[2024暑期交流小组] CSP-S & NOIP模拟赛 3 题解

高义雄 July 23, 2024

By Yixiong Gao, July 23th, 2024

总体情况

- 获得 200 分以上的选手 11 人
- 平均分 120.1
- 第一题通过 16 人，平均分 50.3
- 第二题通过 2 人，平均分 38.5
- 第三题通过 6 人，平均分 33.5
- 第四题通过 0 人，平均分 20

A - 序列异或（xor）

小 Z 有一个长度为 n 的数组 $a_1, a_2, \cdots, a_n$

现在小 Z 想知道有多少个满足条件的四元组 (i, j, k, l)，满足 $1 \leq i < j < k < l \leq n$，同时 $a_i \oplus a_j \oplus a_k \oplus a_l = 0$。

其中 $\oplus$ 表示异或运算，在 C++ 中用 `^` 运算符表示。

A - 序列异或（xor） - 第 1 档部分分

> 测试点 $1-2$ 满足，$n \leq 100, a_i \leq 1000$。

直接 $O(n^4)$ 循环枚举所有可能的答案即可。

A - 序列异或（xor）- 第 2 档部分分

> 测试点 $3-4$ 满足，$n \leq 1000, a_i \leq 1000$。

考虑 $O\left(n^3\right)$ 枚举 i, j, k，此时我们想要知道有多少个 l, 满足 $a_l = a_i \oplus a_j \oplus a_k$

预处理出后缀 $a_{k+1}, a_{k+2}, \cdots, a_n$ 中每种数出现的次数，就可以 $O(1)$ 回答 l 的个数。

A - 序列异或（xor） - 满分

考虑折半思想。

考虑从 1 到 n 枚举第三个元素的位置 k 。对于 k, 用 $cnt[x]$ 记录一下 $i < j < k$ 中所有 $a_i \oplus a_j = x$ 的二元组 (i, j) 的对数。

然后我们再枚举最后一个元素 l，然后查询 $cnt[a[k] \oplus a[l]]$ 的次数。

枚举完这个 k 之后, 循环会来到 $\mathrm{k}+1$, 此时只需要将新的 $a_i \oplus a_j$ 加入到 cnt 数组。

时间复杂度为 $O\left(n^2\right)$ 。

A - 序列异或（xor） - 核心代码

```
cin >> n;
for(int i = 1; i <= n; ++i) cin >> a[i];
for(int i = 1; i <= n; ++i) {
    for(int j = i + 1; j <= n; ++j) ans += cnt[a[i] ^ a[j]];
    for(int j = 1; j < i; ++j) ++cnt[a[i] ^ a[j]];
}
cout << ans;
```

B - 自驾游 (trip)

比特国有 n 个城市，编号 $1, 2, \cdots, n$，由 m 条**单向**的公路连接。

第 i 条公路从第 a_i 个城市出发，前往第 b_i 个城市，路程是 d_i 米。

小 Z 决定从 S 号城市出发，最终到达 T 号城市（保证 $S \neq T$），并且你的车子初始速度 v 为 1 米每秒。

全国共有 p $(0 \leq p \leq n)$ 个维修站，第 i 个位于第 x_i 个城市，小 Z 可以停留在这里花 c_i 秒的时间将车升级，使得车速翻倍。小 Z 可以在某个维修站将车升级多次，但每次升级都会花 c_i 秒的时间。例如，假设小 Z 当前的车速为 v，并且恰好在第 x_i 个城市，那么他可以在第 i 个维修站花 $3c_i$ 的时间将车升级三次，使得车速变为 $2 \times 2 \times 2 \times v = 8v$ 。

B - 自驾游 (trip)

假设小 Z 的车速是 v 米每秒，那么安全通过一条长度为 d 米的公路所需的时间为 $\lceil \frac{d}{v} \rceil$ 秒（$\lceil x \rceil$ 代表最小的不小于 x 的整数）。然而，有些公路是平整的，而另一些则是坑坑洼洼的。每当小 Z 通过一条坑坑洼洼的路，车的所有升级都会损耗，通过这条路之后车速会立即变回 1 米每秒（但在通过这条路的过程中车速不会改变）。

现在小 Z 想知道，他到达 T 号城市最少需要花多少秒?

B - 自驾游 (trip) - 第 1 档部分分

> 对于 10% 的数据，保证 $n = 2, m = 1$。

判断是否连通，然后枚举翻倍次数即可。

B - 自驾游 (trip) - 第 2 档部分分

> 对于另外 20% 的数据，保证所有的 $d_i = 1$ 。

所有边长度都是 1，由于通过时间为上取整没必要提高车速。

相当于图没有边权求最短路，BFS 即可，复杂度 $O(n + m)$ 。

B - 自驾游 (trip) - 第 3 档部分分

> 对于另外 20% 的数据，保证 $p = 0$ 。

没有维修站，车速恒定为 1 ，使用 Dijkstra 算法求解最短路即可。

复杂度 $O(m \log n)$ 。

B - 自驾游 (trip) - 第 4 档部分分

> 对于另外 20% 的数据，保证 $d_i \leq 100$ 。

注意到车速超过 100 后通过每条路的时间就不会有变化了。

建立分层图，将图复制 100 层，第 x 层的第 u 个节点代表到达城市 u ，车速为 x 。

通过一条好的路：$(u, x) \to (v, x)$，边权为 $\lceil \frac{d_i}{x} \rceil$

通过一条不好的路：$(u, x) \to (v, 1)$，边权为 $\lceil \frac{d_i}{x} \rceil$

维修站升级车速：$(u, x) \to (u, 2x)$，边权为 c_i

然后在新的图上运行 Dijkstra 即可，复杂度 $O(\max d_i \times m \log n)$ 。

B - 自驾游 (trip) - 满分做法

注意到车速只会变成二倍和回到 1，因此车速一直是 2 的幂次。

建立分层图，将图复制 20 层，第 x 层的第 u 个节点代表到达城市 u，车速为 2^x。

通过一条好的路：$(u, x) \to (v, x)$，边权为 $\lceil \frac{d_i}{2^x} \rceil$

通过一条不好的路：$(u, x) \to (v, 0)$，边权为 $\lceil \frac{d_i}{2^x} \rceil$

维修站升级车速：$(u, x) \to (u, x + 1)$，边权为 c_i

然后在新的图上运行 Dijkstra 即可，复杂度 $O(\log d_i \times m \log n)$。

C - 车马象 (chess)

一共 m 个问题，第 i 个问题是：

- 如果棋盘大小为 $a_i \times b_i$ （格点），棋盘上只有一个 (车/马/象) 在第 c_i 行第 d_i 列的位置。那么经过 e_i 步，这个棋子可以形成多少种不同的路径?

具体的，我们将棋子 e_i 步的移动经过的 $e_i + 1$ 个点的坐标记录为一个序列。

我们认为两条路径不同，当且仅当序列中至少存在一个位置坐标不同。

C - 车马象 (chess) - 第 1 档部分分

> 数据范围：$a_i * b_i \leq 25$, $e_i \leq 8$，只有马和象。

DFS 所有的路径即可。

马复杂度 $\mathrm{O}(8^{e_i})$，象复杂度 $\mathrm{O}(4^{e_i})$。

C - 车马象 (chess) - 第 2 档部分分

> 数据范围：$a_i * b_i \leq 100,\ e_i \leq 100$ 。

设 $f[i][j][k]$ 表示当前在 (i, j)，已经走了 k 步的方案数。

按照步数分层，从可能的前置状态转移过来即可。

车复杂度 $\mathrm{O}(e_i(a_i * b_i)^2)$，象/马复杂度 $\mathrm{O}(e_i a_i b_i)$ 有常数。

C - 车马象 (chess) - 第 3 档部分分

> 数据范围：$a_i * b_i \leq 100,\ e_i \leq 10000$ 。

一边 DP 一边维护 f 数组每一行/每一列的和。

将车的转移复杂度降低到 $\mathrm{O}(1)$，总复杂度 $\mathrm{O}(e_i a_i * b_i)$ 有常数。

C - 车马象 (chess) - 第 4 档部分分

> 数据范围：$a_i * b_i \leq 100,\ e_i \leq 10^9$，只有车。

答案是 $(n+m-2)^{e_i}$，快速幂即可。

C - 车马象 (chess) - 满分做法

> $1 \leq m \leq 50$, $1 \leq c_i \leq a_i$, $1 \leq d_i \leq b_i$, $1 \leq a_i * b_i \leq 100$, $1 \leq e_i \leq 10^9$

将可以转移的位置建图连边。

邻接矩阵快速幂加速递推，总复杂度 $\mathrm{O}((a_i * b_i)^3 \log e_i)$ 有常数。

D - 距离 (distance)

树由 n 个节点构成，编号 $1, 2, \cdots, n$，其中 1 号节点是树的根。

我们称节点 u 是节点 v 的祖先，当且仅当 u 在 v 到根的路径上。

我们定义 $subtree_u$ 表示 u 的子树里的点集，也就是所有以 u 为祖先的节点集。

现在小 Z 给了每个节点 i 一个权值 a_i，小 Y 给了每个节点 i 一个权值 b_i。

我们定义一个节点 u 的价值为

$$ans_u = \sum_{x \in subtree_u} \sum_{y \in subtree_u} \min\{|a_x - a_y|, |b_x - b_y|\} \mod 10^9 + 7$$

请你对每个节点 u 求出其价值 ans_u。

D - 距离 (distance) - 第 1 档部分分

> 对于 40% 的测试点，保证 $1 \leq n \leq 100, 1 \leq a_i, b_i \leq 10^9$ 。

直接暴力枚举 u ，然后枚举子树内节点 x, y 统计即可，复杂度 $O(n^3)$

D - 距离 (distance) - 第 2 档部分分 - 做法 1

> 对于另外 30% 的测试点，保证 $1 \leq n \leq 5000, 1 \leq a_i, b_i \leq 10^9$ 。

考虑枚举两个点 x, y ，那么他们会为其 LCA 及向上的所有节点带来贡献。

将贡献标记到 LCA 处，最后求一遍子树和即可。复杂度 $O(n^2 \log n)$ 。

D - 距离 (distance) - 第 2 档部分分 - 做法 2

本题中要求$\mathbf{min}\{|a_i - a_j|, |b_i - b_j|\}$，通过$a + b = \mathbf{max}(a, b) + \mathbf{min}(a, b)$转换可得。

考虑把(a_i, b_i)二元组视作二维平面上的整点，两个点(i, j)之间的产生的贡献实际上等于两个整点之间的切比雪夫距离，即$\mathbf{max}\{|a_i - a_j|, |b_i - b_j|\}$。

通过旋转坐标系可以实现切比雪夫距离转曼哈顿距离：令$p_i = \frac{a_i + b_i}{2}, q_i = \frac{a_i - b_i}{2}$，则$\mathbf{max}\{|a_i - a_j|, |b_i - b_j|\} = |p_i - p_j| + |q_i - q_j|$。

上式可以通过拆绝对值符号证明。

$$\begin{aligned}\mathbf{min}\{|a_i - a_j|, |b_i - b_j|\} &= |a_i - a_j| + |b_i - b_j| - \mathbf{max}\{|a_i - a_j|, |b_i - b_j|\} \\ &= |a_i - a_j| + |b_i - b_j| - (|p_i - p_j| + |q_i - q_j|)\end{aligned}$$

D - 距离 (distance) - 第 2 档部分分 - 做法 2

> 对于另外 30% 的测试点，保证 $1 \leq n \leq 5000, 1 \leq a_i, b_i \leq 10^9$ 。

处理出每个子树的数集，排序统计。复杂度 $O(n^2 \log n)$ 。

D - 距离 (distance) - 满分做法

考虑解决 4 个形如 $\sum_{i\in\text{subtree } u}\sum_{j\in\text{subtree } u}|a_i-a_j|$ 的问题即可。

注意到，如果我们用线段树维护出关于 a_i 的桶数组，当我们插入一个新的数字 a_j 的时候，我们只需要知道小于 a_j 的元素的和，以及小于 a_j 的元素的数量，就可以知道 a_j 对新的答案产生的贡献，大于 a_j 的部分同理。

可以通过树上启发式合并的方式，对于每一个点维护出子树的答案。

时间复杂度为 $O(n\log^2 n)$。如果实现细节极其优秀，有可能通过。

D - 距离 (distance) - 满分做法

注意到在启发式合并的过程中一些性质被我们浪费了，当我们维护出一个关于a_i的有序数组时，答案可以分治的去维护。

假设我们当前维护出的数组为 a，长度为 n，取中点 $\mathrm{m} = \frac{n+1}{2}$。那么对于 $a[1,n]$ 的答案可以由以下式子表示：

$$\begin{aligned}\operatorname{ans} a[1,n] =& \operatorname{ans} a[1,m] + \operatorname{ans} a[m+1,n] \\ &+ \operatorname{sum} a[m+1,n] \times \operatorname{cnt} a[1,m] \\ &- \operatorname{sum} a[1,m] \times \operatorname{cnt} a[m+1,n]\end{aligned}$$

于是，我们可以通过线段树合并的方式维护出有序数组 a，在合并的时候按如上方式更新出答案即可。时间复杂度为 $O(n \log n)$。

Thanks for Listening!

Contact Me

Mail : yixionggao@outlook.com