题解报告

题目概览

题号	名称	类型	大致 rating
A	大雪封路	并查集	1800
B	本题由 ChatGPT 生成	签到	500
C	简单的数学题	动态规划，数论	2100
D	圆的面积	数学	1600
E	英雄熟练度	构造	1200
F	乘号漂移	枚举	800
G	铲雪	贪心	1200
H	厨房安排	二分	1400
I	半排列	数据结构	1400
J	洞窟探险	图论	1600

Problem A. 大雪封路

不能连续走被标记的点，也就是说如果某一条边所连接的两个顶点一旦都被标记，就可以当成一次删边操作。将询问离线，从后往前倒推，把删边操作反向为加边操作，就能够很方便地用并查集来维护连通关系。

对于每个时刻，用数组存储当前时刻的两种询问。对于询问 1 u，存储这个询问导致哪些边被删了。要查询哪些边被删，可以遍历 u 所连接的边。因为保证了每个顶点最多只会有一个询问 1，所以最多只会遍历 $2 \times M$ 条边。

反向状态的一开始，在并查集中把结束状态时依然连通的顶点合并，在倒推过程中逐渐加边，对每个询问 2 保存答案。

Problem B. 本题由 ChatGPT 生成

输出 “The Sound of Music”。

Problem C. 简单的数学题

考虑固定 N, K，定义数列 a_i 为选了 K 个数之后，这些数的和模 N 余 i 的方法数，则 $a_i = a_{i+\gcd(N,K)}$。

证明：对于一个集合 $\{v_1, v_2, ..., v_K\}$，如果把每个元素增加 1，得到 $\{v_1 + 1, v_2 + 1, ..., v_K + 1\}$（模 N）。重复这个操作 N 次就能得到原本的集合。根据裴蜀定理，每次操作后得到的集合元素总和与操作之前的元素总和在模 $\gcd(N, K)$ 下同余，因此方法数相同。

因此，定义 $dp[i][j]$ 为选取 i 个数字，使得总和模 $\gcd(N, K)$ 为 j 的方法数。在转移时把所有的 $a + b \times \gcd(N, K)$ 一起考虑。因为 $\gcd(N, K) \le K$，所以时间复杂度为 $\mathcal{O}(K^4)$。

```
void solve() {
    int n, K;
    cin >> n >> K;
    int g = gcd(n, K);
    int num = n / g;
    vector<Z> num_C_K(K + 1, 1);
    for (int i = 1; i <= K; ++i) {
        num_C_K[i] = Z(num - i + 1) / i * num_C_K[i - 1];
    }
    vector<vector<Z>> dp(K + 1, vector<Z>(g, 0));
    dp[0][0] = 1;
    for (int v = 0; v < g; ++v) {
        for (int k = K - 1; k >= 0; --k) {
            for (int i = 0; i < g; ++i) {
                if (dp[k][i].val() == 0) {
                    continue;
                }
                for (int dk = 1, x = i; dk + k <= K && dk <= num; ++dk) {
                    x = (x + v) % g;
                    dp[k + dk][x] += dp[k][i] * num_C_K[dk];
                }
            }
        }
    }
    cout << dp[K][0] / num << endl;
}
```

Problem D. 圆的面积

$$
\begin{aligned}
\text{原式} &= \sum_{i=1}^{N}\sum_{j=1}^{N} S(i,j) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{\pi}{4}((x_i-x_j)^2+(y_i-y_j)^2) \\
&= \frac{\pi}{4}\sum_{i=1}^{N}\sum_{j=1}^{N}((x_i-x_j)^2+(y_i-y_j)^2)
\end{aligned}
$$

把 $\frac{\pi}{4}$ 去掉就是要输出的结果。而

$$
\sum_{i=1}^{N}\sum_{j=1}^{N}((x_i-x_j)^2+(y_i-y_j)^2) = \sum_{i=1}^{N}\sum_{j=1}^{N}(x_i-x_j)^2 + \sum_{i=1}^{N}\sum_{j=1}^{N}(y_i-y_j)^2
$$

也就是说 x_i 和 y_i 对答案的贡献是独立的，可以拆开分别计算。

现在考虑 x 坐标对答案的贡献：

$$\begin{aligned}\sum_{i=1}^{N}\sum_{j=1}^{N}(x_i-x_j)^2 &= 2\sum_{i=1}^{N}\sum_{j=i+1}^{N}(x_i-x_j)^2 \\ &= 2((N-1)\sum_{i=1}^{N}x_i^2-2\sum_{i=1}^{N}\sum_{j=i+1}^{N}x_ix_j) \\ &= 2(N-1)\sum_{i=1}^{N}x_i^2-4\sum_{i=1}^{N}\sum_{j=i+1}^{N}x_ix_j\end{aligned}$$

前一项可以 $\mathcal{O}(N)$ 计算出来，现在观察后一项：

$$\begin{aligned}\sum_{i=1}^{N}\sum_{j=i+1}^{N}x_ix_j = \quad & x_1 \\ & + x_2 \times x_1 \\ & + x_3 \times (x_1+x_2) \\ & + ... \\ & + x_{N-1} \times (x_1+x_2+...+x_{N-2}) \\ & + x_N \times (x_1+x_2+...+x_{N-1})\end{aligned}$$

也就是说这一项也可以在计算过程中用求前缀和的方式 $\mathcal{O}(N)$ 计算出来。

那么 x 坐标对答案的贡献 $\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N}(x_i-x_j)^2$ 就能在时间限制内计算出来了，再计算一次 y 坐标的贡献就能得到答案。

Problem E. 英雄熟练度

有很多种方法能保证一定能成功。其中的一种是：一直增加最小的数，直到有两个数相等，然后一直使这两个数减小。

也可以每次都随机选择，只要不让数小于 0 就输出，直到满足要求。

Problem F. 乘号漂移

枚举乘号的新位置，判断是否出现一个结果与原式相等。使用 substr 和 stoi 或 stoll 能避免写很多模拟。

Problem G. 铲雪

最终的目标高度 k 可以用总和除以 $(N \times M)$ 得到。接下来要检查是否能通过一定的操作把所有元素变成 k。

做法 1：对于所有非最后一行、非最后一列的元素操作，使得值变为 k。最后再检查最后一行、最后一列是否全为 k。

做法 2：每次操作，相同行或列的总和不会改变。检查每一行的总和是否相等、每一列的总和是否相等。

Problem H. 厨房安排

二分答案，使用优先队列计算每个答案的制作的最小时间。

Problem I. 半排列

从左到右依次用树状数组或线段树计算出位置 i 之前比 x_i 大的数字的个数 $cnt[i]$。

在处理过程中记录每个元素出现的下标，假设一对相同的数出现在 i 和 j ，他们对答案的贡献是 $cnt[j]-cnt[i]$。

Problem J. 洞窟探险

因为可以在一个强连通分量内随便乱走，所以可以把强连通分量缩点，缩点后的点权为原先的点权与边权之和。

缩点之后得到一张有向无环图，按照拓扑序 dp。

如果拓扑序内存在边 u,v,w，则 $dp[v]=\max(dp[u]+c[v]+w)$，这里的 u,v,w,c 都是缩点后的新值。