w", stdout);
    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
用整数读入本体	前导零丢失：权重位置全部错位、输出还少位数；数据保证每个无性质点都有前导零本体，逐点必 WA	0
权重顺序记反	按 13, 11, ..., 2 从低位到高位加权；与正确规则完全错开，全场无一点碰巧	0

变式拓展

- 洛谷 P1055（ISBN 号码）：权重 1~9、mod 11、10 记 X——本题的"同族练习"，今晚就刷。
- 为什么选质数做权重？——让相邻两位的差异充分散开，任意一位抄错都几乎必然改变校验符。
- 若校验位改成 2 位（mod 97）呢？想想输出格式和映射规则要动哪里。

T3 程老师的周转交接 (stock.cpp / stock.in / stock.out)

考点	难度	暴力分	时限 / 内存
前缀和 + 后缀最值	黄 ★★★	30	1000ms / 512MB

题意复述

n 个班组按固定顺序与仓库交接： $a_i > 0$ 存入、 $a_i < 0$

取走。硬规矩：任意时刻库存不能为负。对每个起点 i （前面 $i-1$ 个班组跳过），分别求“从第 i 个班组开始交接”所需的最少预存量。

输入输出与数据范围

$n \leq 10^5$, $|a_i| \leq 10^6$ ，一行输出 n 个答案。数值警示：前缀和最大可达 10^{11} ，int 必爆，必须 long long。

样例与解释

样例输入	样例输出
4 3 -5 2 -4	4 7 2 4
3 2 3 4	0 0 0

样例 1：从班组 1 开始，库存依次 3、-2、0、-4，最低 -4，预存 4；从班组 2 开始，库存 -5、-3、-7，最低 -7，预存 7；从班组 3 开始，库存 2、-2，预存 2；从班组 4 开始，库存 -4，预存 4。

样例 2：全在存入，库存只增不减，任何起点都不用预存——答案全 0 不是“没输出”，是 n 个 0。

部分分策略（先保底，再冲刺）

$O(n^2)$ 按定义模拟：外层枚举起点 i ，内层从 i 累加到末尾、跟踪最低库存 low，答案就是 $\max(0, -\text{low})$ 。 $n \leq 2500$ 时约 3×10^6 次累加，实测过测试点 1-6，保底 30 分；第 7 点（ $n = 9 \times 10^4$ ）起累加次数超 8×10^9 ，干净 TLE。

教学点：写完暴力别急着优化，先回答一个问题：对每个起点，我都在重算什么？——从 i 开始的库存曲线，形状完全一样，只是整体平移。能不能只算一次、处处复用？

满分思路

正着问、倒着想。设前缀和 P ($P_0 = 0$, $P_i = a_1 + \dots + a_i$)。从起点 i 出发、走到第 t 个班组时，库存 = $P_t - P_{i-1}$ 。要全程非负，预存量 $\geq P_{i-1} - \min\{P_t \mid t \geq i\}$ ——每个起点的答案只取决于“ i 之后的前缀和最小值”。

于是倒序扫一遍，维护后缀最小值 $\text{sf}[i] = \min(P[i], \text{sf}[i+1])$ ，则 $\text{ans}[i] = \max(0, P[i-1] - \text{sf}[i])$ 。预处理 $O(n)$ 、每个答案 $O(1)$ ，总复杂度 $O(n)$ ，实测最大点 17.5ms。

这题的正解没有新数据结构，只有一个新视角：把“每个起点分别模拟”换成“一条前缀和曲线 + 后缀最值”。正问反想，是 $O(n^2)$ 到 $O(n)$ 之间最短的路。

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 n;
int a[N], p[N], sf[N];

void solve() {
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        p[i] = p[i - 1] + a[i];
    }
    sf[n] = p[n];
    for (int i = n - 1; i >= 1; i--) sf[i] = min(p[i], sf[i + 1]);
    for (int i = 1; i <= n; i++) {
        int v = p[i - 1] - sf[i];
        if (v < 0) v = 0;
        if (i > 1) cout << " ";
        cout << v;
    }
    cout << "\n";
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    freopen("stock.in", "r", stdin);
    freopen("stock.out", "w", stdout);
    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
只看 a_1 本身 ($\max(0, -a_1)$)	无视累积效应：库存是一路上的累计盈亏，单个交接量决定不了最低库存；20 个点无一碰巧	0
前缀和用 int 存	$ P $ 超 2.5×10^9 即溢出；前 6 档 ($n \leq 2500$) 值域小碰巧全对，第 7 档起全错	30

变式拓展

- 只问起点 1 的版本就是经典“预存 / 加油站”问题——一遍扫描取前缀和最小值即可。
- 若班组排成环、可以任选一个起点开始呢？（提高组衔接，只点名：环形结构常“倍长展开”处理。）
- 体会本题的方法论：问每个后缀 $\rightarrow$ 倒序预处理。对称地，“问每个前缀”就正序预处理——方向感比代码重要。

T4 程老师的跳格训练 (jump.cpp / jump.in / jump.out)

考点	难度	暴力分	时限 / 内存
DP / 状态设计	绿 ★★★	10	1000ms / 512MB

题意复述

格子 0 到 n ，从 0 出发恰好落到 n 。每步跳 $1 \sim k$ 格，落到格子 i 得 v_i 分（可负）。节奏奖励：本次步长与上一次相同，额外 $+b$ （连续多步保持同一长度，每步都拿；第一步没有奖励）。求最大总积分。

输入输出与数据范围

$n \leq 10^5$, $k \leq 300$, $b \leq 10^5$, $|v_i| \leq 10^4$ 。输出一行最大总积分。

样例与解释

样例输入	样例输出
5 2 3 1 2 -5 4 2	16
4 2 5 3 -1 -1 3	19

样例 1：最优是每步跳 1 格走 $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$ ：落格分 $1 + 2 - 5 + 4 + 2 = 4$ ，第 2~5 步各 $+3$ 奖励共 12，总计 16。改走 $0 \rightarrow 2 \rightarrow 4 \rightarrow 5$ 落格分 8 更高，但奖励只有 1 次，总计 11——奖励会改变最优路线。

样例 2：同样全跳 1 格最优： $3 - 1 - 1 + 3 = 4$ ，后三步奖励 $3 \times 5 = 15$ ，总计 19。

部分分策略（先保底，再冲刺）

指数 DFS：枚举每一步的步长（ $1 \sim k$ ），递归到 n 为止取最大。跳法数是 " k 步斐波那契" 型增长， $n = 8$ 、 $k = 3$ 时仅几百种，实测过测试点 1-2，保底 10 分；第 3 点（ $n = 32$ ）起跳法数爆炸，干净 TLE。

教学点：DFS 枚举是“定义即代码”，10

分拿到手就该数状态了：后面的选择只和“到哪个格子、上一步多长”有关——这就是 DP 状态的雏形。

满分思路

状态设计： $dp[i][l]$ = 落到格子 i 、且最后一步长度为 l

的最大积分。转移看“上一步”：换长度，从历史最优接入；不换长度，额外拿奖励 b —— $dp[i][l] = v_i + \max(mx[i-l], dp[i-l][l] + b)$ ，其中 $mx[j] = \max_l dp[j][l]$ 。

为什么 mx 能吸收“换长度”的所有历史？——换长度时上一步具体多长已经不重要，只要 $mx[i-l]$ 最大即可；只有“延续同长度”的候选需要单独加 b 比较。两者取 $\max$ ，转移 $O(1)$ ，总复杂度 $O(nk) = 3 \times 10^7$ ，实测最大点约 115ms。

空间细节： $dp[i][l]$ 只依赖第 $i-l$ 行（ $1 \leq k \leq 300$ ），开 $dp[301][301]$ 滚动数组即可， $i \% 301$ 定位槽位；不可达状态置 $-INF$ ，注意 $-INF$ 加 b 会溢出，要先判可达。

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;
const int R = 301;           // 滚动槽数 (> k 上限 300)
const int K = 305;

int n, k, b;
int v[N], mx[N];             // mx[i] = max_l dp[i][l]
int dp[K][K];                // dp[i % R][l] = 落到 i、末跳长 l 的最大积分

void solve() {
    cin >> n >> k >> b;
    for (int i = 1; i <= n; i++) cin >> v[i];
    mx[0] = 0;
    const int NEG = -(1LL << 60);
    for (int i = 0; i < K; i++)
        for (int j = 0; j < K; j++) dp[i][j] = NEG;
    for (int i = 1; i <= n; i++) {
        int bs = i % R;
        int bt = NEG;
        for (int l = 1; l <= k; l++) {
            if (i - l < 0) break;
            int cd = mx[i - l];
            if (i - l >= 1) {
                int ps = (i - l) % R;
                if (dp[ps][l] > NEG / 2) cd = max(cd, dp[ps][l] + b);
            }
            dp[bs][l] = v[i] + cd;
            bt = max(bt, dp[bs][l]);
        }
        mx[i] = bt;
    }
    cout << mx[n] << "\n";
}

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    freopen("jump.in", "r", stdin);
    freopen("jump.out", "w", stdout);
    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
忽略节奏奖励（按 $b = 0$ 做普通跳格 DP）	b 在全部 20 个点都进入最优结构（全 1 格吃满奖励常常是答案）；20 点全 WA	0
每步贪心跳到 v 最大的格子	局部最优不构成全局最优——为高分格可能牺牲整串奖励；只有最小点碰巧	5

变式拓展

- 奖励改成"与上一步长度不同才 +b": 转移变成 $\max$ (除自身长度外的最优 + b, 自身长度) ——维护 mx 的最大值与次大值即可, 想想为什么。
- 若 $k = n$ (一步可直达), 直接跳过去是不是最优? ——不一定: 直达只拿 v_n , 多跳几段可能多拿奖励。性质 B 只是数据形态, 不是结论。
- 与 Day 1 T3 补给站、Day 3 T3 值日表串起来: DP 的状态 = 题目规则的"记忆"。规则多一条 (记住上一步长度), 状态多一维——这就是状态设计的全部秘密。

本场得分策略小结

- 基本盘 240 分：T1 (100) + T2 (100) 两道签到全拿，T3 按定义模拟 (30) + T4 指数 DFS (10)。240 只是保底——T3、T4 的正解分别考"倒着想"和"状态设计"，想通就是满分。
 - 考场时间建议：T1、T2 各 20 分钟内解决（规则直译 + 样例手推）；T3 先写 $O(n^2)$ 保底 30，再推后缀最小值；T4 先写 DFS 保底 10，再数状态设计 DP。
 - 检查清单：文件名 borrow / badge / stock / jump 与 freopen 拼写；T1 第二行的空行；T2 必须 string 读入（前导零）；T3 的 long long ($|P|$ 可达 10^{11})；T4 的 $-\text{INF}$ 不可加 b。
 - 与下午专题的衔接：下午图论把"状态"看成点、"转移"看成边——T4 的每个（格子，上一步长度）其实就是图上的一个点，DFS 能走的路就是 DP 的转移。带着这个视角去听课。
- 本场题眼：T1 状态数组天然有序，空行必须输出；T2 号码一律 string 读，前导零是生命线；T3 正问反想，后缀最值一遍出全部答案；T4 规则多一条、状态多一维，换长度的历史由 mx 吸收。