P16228 [蓝桥杯 2026 省 A] 读取指令
题解

1 题意概述

硬盘有 N 个扇区，第 i 个扇区的数据量为 $C \times i$ 字节。每次读取指令可以读取一段连续的扇区区间 $[l, r]$，每个扇区最多读一次。问恰好读取 W 字节最少需要多少次读取指令。无解输出 -1。

2 第一层思考：等价转化，剥离常数 C

所有扇区的数据量都是 C 的倍数。如果 W 不是 C 的倍数，那么无论怎么选取扇区区间，总字节数都是 C 的倍数，永远凑不出 W，此时答案为 -1。

若 $C \mid W$，令 $K = W/C$。问题等价转化为：

> 从数列 $1, 2, 3, \ldots, N$ 中，选取若干个**不相交的连续子段**，使得它们的元素之和恰好为 K，求最少的子段数。

特别地，若 $K = 0$，不需要任何读取指令，答案为 0；若 $K > \frac{N(N+1)}{2}$（超过整块硬盘的数据总量），答案为 -1。

3 第二层思考：答案只可能是 0、1、2 或 -1

对于 $0 < K \le \frac{N(N+1)}{2}$ 的情况，我们断言：

> 答案要么是 1（存在单个连续子段凑出 K），要么是 2。

这意味着我们只需要判断” 能否只用一次指令搞定”，如果不能，答案就是 2。

3.1 为什么答案至多为 2？——证明

取最小的正整数 r，使得 $1 + 2 + \cdots + r \ge K$（记此和为 T_r）。若 $T_r = K$，则 $[1, r]$ 即为解，1 次指令。以下证明 $T_r > K$ 时只需 2 次指令。

第一步： $D < r$。　令超出量 $D = T_r - K$。由 r 的**最小性**，少取一个元素就不够：$T_{r-1} < K$。因此

$$D = T_r - K < T_r - T_{r-1} = r$$

又 $D \ge 1$，故 $1 \le D \le r - 1$。

如图 1 所示，K 被夹在两个相邻三角数 T_{r-1} 与 T_r 之间，二者的间距恰好是 r。D 只是 K 到 T_r 的距离，自然**严格小于**这个间距。

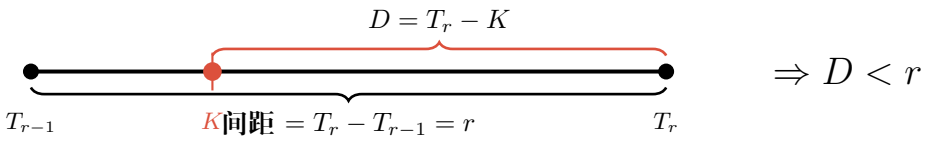

Figure 1: K 被夹在 T_{r-1} 与 T_r 之间。D 是 K 到右端 T_r 的距离，严格小于总间距 r。

第二步：D 一定在 $[1,r)$ 中。 由第一步，$1 \le D < r$。而 $[1,r)$ 中包含了**每一个**小于 r 的正整数——D 一定是其中之一，即 D 是前缀 $\{1,2,\ldots,r\}$ 中的某个元素。从 $[1,r]$ 中去掉 D，区间分裂为 $[1,D-1]$ 和 $[D+1,r]$ 两段（如图 2 所示），二者之和 $= T_r - D = K$，恰好 2 次指令。

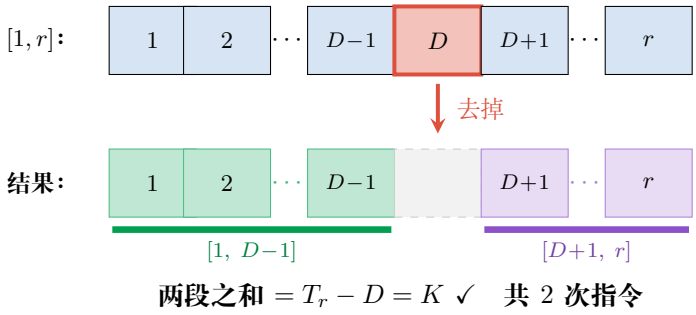

Figure 2: 从 $[1,r]$ 中去掉元素 D，区间分裂为 $[1,D-1]$ 与 $[D+1,r]$ 两段，总和恰为 K。

$r \le N$ 由最小性保证（若 $r > N$ 则 $T_r > T_N \ge K$，矛盾）。证毕。

4 第三层思考：如何判断答案是否为 1

对 $1,2,\ldots,N$ 这个递增正整数数列，连续子段 $[l,r]$ 的元素和为：

$$S(l,r) = \sum_{i=l}^{r} i = \frac{r(r+1)}{2} - \frac{(l-1)\cdot l}{2}$$

我们需要判断是否存在 $1 \le l \le r \le N$ 使得 $S(l,r) = K$。利用前缀和的单调性，可以用**双指针**（滑动窗口）在 $O(N)$ 时间内完成判定。

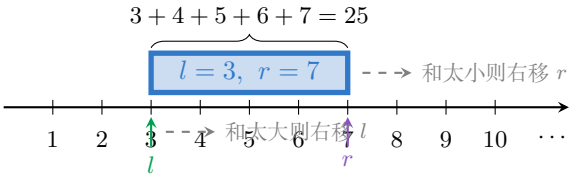

Figure 3: 双指针在数列 $1,2,\ldots,N$ 上滑动，寻找和恰好为 K 的连续子段。

代码实现中使用了等价的方式：枚举左端点 l，利用前缀和数组的单调性，对目标值 $K + \mathrm{pre}[l-1]$ 进行二分查找，判断其是否恰好等于某个 $\mathrm{pre}[r]$。二分查找与双指针的时间复杂度相同，均为 $O(N)$（二分是 $O(N \log N)$，但不影响最终通过）。

5 完整算法流程

1. 若 $C \nmid W$，输出 -1。
2. 令 $K \leftarrow W/C$。若 $K = 0$，输出 0。
3. 若 $K > \frac{N(N+1)}{2}$，输出 -1。
4. 用双指针/二分查找判断是否存在连续子段和为 K。若存在，输出 1。
5. 否则，输出 2。

6 代码实现

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

```cpp
using ll = long long;

struct Solve {
  ll N, W, C;
  vector<ll> A;
  vector<ll> preA;

  Solve() {
    cin >> N >> C >> W;
    A.resize(N + 2);
    if (W % C != 0) {
      cout << -1 << "\n";
      return;
    }
    ll chu = W / C;
    if (chu == 0) {
      cout << 0 << "\n";
      return;
    }
    if (chu > (N + 1) * N / 2) {
      cout << -1 << "\n";
      return;
    }
    for (int i = 1; i <= N; ++i)
      A[i] = i;
    preA.resize(N + 2);
    partial_sum(A.begin(), A.end(), preA.begin());

    // 枚举左端点，二分查找右端点
    for (int i = 1; i <= N; ++i) {
      ll tar = chu + preA[i - 1];
      if (binary_search(preA.begin(), preA.end(), tar)) {
        cout << 1 << "\n";
        return;
      }
    }
    // 由构造性证明，此时答案必为 2
    cout << 2 << "\n";
  }
};

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

7 复杂度分析

- **时间复杂度**：单组 $O(N \log N)$（枚举 N 个左端点，每次二分 $O(\log N)$）。T 组合计 $O(T \cdot N \log N)$。在 $N \le 10^5$、$T \le 10$ 的范围下约 1.7×10^7 次运算，远低于时限。若使用双指针可进一步优化至单组 $O(N)$。
- **空间复杂度**：$O(N)$，用于存储前缀和数组。