20220507 PK 赛题解

20220507 PK 赛题解
- Algorithm
- Chess
- Seq
- Tour
- Graph
- Game

ID	Algorithm	Chess	Seq	Tour	Graph	Gam
Idea	Easy	Easy-Mid	Easy	Easy-Mid	Mid-Hard	Mid-Hard
Coding	Easy	Mid	Easy	Easy-Mid	Easy-Mid	Mid
Summary	Easy	Mid	Easy	Easy-Mid	Mid-Hard	Mid-Hard

Algorithm

对于 10% 的数据，$b_i = 0$

所有算法都有前置知识，所以代价就是 $\sum a_i$

对于另外 20% 的数据，$b_i = i - 1$

所有算法依赖关系形成一条链，所以代价就是 $\max\{a_i\}$

对于另外 20% 的数据，$b_1 = 0$，其他所有 $b_i = 1$

除了 1 号点，所有算法都是直接依赖于 1 号点。

如果存在一个点 $a_i > a_1$，那么第一次操作他，答案就是 $\sum_{i=2}^{n} a_i$

如果 $a_1 > \max_{i=2}^{n} a_i$，那么可以让 1 号点带走一个最大的儿子，答案就是 $\sum a_i - \max_{i=2}^{n} a_i$

对于 80% 的数据，保证只有一个算法 $b_i = 0$

可以发现在此约束条件下，依赖关系是一棵树，每个节点都会被某一个儿子延伸上来的链带走。

设 f_i 表示最后从 i 的子树里延伸上来的链，带走 i 的那条链到现在为止的链上最大值是多少。

那么 i 会选择谁带走自己呢？答案是 f_{son} 最大的那个儿子 son。

考虑方法同上一个部分分：

- 如果 $\max f_{son} > a_i$，那么这个点的代价就可以省略；
- 否则 a_i 的代价不得不选，那么就让他带走一个代价最大的儿子。

所以每次找到 f_{son} 最大的儿子继承过来，其他儿子的代价到此为止。

对于全部数据，$1 \le n, a_i \le 2 * 10^6, 0 \le b_i \le n$

发现依赖关系是一个森林，对于每一棵树都做一遍上述的树形 DP 即可，复杂度 $\mathcal{O}(n)$。

Chess

数据范围 1： $a_i * b_i \leq 25,\ e_i \leq 8$ ，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 $\mathrm{O}(4^{e_i})$ 。

数据范围 2： $a_i * b_i \leq 100,\ e_i \leq 100$ 。

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。

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。

数据范围 3： $a_i * b_i \leq 100,\ e_i \leq 10000$ 。

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。

数据范围 4： $a_i * b_i \leq 100,\ e_i \leq 10^9$ ，只有车 。

答案是 $(n + m - 2)^{e_i}$ ，快速幂即可。

数据范围 5： $a_i * b_i \leq 100,\ e_i \leq 10^9$ 。

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。

Seq

对于前 20% 的数据：$1 \leq n \leq 3, 1 \leq q \leq 10, 1 \leq k, a_i \leq 100$

枚举每个位置的数字，统计合法的答案数。

对于前 40% 的数据：$1 \leq n \leq 100, 1 \leq q \leq 100, 1 \leq k, a_i \leq 100$

对于前 60% 的数据：$1 \leq n \leq 10^5, 1 \leq q \leq 100, 1 \leq k, a_i \leq 10^4$

对于前 80% 的数据：$1 \leq n \leq 10^5, 1 \leq q \leq 100, 1 \leq k, a_i \leq 10^6$

观察到每一位的方案是独立的，处理出来 $1 \ldots k$ 里的数有哪些数可以放，每次对于 $a[i]$ 查询有多少个合法的数字比他小就可以了，方案数就是每一位方案数的乘积。

复杂度 $O(qn\sqrt{k})$ ，根据常数表现和实现方式期望得分 40～80 分。

对于所有测试数据：$1 \leq n \leq 10^5, 1 \leq q \leq 100, 1 \leq k, a_i \leq 10^9$

我们 $O(\sqrt{k})$ 处理出来 k 的所有约数，将他们排序得到序列 s 。

对于某一个 $a[i]$ ，可以放的数是序列 s 的一个前缀里的数，并且随着 $a[i]$ 的变大，这个前缀只会单调变长。

$a[i]$ 的顺序与答案无关，将 $a[i]$ 排序，维护当前可以放的前缀长度，这个长度就是这一位的方案数。

由于排序后，长度的变化是单调的，只会往大移动，均摊复杂度 $O(q(\sqrt{k} + n \log n)))$ 。

Tour

对于 10% 的数据，$f_i = i - 1$

一条链，只需要一次机会就肯定能走完，所以输出 n 个 n 。

对于另外 20% 的数据，$f_i = 1$

第一次可以走根和一个点，之后每次多走一个点，答案是 $2, 3, \ldots, n-1, n, n$

对于 60% 的数据，保证 $n \leq 300$

可以发现每次使用特权一定跳到根最优，因为任何点都可以从根走到。

设 $f[i][j]$ 表示从 i 出发，可以向下走 j 趟，最多能走的点数，是一个裸的树形背包，复杂度 $\mathcal{O}(n^3)$ 。

对于 80% 的数据，保证 $n \leq 10^3$

使用树形背包常见的复杂度优化，每次背包大小枚举到当前的 size ，复杂度就变成了 $\mathcal{O}(n^2)$

```
int dfs(int u) {
    int sz = 1;
    for (auto v : son[u]) {
        int szs = dfs(v); sz += szs;
        for (int j = sz; j >= 1; --j)
            for (int k = 1; k <= szs; ++k)
                if (j >= k) f[u][j] = max(f[u][j], f[u][j - k] + f[v][k]);
    }
    for (int i = 1; i <= sz; ++i) ++f[u][i];
    return sz;
}
```

对于全部数据，$1 \leq n \leq 2 * 10^5, 1 \leq f_i < i$

问题可以转化为，在树上选 k 条从根开始的链，这些链并起来最多有多少个点?

可以通过分类讨论证明，$k = x + 1$ 的方案一定包含 $k = x$ 的方案，也就是说，每次 k 变大，只需要多选一根效果最好的链，贪心是正确的。

因此我们需要动态维护，对于每个点，到根还有多少个点没被选走过，我们称这是这个点的“价值”。

因为每个点只会被删除一次，所以可以对每个点在删除时都维护他的影响：其子树内每个点的“价值” - 1

支持查询全局最大值对应的点，子树 -1 ，可以线段树维护 DFS 序实现（DFS序子树点编号连续）。

复杂度为 $\mathcal{O}(n \log n)$ ，见 std 。

做法还有很多，比如一种做法是维护求解的过程。

考虑每个点什么时候会被带走，答案是子树最深的点会把他带走，其他的到此为止就结束了。

因此可以使用一个优先队列维护决策的过程，每次把最深的点留下，其他点放到备选答案里。

每次选一个最大价值的点加入答案即可，复杂度也是 $\mathcal{O}(n \log n)$

Graph

对于 30% 的数据，$n \leq 50$

假设按照边权从大到小加边，新加入的边连通的两个点是 u, v，那么：

- u, v 原来都已经有边相连，那么当前这条边是废弃的
- u, v 某一个原来已经有边相连，那么当前这条边会把 v 连到 u 所在的连通块
- u, v 都没有边相连，那么新开一个连通块

可以发现，核心的结论是连通块数不会减少（只考虑有连边的点）。

dp 计数，设 `f[i][j][k]` 表示当前从大到小加到第 i 条边，有 j 个点已经有边连接，目前有 k 个连通块的**方案数**。

- u, v 原来都已经有边相连，方案数是 $\binom{j}{2} - i$
- u, v 某一个原来已经有边相连，方案数是 $j * (n - j)$
- u, v 都没有边相连，方案数是 $\binom{n-j}{2}$

这样直接 DP 复杂度是 $\mathcal{O}(n^4)$ 的。

```
inline void work() {
  int n = rd(), K = rd();
  int m = n * (n - 1) / 2;
  memset(f, 0, sizeof(f));
  f[0][0][0] = 1;
  for (int i = 0; i < m; ++i)
    for (int j = 0; j <= n; ++j)
      for (int k = 0; k <= n; ++k)
        if (f[i][j][k]) {
          add(f[i + 1][j][k], 1ll * f[i][j][k] * (C(j, 2) - i + mod) % mod);
          add(f[i + 1][j + 1][k], 1ll * f[i][j][k] * j % mod * (n - j) % mod);
          add(f[i + 1][j + 2][k + 1], 1ll * f[i][j][k] * C(n - j, 2) % mod);
        }
  printf("%lld\n", 1ll * f[m][n][K] * ifac[m] % mod);
}
```

对于 50% 的数据，$n \leq 1000$

其实我们并不关心当前到底是第几条边，只要把所有的点都连完了就结束了。

设 `f[j][k]` 表示加边过程中，有 j 个点已经有边连接，目前有 k 个连通块的**概率**。

那么上述的转移中，第一种转移我们并不关心，后两种按照方案数比例转移就行了，复杂度是 $\mathcal{O}(n^2)$ 的。

```
inline void work() {
    int n = rd(), K = rd();
    int m = n * (n - 1) / 2;
    memset(f, 0, sizeof(f));
    f[0][0] = 1;
    for (int j = 0; j <= n; ++j)
        for (int k = 0; k <= K; ++k)
            if (f[j][k]) {
                int c1 = 1ll * j * (n - j) % mod;
```

```
                int c2 = C(n - j, 2);
                int p1 = 1ll * c1 * fpow(c1 + c2) % mod;
                int p2 = (mod + 1 - p1) % mod;
                add(f[j + 1][k], 1ll * f[j][k] * p1 % mod);
                add(f[j + 2][k + 1], 1ll * f[j][k] * p2 % mod);
            }
    printf("%lld\n", f[n][K]);
}
```

对于 100% 的数据，$n \leq 5 * 10^5$

观察一下每个连通块的结构：一个核心边（块内边权最大的）被点亮了两次，其他每个边都只被点亮一次。

因此我们本质上是考虑有 k 个核心边的概率。

设 S 是一个边集，$f(S)$ 表示图最终的核心边集合是 S 的概率，答案就是 $\sum_{|S|=k} f(S)$。

考虑容斥，设 $g(S)$ 表示图最终核心边集合包含 S 的概率，有 $f(X) = \sum_{Y \supseteq X} (-1)^{|Y|-|X|} g(Y)$，因此：

$$ans = \sum_{|X|=k} f(X) = \sum_{|X|=k} \sum_{Y \supseteq X} (-1)^{|Y|-|X|} g(Y) = \sum_{|Y| \geq k} \binom{|Y|}{k} (-1)^{|Y|-k} g(Y) = \sum_{i \geq k} \binom{i}{k} (-1)^{i-k} \sum_{|X|=i}$$

考虑每次选中 X 中一条边的概率，以及 X 中每条边的优先级，有 $g(X) = i! \prod_{j=1}^{i} \frac{1}{\binom{n}{2} - \binom{n-2j}{2}}$

再考虑 $|X| = i$ 的 X 有多少个：是 $\frac{1}{i!2^i} * \frac{n!}{(n-2i)!}$，所以可以推得：

$$ans = \sum_{i \geq k} \binom{i}{k} (-1)^{i-k} * \frac{1}{i!2^i} * \frac{n!}{(n-2i)!} * i! \prod_{j=1}^{i} \frac{1}{\binom{n}{2} - \binom{n-2j}{2}}$$

就可以 $\mathcal{O}(n)$ 计算了。

此外 $g \to f$ 是卷积，其实可以 $\mathcal{O}(m \log m)$ 求 $k = 1, \ldots, \lfloor \frac{n}{2} \rfloor$ 的答案，估计大家没时间写就没加强。

Game

对于 50% 的数据，暴力搜索是否必胜即可。

首先观察这个地图可以拆成两张，按照 (行号 + 列号) 的奇偶性可以把图分开，互不影响。

进一步，如果把坐标系转 45°，可以发现每次操作就相当于把一个以黑色为边界的矩形横竖各切一刀。

所以其实是把当前的游戏转化成了四个子游戏的并，根据 SG 引理，当前状态的 SG 值就是四个子游戏的 SG 值的异或。

本质不同的游戏数取决于当前"矩形"在原地图中的位置，所以有 $O(R^2 \times C^2)$ 个。

枚举下一个操作的是哪个位置，复杂度 $\mathcal{O}(R \times C)$，所以记忆化搜索 SG 函数总复杂度 $\mathcal{O}(R^3 \times C^3)$。

第一次的选择的时候，是把游戏划分为五种情况的并，除了四个子矩形，还有另外一张图（奇偶性不同）。

实现上时细节比较多，注意常数。

by SGColin