2026 HGNU-ACM 结训赛

题解 · A–J

时间：2026 年 11 月 8 日 18:00–21:00

出题：HGNU-ACM 实验室算法组，GPT-6

验题：HGNU-ACM 实验室算法组，GPT-6，deepseek-v4-flash





阅读说明

- 每题先把故事改写成可计算的条件或目标，再说明算法与边界。
- 代码框展示与题库标程一致的核心逻辑；完整可编译程序位于各题 `src/std.cpp`。
- 正式题号采用当前正赛 A–J 顺序，与单题 PDF、合订题面一致。

题目顺序

A 贪吃的乐乐　B 入场票价　C 刘学长开播啦　D 迟到登记　E 简单字符串
F 纸杯回收站　G 小新再次来袭　H 证书评定　I 参加比赛　J 字符串反转

A ｜贪吃的乐乐：题意转化

- 有 m 份食物，每份完整吃完用 t 分钟，总时长 120 分钟。
- 求剩余份数；仅当剩余份数严格大于 k 时扣钱。

解法要点

若 $t > 0$，最多完整吃完 $\lfloor 120/t \rfloor$ 份；还不能超过已有的 m 份。若 $t = 0$，全部吃完。

公式

$$e = \begin{cases} m, & t = 0, \\ \min(m, \lfloor 120/t \rfloor), & t > 0, \end{cases} \qquad r = m - e.$$

输出 r，再输出 $[r > k]$。

复杂度：$O(1)$ 时间，$O(1)$ 空间

A | 核心代码与边界

```
int eaten = (t == 0) ? m : min(m, 120 / t);
int left = m - eaten;
cout << left << '\n' << (left > k ? 1 : 0) << '\n';
```

- $t = 0$ 必须单独处理，避免除以零。
- $120/t$ 向下取整；剩余时间不足一份时，该份仍算未吃完。
- $r = k$ 时不扣钱，不要写成 $r \geq k$。

B | 入场票价：题意转化

- 输入年龄、入场时刻、学生证、优惠券；输出最终票价。
- 规则有先后：基础票价 $\rightarrow$ 学生优惠 $\rightarrow$ 高峰加价 $\rightarrow$ 优惠券。

解法要点

按题面顺序逐层修改一个价格变量。学生优惠只作用于 18–59 岁；高峰加价只作用于非免费票。

关键边界

年龄 5/6、17/18、59/60；时刻 16/17、20/21；免费票遇到优惠券。

复杂度：$O(1)$ 时间，$O(1)$ 空间

B | 核心代码与规则顺序

```
int price;
if (age < 6) price = 0;
else if (age < 18) price = 40;
else if (age < 60) price = student ? 50 : 80;
else price = 30;

if (price > 0 && 17 <= hour && hour <= 20)
    price += 20;
if (coupon) price = max(0, price - 15);
```

先完成免费票和年龄判断，再考虑加价；优惠券在最后使用，结果不能小于零。

C | 刘学长开播啦：题意转化

- 购买 h 发子弹、t 个陷阱：$0 \le h \le 8,\ 0 \le t \le 2$。
- 预算约束：$150h + 200t \le M$；火力：$F = 55h + 30t$。
- 目标是先最大化 F，同分再最大化 h，仍同分再最大化 t。

解法要点

候选方案只有 $9 \times 3 = 27$ 个，直接枚举全部 (h, t)。不满足预算的跳过，其余按 (F, h, t) 的字典序取最大。

复杂度：$O(27) = O(1)$ 时间，$O(1)$ 空间

C | 核心代码与比较次序

```
long long bestF = -1, bestH = 0, bestT = 0;
for (long long h = 0; h <= 8; ++h)
  for (long long t = 0; t <= 2; ++t) {
    if (150*h + 200*t > M) continue;
    long long F = 55*h + 30*t;
    if (F > bestF ||
        (F == bestF &&
         (h > bestH || (h == bestH && t > bestT)))) {
      bestF = F; bestH = h; bestT = t;
    }
  }
```

$(0,0)$ 永远合法，因此一定能找到答案；金币可以有剩余，无需恰好花完。

D | 迟到登记：题意转化

- 对到达时刻 a_i，只有 $a_i > t$ 才迟到。
- 迟到人数是满足条件的项数；迟到总分钟数是 $\sum_{a_i>t}(a_i - t)$。

解法要点

读入时逐项判断并累加，无需存储整个数组。

边界与类型

$a_i = t$ 不迟到；总分钟数可能达到 10^{14} 量级，需用 `long long`。

复杂度：$O(n)$ 时间，$O(1)$ 额外空间

D | 核心代码

```
long long count = 0, total = 0;
for (int i = 0; i < n; ++i) {
    long long arrival;
    cin >> arrival;
    if (arrival > t) {
        ++count;
        total += arrival - t;
    }
}
cout << count << ' ' << total << '\n';
```

条件统一用严格大于；不要把所有人的到达时刻都加入总迟到时间。

E | 简单字符串：题意转化

- 把单词统一为全大写或全小写，使改变字符的数量最少。
- 若大写字母数 U 多于小写字母数 L，转大写；否则转小写。

解法要点

转大写要改动 L 个字符；转小写要改动 U 个字符。因此选择改动更少的方向。$U = L$ 时题面指定转小写。

复杂度：$O(|s|)$ 时间，$O(1)$ 额外空间

E | 核心代码与并列情况

```
int upper = 0, lower = 0;
for (char ch : s) {
    if (isupper((unsigned char)ch)) ++upper;
    else ++lower;
}
for (char& ch : s) {
    if (upper > lower) ch = toupper((unsigned char)ch);
    else ch = tolower((unsigned char)ch);
}
cout << s << '\n';
```

严格比较 $U > L$ 很重要；等量时走小写分支。

F ｜纸杯回收站：题意转化

- 每个箱子有 a_i 个纸杯，每 k 个换一张贴纸。
- 箱子分别兑换；余数不能跨箱合并。

解法要点

第 i 个箱子贡献 $\lfloor a_i/k \rfloor$ 张贴纸、$a_i \bmod k$ 个剩余纸杯。分别求和。

边界与类型

$k = 1$ 时余数全为零；$a_i < k$ 时该箱无法兑换。两个总量使用 `long long`。

复杂度：$O(n)$ 时间，$O(1)$ 额外空间

F ｜核心代码

```
long long stickers = 0, remaining = 0;
for (int i = 0; i < n; ++i) {
    long long cups;
    cin >> cups;
    stickers += cups / k;
    remaining += cups % k;
}
cout << stickers << ' ' << remaining << '\n';
```

不要先计算所有纸杯总数再除以 k；那会错误地把不同箱子的余数合并。

G | 小新再次来袭：题意转化

- 每次操作必须同时把所有 a_i 变为 $|a_i - 2|$；可以操作任意次。
- 选择操作次数，使最终某个值出现次数最大。

解法要点

不模拟大量操作。把数按 $a_i \bmod 4$ 分类，研究各类最终能否在同一步变成相同值。

轨迹观察

奇数最终到 1；偶数最终在 0 和 2 之间往返。

G | 为什么是模 4 分类

- 所有奇数（余数 1 或 3）经过足够多次同步操作后都变成 1，可合并为一类。
- 偶数余数 0 与余数 2 每步交换相位：一个最终为 0 时，另一个为 2，两类不能合并。
- 同一偶数类里的数最终能在同一步到达相同的 0 或 2。

结论

$\max(c_0,\ c_2,\ c_1 + c_3)$，其中 c_r 是模 4 余 r 的个数。

复杂度：$O(n)$ 时间，$O(1)$ 额外空间

G | 核心代码与反例

```
int count[4] = {};
for (int i = 0; i < n; ++i) {
    int x;
    cin >> x;
    ++count[x % 4];
}
cout << max({count[0], count[2],
             count[1] + count[3]}) << '\n';
```

例如 $[6, 7, 8]$ 中，$6 \equiv 2 \pmod 4$、$8 \equiv 0 \pmod 4$，两者始终不同；7 为奇数。答案是 1，不能把所有偶数直接合并。

H | 证书评定：题意转化

- 输入成绩 p、出勤次数 a、违规标志 v。
- 从高到低检查规则，只执行第一条满足的规则。

解法要点

把规则写成一条按优先级排列的 `if / else if` 链。违规先于出勤；出勤先于成绩；PERFECT 先于 GOLD。

易错边界

$v = 1$ 且满分仍是 INVALID；$a = 7$ 且高分仍是 INELIGIBLE；$p = 100, a = 10$ 才是 PERFECT。

复杂度：$O(1)$ 时间，$O(1)$ 空间

H | 核心代码：保留优先级

```
if (v) cout << "INVALID\n";
else if (a < 8) cout << "INELIGIBLE\n";
else if (p == 100 && a == 10) cout << "PERFECT\n";
else if (p >= 90) cout << "GOLD\n";
else if (p >= 75) cout << "SILVER\n";
else if (p >= 60) cout << "BRONZE\n";
else cout << "NONE\n";
```

使用互斥的 `else if`，避免多次输出；阈值从高到低排列。

I | 参加比赛：题意转化

- 对每道题，三人分别给出 $0/1$，表示是否有解题思路。
- 至少两人会做，才将这道题计入答案。

解法要点

对每行计算 $a+b+c$；若和 ≥ 2，答案加一。扫描所有题即可。

边界

$0,1,2,3$ 个确认者中，只有后两种应被计数。

复杂度：$O(N)$ 时间，$O(1)$ 额外空间

I | 核心代码

```
int answer = 0;
for (int i = 0; i < N; ++i) {
    int a, b, c;
    cin >> a >> b >> c;
    if (a + b + c >= 2) ++answer;
}
cout << answer << '\n';
```

输入中的每个数都是 0 或 1，因此求和正好等于“确认有思路的人数”。

J | 字符串反转：题意转化

- 给定数字字符串，先在原字符串中统计字符 3、字符 6、相邻子串 18。
- 最后再将整个字符串反转，按四行输出。

解法要点

扫描每个位置，分别判断单字符和相邻两字符。之后调用 `reverse`。

边界

长度为 1 时没有两字符子串；判断 18 前先确保 $i+1<|s|$。1818 含两处 18。

复杂度：$O(|s|)$ 时间，$O(1)$ 额外空间

J | 核心代码：先计数再反转

```
int c3 = 0, c6 = 0, c18 = 0;
for (size_t i = 0; i < s.size(); ++i) {
    if (s[i] == '3') ++c3;
    if (s[i] == '6') ++c6;
    if (i + 1 < s.size() &&
        s[i] == '1' && s[i + 1] == '8') ++c18;
}
reverse(s.begin(), s.end());
cout << c3 << '\n' << c6 << '\n'
     << c18 << '\n' << s << '\n';
```

不要在反转后再统计 18；反转会改变相邻子串。

THANK YOU

题解 · A–J

时间：2026 年 11 月 8 日 18:00–21:00

出题：HGNU-ACM 实验室算法组，GPT-6

验题：HGNU-ACM 实验室算法组，GPT-6，deepseek-v4-flash



