NOI 模拟赛 3 试题讨论

wzj52501

Peking University

2018 年 12 月 10 日

Results

Results

50pts

- 不难发现题目给定的图结构其实是一个置换的若干循环，而每个循环只要分配 1 名传教士即可。预处理所有循环的大小 $Size_i$，考虑使用背包 DP 解决。

50pts

- 不难发现题目给定的图结构其实是一个置换的若干循环，而每个循环只要分配 1 名传教士即可。预处理所有循环的大小 $Size_i$，考虑使用背包 DP 解决。
- 设 $f(i,j)$ 表示前 i 个循环分配了 j 个传教士的方案，不难得出转移 $f(i,j)=\sum_{k=1}^{Size[i]} f(i-1,j-k)\times\binom{Size[i]}{k}$。

50pts

- 不难发现题目给定的图结构其实是一个置换的若干循环，而每个循环只要分配 1 名传教士即可。预处理所有循环的大小 $Size_i$，考虑使用背包 DP 解决。
- 设 $f(i,j)$ 表示前 i 个循环分配了 j 个传教士的方案，不难得出转移 $f(i,j)=\sum_{k=1}^{Size[i]} f(i-1,j-k)\times\binom{Size[i]}{k}$。
- 虽然这个 DP 看起来很像 $O(n^3)$，其实观察到这个式子本质上枚举了前 $i-1$ 个循环的所有元素与第 i 个循环的所有元素的所有配对，而任意两个元素之间最多被配对一次，所以时间复杂度其实是 $O(n^2)$ 的。

50pts

- 不难发现题目给定的图结构其实是一个置换的若干循环，而每个循环只要分配 1 名传教士即可。预处理所有循环的大小 $Size_i$，考虑使用背包 DP 解决。
- 设 $f(i,j)$ 表示前 i 个循环分配了 j 个传教士的方案，不难得出转移 $f(i,j)=\sum_{k=1}^{Size[i]} f(i-1,j-k)\times\binom{Size[i]}{k}$。
- 虽然这个 DP 看起来很像 $O(n^3)$，其实观察到这个式子本质上枚举了前 $i-1$ 个循环的所有元素与第 i 个循环的所有元素的所有配对，而任意两个元素之间最多被配对一次，所以时间复杂度其实是 $O(n^2)$ 的。
- 时间复杂度为 $O(T\times n^2)$。

80pts

- 考虑每个循环的生成函数 $A(x)$，x^i 的系数为该循环分配 i 个传教士的方案数贡献。我们将所有的 $A(x)$ 相乘得到 $G(x)$，则 $G(x)$ 的 x^i 系数即为分配 i 个传教士满足任意一个循环均有至少 1 个传教士的方案数。

80pts

- 考虑每个循环的生成函数 $A(x)$，x^i 的系数为该循环分配 i 个传教士的方案数贡献。我们将所有的 $A(x)$ 相乘得到 $G(x)$，则 $G(x)$ 的 x^i 系数即为分配 i 个传教士满足任意一个循环均有至少 1 个传教士的方案数。
- 经过一些分析可以得到 $A(i) = \sum_{k=1}^{Size[i]} \binom{Size[i]}{k} x^k$。

80pts

- 考虑每个循环的生成函数 $A(x)$，x^i 的系数为该循环分配 i 个传教士的方案数贡献。我们将所有的 $A(x)$ 相乘得到 $G(x)$，则 $G(x)$ 的 x^i 系数即为分配 i 个传教士满足任意一个循环均有至少 1 个传教士的方案数。
- 经过一些分析可以得到 $A(i) = \sum_{k=1}^{Size[i]} \binom{Size[i]}{k} x^k$。
- 考虑不同的 $Size$ 至多有 $O(\sqrt{n})$ 种，我们可以对于不同的 $Size$ 使用 NTT 计算 $A(x)$ 的点值再快速幂合并起来。

80pts

- 考虑每个循环的生成函数 $A(x)$，x^i 的系数为该循环分配 i 个传教士的方案数贡献。我们将所有的 $A(x)$ 相乘得到 $G(x)$，则 $G(x)$ 的 x^i 系数即为分配 i 个传教士满足任意一个循环均有至少 1 个传教士的方案数。
- 经过一些分析可以得到 $A(i) = \sum_{k=1}^{Size[i]} \binom{Size[i]}{k} x^k$。
- 考虑不同的 $Size$ 至多有 $O(\sqrt{n})$ 种，我们可以对于不同的 $Size$ 使用 NTT 计算 $A(x)$ 的点值再快速幂合并起来。
- 时间复杂度为 $O(T \times n^{1.5} \log n)$。

100pts

- 注意到我们在使用 NTT 合并两个大小为 Sa 与 Sb 的循环时，复杂度其实是 $O((Sa+Sb)\log(Sa+Sb))$ 的。

100pts

- 注意到我们在使用 NTT 合并两个大小为 Sa 与 Sb 的循环时，复杂度其实是 $O((Sa+Sb)\log(Sa+Sb))$ 的。
- 这与哈夫曼编码的过程其实很相似，我们使用哈夫曼编码的顺序来合并，可以保证合并的复杂度最优。

100pts

- 注意到我们在使用 NTT 合并两个大小为 Sa 与 Sb 的循环时，复杂度其实是 $O((Sa+Sb)\log(Sa+Sb))$ 的。
- 这与哈夫曼编码的过程其实很相似，我们使用哈夫曼编码的顺序来合并，可以保证合并的复杂度最优。
- 另一种的合并方法是每轮将相邻的两个多项式合并，这样每轮的复杂度均为 $O(\log n)$，而至多进行 $O(\log n)$ 轮（因为每轮多项式数量减半）

100pts

- 注意到我们在使用 NTT 合并两个大小为 Sa 与 Sb 的循环时，复杂度其实是 $O((Sa+Sb)\log(Sa+Sb))$ 的。
- 这与哈夫曼编码的过程其实很相似，我们使用哈夫曼编码的顺序来合并，可以保证合并的复杂度最优。
- 另一种的合并方法是每轮将相邻的两个多项式合并，这样每轮的复杂度均为 $O(\log n)$，而至多进行 $O(\log n)$ 轮（因为每轮多项式数量减半）
- 时间复杂度为 $O(T\times n\log^2 n)$。

Results

30pts

- 对于列数较小的情况，这是一个经典的轮廓线 DP 问题。

30pts

- 对于列数较小的情况，这是一个经典的轮廓线 DP 问题。
- 时间复杂度为 $O(n \times m \times 2^m)$。

50 - 65pts

- 如此大的数据范围让我们很自然联想到了矩阵快速幂。

50 - 65pts

- 如此大的数据范围让我们很自然联想到了矩阵快速幂。
- 设 $f(i, S)$ 表示前 $i-1$ 列已经填满，第 i 列的状态为 S 的方案数。
- 预处理 $f(i, S)$ 到 $f(i+1, S2)$ 的转移矩阵，然后快速幂优化（需要写十进制快速幂）

50 - 65pts

- 如此大的数据范围让我们很自然联想到了矩阵快速幂。
- 设 $f(i, S)$ 表示前 $i-1$ 列已经填满，第 i 列的状态为 S 的方案数。
- 预处理 $f(i, S)$ 到 $f(i+1, S2)$ 的转移矩阵，然后快速幂优化（需要写十进制快速幂）
- 时间复杂度为 $O(6^m + 8^m \log r)$。

80 - 90pts

- 实际上有用的状态数不足 2^m，把从初始状态不可能到达与不可能从此状态出发到达最终状态的状态去除，当 $m = 8$ 时仅有 71 个有用状态。

80 - 90pts

- 实际上有用的状态数不足 2^m，把从初始状态不可能到达与不可能从此状态出发到达最终状态的状态去除，当 $m = 8$ 时仅有 71 个有用状态。
- 沿用上一个算法。
- 时间复杂度为 $O(6^m + k^3 \log r)$，$k \leq 71$。

100pts

- 矩阵乘法本身的复杂度很难得到优化（使用分治矩阵乘法优化并不明显），我们考虑一个 $k \times k$ 的矩阵 A 的特征多项式 $f(x) = |A - \lambda I|$。

100pts

- 矩阵乘法本身的复杂度很难得到优化（使用分治矩阵乘法优化并不明显），我们考虑一个 $k \times k$ 的矩阵 A 的特征多项式 $f(x) = |A - \lambda I|$。
- 根据 Cayley-Hamilton 定理有 $f(A) = 0$，故有 $A^r \equiv (A^r \mod f(A))$。

100pts

- 矩阵乘法本身的复杂度很难得到优化（使用分治矩阵乘法优化并不明显），我们考虑一个 $k \times k$ 的矩阵 A 的特征多项式 $f(x) = |A - \lambda I|$。
- 根据 Cayley-Hamilton 定理有 $f(A) = 0$，故有 $A^r \equiv (A^r \mod f(A))$。
- 这样只需要求出 $f(A)$ 后计算 $A^r \mod f(x)$（暴力 $O(k^2 \log r)$ 或者多项式除法优化 $O(k \log k \log r)$）就可以在 $O(k^4)$ 的时间内求出来 A^k。

100pts

- 矩阵乘法本身的复杂度很难得到优化（使用分治矩阵乘法优化并不明显），我们考虑一个 $k \times k$ 的矩阵 A 的特征多项式 $f(x) = |A - \lambda I|$。
- 根据 Cayley-Hamilton 定理有 $f(A) = 0$，故有 $A^r \equiv (A^r \mod f(A))$。
- 这样只需要求出 $f(A)$ 后计算 $A^r \mod f(x)$（暴力 $O(k^2 \log r)$ 或者多项式除法优化 $O(k \log k \log r)$）就可以在 $O(k^4)$ 的时间内求出来 A^k。
- 考虑到直接求元素是多项式的矩阵的行列式很麻烦，我们可以先随便代入 $k+1$ 个不同的求出的 $k+1$ 个点值，最后通过拉格朗日插值还原出 $f(x)$。

100pts

- 矩阵乘法本身的复杂度很难得到优化（使用分治矩阵乘法优化并不明显），我们考虑一个 $k \times k$ 的矩阵 A 的特征多项式 $f(x) = |A - \lambda I|$。
- 根据 Cayley-Hamilton 定理有 $f(A) = 0$，故有 $A^r \equiv (A^r \mod f(A))$。
- 这样只需要求出 $f(A)$ 后计算 $A^r \mod f(x)$（暴力 $O(k^2 \log r)$ 或者多项式除法优化 $O(k \log k \log r)$）就可以在 $O(k^4)$ 的时间内求出来 A^k。
- 考虑到直接求元素是多项式的矩阵的行列式很麻烦，我们可以先随便代入 $k+1$ 个不同的求出的 $k+1$ 个点值，最后通过拉格朗日插值还原出 $f(x)$。
- 时间复杂度为 $O(6^m + k^2 \log r + k^4)$，$k \leq 71$。

Results

30pts

- 对于每个询问，把路径上所有点的信息取出，问题转化为数轴上有若干个点，每个位置有 A_i 个点，你要在其中选择一个位置作为集合点，让所有点到该位置集合的距离之和最小。

30pts

- 对于每个询问，把路径上所有点的信息取出，问题转化为数轴上有若干个点，每个位置有 A_i 个点，你要在其中选择一个位置作为集合点，让所有点到该位置集合的距离之和最小。
- 显然应该取这若干个点的中位数位置。

30pts

- 对于每个询问，把路径上所有点的信息取出，问题转化为数轴上有若干个点，每个位置有 A_i 个点，你要在其中选择一个位置作为集合点，让所有点到该位置集合的距离之和最小。
- 显然应该取这若干个点的中位数位置。
- 时间复杂度为 $O(nq)$。

100pts

- 考虑用数据结构优化这一过程，我们需要解决的问题有 3 个：

100pts

- 考虑用数据结构优化这一过程，我们需要解决的问题有 3 个：
- 更改一个节点的权值。
- 查询一段路径 $u \to v$ 上带权的中位数节点。
- 查询一段路径 $u \to v$ 上所有点到某一点 x 带权距离之和。

100pts

- 考虑用数据结构优化这一过程，我们需要解决的问题有 3 个：
- 更改一个节点的权值。
- 查询一段路径 $u \to v$ 上带权的中位数节点。
- 查询一段路径 $u \to v$ 上所有点到某一点 x 带权距离之和。
- 对于问题 1、2，我们可以考虑在 ST 表上倍增，使用另一个数据结构带修改维护路径权值和就可以在 $O(n \log n) - O(\log^2 n)$ 的时间内解决。也可以使用 LCT 维护树结构，在 splay 树上查找答案，这样的复杂度是 $O(n) - O(\log n)$。

cont'd

- 对于问题 1、3，把路径拆成 3 段：$u \to x$，$x \to lca$，$lca \to v$，每一段都可以使用 DFS 序 + 树状数组或树链剖分 + 线段树在 $O(n \log n) - O(\log n)$ 或 $O(n \log n) - O(\log^2 n)$ 的时间内解决。

cont'd

- 对于问题 1、3，把路径拆成 3 段：$u \to x$，$x \to lca$，$lca \to v$，每一段都可以使用 DFS 序 + 树状数组或树链剖分 + 线段树在 $O(n \log n) - O(\log n)$ 或 $O(n \log n) - O(\log^2 n)$ 的时间内解决。
- 时间复杂度：$O(n \log n) - O(\log^2 n)$ 或 $O(n \log n) - O(\log n)$。

Q & A