《逛公园》命题报告

福建省福州第一中学 林昊翰

摘 要

《逛公园》是我为 IOI2021 候选队员互测活动命制的一道试题，是一道“广义串并联图”相关的问题。本文先介绍了以往的通过收缩操作和建立串并联树进行动态规划的方法，但该方法扩展性较差，仅能用于规划类问题，无法完全解决这道题目。接下来引入了内部图和外部图的概念，说明了“内部图”的性质与树上问题时的“子树”有很大的相似性，并利用该性质在广义串并联图上进行类似边分治的算法来解决这个问题。我希望《逛公园》这道题能够带给大家更多关于广义串并联图的相关思考。

1 试题

1.1 题目描述

我们称满足对于任意 4 个节点都不存在 6 条两两除公共端点外没有公共点的路径连接这 4 个节点中的每一对节点的无向连通图为广义串并联图。

给定一张 n 阶简单（无自环无重边）广义串并联图 $G=(V,E)$，每条边有长度 l_i。有 q 次询问，每种询问形如一下两种之一：

1、给定一个点集 S，求 S 中每一对不同的无序点对 (S_i,S_j) 的最短路径长度和。

2、给定一个点集 S 和参数 v，求 S 中有多少对不同的无序点对 (S_i,S_j) 满足其最短路径长度不超过 v。

1.2 输入格式

第一行两个正整数 n,m，其中 n 表示图 G 的节点数，m 表示图 G 的边数。

接下来 m 行，每行三个正整数，其中第 i 行为 x_i,y_i,l_i，表示第 i 条道路的连接的两个点编号为 x_i,y_i，长度为 l_i。

第 $m+2$ 行一个非负整数 q。

接下来 q 组询问，每组形如以下两种之一：

1、为上述第一种询问。第一行两个正整数 $1\ k$，其中 k 表示选择的点集 S 的大小。接下来一行 k 个整数 $S_1, S_2, \cdots, S_k$，表示选择的点集 S。

2、为上述第二种询问。第一行三个正整数 $2\ k\ v$，其中 k, v 分别表示选择的点集 S 的大小和给定的参数。接下来一行 k 个整数 $S_1, S_2, \cdots, S_k$，表示选择的点集 S。

1.3 输出格式

输出 q 行，其中第 i 行表示第 i 个询问的答案。

1.4 样例输入

```
2 1
1 2 5
2
1 2
1 2
2 2 6
1 2
```

1.5 样例输出

```
5
1
```

1.6 样例解释

$1, 2$ 之间的边的长度 $l = 5$。

第一个询问相当于求 $1, 2$ 之间的最短路径长度。答案为 5。

第二个询问相当于求 $1, 2$ 之间的最短路径长度是否 ≤ 6，是则输出 1，否则输出 0。答案为 1。

1.7 数据范围

设 K 为所有询问给定的点集 S 的大小之和。

保证 $2 \leq n \leq 5 \times 10^5$，$1 \leq q, K \leq 5 \times 10^5$。

保证 $1 \leq l_i \leq 10^9$，$1 \leq v \leq 10^{18}$。

子任务 1（5 分）：$n, m, K \leq 3000$；

子任务 2（5 分）：$q = 1$，仅有第一种询问，给定的图为树；
子任务 3（20 分）：给定的图为树；
子任务 4（5 分）：$q = 1$，$k_i = n$，仅有第一种询问，给定的图为仙人掌；
子任务 5（20 分）：给定的图为仙人掌；
子任务 6（25 分）：$k_i = 2$；
子任务 7（20 分）：无特殊限制。

1.8 时空限制

时间限制：5s
空间限制：1GB

2 一些约定

对于图 G，用 $dis(x,y)$ 表示点 x 与点 y 之间的最短路径长度。
用中括号包起来的逻辑表达式 $[x]$ 的取值为：若 x 成立 $[x] = 1$，否则 $[x] = 0$。

3 初步分析

3.1 算法一

对于子任务 1，n, m, K 不超过 3000，可以考虑对于每一对点预处理出它们之间的最短路径长度，对于询问枚举每一对点并直接统计答案。

该算法时间复杂度为 $O(nm \log n + K^2)$。

期望得分 5 分。

该算法效率十分低下，是因为该算法没有利用到 G 是广义串并联图的性质。

接下来先从更特殊的情况入手考虑这个问题。

4 树

4.1 算法二 树，仅有一组的第一种询问

由于树上两点之间的简单路径唯一，可以考虑每一条边被经过了多少次。

对于一条边 i，它将树分成两颗子树，若其中一颗子树有 a_i 个点在询问点集 S 中，另一颗有 $b_i = k - a_i$ 个，那么它就被经过了 $a_i \times b_i$ 次。

可以进行一次树形动态规划求出所有 a_i, b_i，答案即为 $\sum_{i=1}^{m}(a_i \times b_i \times l_i)$。

时间复杂度 $O(n)$。

期望得分 5 分。

4.2 算法三 树，一般情况

此时这个问题是一个比较经典的树上简单路径类问题，有两种常见做法：虚树和点分治。由于虚树做法难以扩展到广义串并联图上，与本文内容关系不大，下面仅介绍点分治做法：

对于重心 u，预处理出 u 到每一个点 x 的最短路长度 $dis(x,u)$，然后离线处理每一个询问：

询问一：对于每个点 x，其贡献为 $dis(x,u) \times (k-c)$ 其中 k 为询问点集大小，c 为删去重心 u 后与 x 在同一连通块的询问点个数。

询问二：对于每个点 x，其贡献为满足 $dis(x,u) + dis(y,u) \le v$ 的，且删去重心 u 后与 x 不在同一连通块 y 的个数。考虑将所有满足条件的 y 减掉与 x 在同一连通块的 y。统计所有满足条件的 y 可以通过按 $dis(x,u)$ 排序后双指针扫描实现，而与 x 在同一连通块的 y 则可以将每个连通块分别按 $dis(x,u)$ 排序并统计。（注意这样统计出来的答案为实际答案的两倍，最终答案要除以 2 ）。

然后删去重心 u，对于剩下的图上的每个连通块分别递归下去。

时间复杂度为 $O((n+K)\log^2 n)$。

期望得分 25 分。

5 仙人掌

对于仙人掌上的问题，一种经典的方法是建立圆方树并转化为类似树上问题。

其中子任务 4 可以通过圆方树上动态规划解决，子任务 5 可以通过圆方树上点分治解决。

该部分内容与本文主题无关，因此不再赘述。相关内容可以参见陈俊锟在 IOI2017 候选队的论文《〈神奇的子图〉命题报告及其拓展》以及他的 UOJ 博客[1]。

[1] http://immortalco.blog.uoj.ac/blog/1955

6 广义串并联图

6.1 重要性质

对于一张图 G，我们可以进行一些操作来缩小图的规模，我们称它们为**收缩操作**。

定义 6.1. 若图中存在点度为 1 的点，直接将其删除，称其为**删 1 度点操作**。

定义 6.2. 若图中存在一对重边，将其合并成为一条，合并成的新边的长度为这一对重边长度的较小值，称其为**叠合重边操作**。

定义 6.3. 若图中存在点度为 2 的点，如果与该点相连的两条边是一对重边，则先进行叠合重边操作；否则设与该点相连的两个点分别为 x,y，删除这个点和与其相连的边，在 x,y 之间建立新边，新边的长度为被删除的两条边的长度之和，称其为**缩 2 度点操作**。

引理 6.1. 以上操作不会影响**未被删除**的点之间的最短路径长度。

证明.

1、对于删 1 度点操作，其它点对的最短路径不会经过 1 度点。

2、对于叠合重边操作，其他点对的最短路径在经过重边时一定选择长度较小的一条。

3、对于缩 2 度点操作，其他点对的最短路径若经过这个 2 度点，那么一定会同时经过与它相连的两条边。 □

定理 6.1. 任意广义串并联图都可以在若干次缩 2 度点、叠合重边、删 1 度点的操作后变为一个只包含一个节点的图。

定理 6.1 的证明十分繁琐，故略去，具体可以参见吴作同在 IOI2019 候选队的论文《〈公园〉命题报告》。

定理 6.2. 任意简单（无重边无自环）广义串并联图满足边数不超过点数的两倍。

证明. 由于每次缩 2 度点和删 1 度点操作都各减少一条边的一个点，叠合重边操作减少一条边，但每次缩 2 度点后至多进行一次叠合重边操作，因此任意无重边的广义串并联图满足边数不超过点数的两倍。 □

6.2 建立串并联树

考虑将上述操作变成树形结构：

初始图中每一个点和每一条边均对应为串并联树的叶子节点，然后在收缩过程中建立这个串并联树：

1、删 1 度点操作：建立一个新的节点 v'，将被删除点 x，与其相连的边 a 和 a 的另一个端点 z 对应的树上节点 v_x, v_a, v_z 的父亲设为 v'，然后将点 z 对应的树上节点设为 v'。2、

叠合重边操作：建立一个新的节点 v'，将两条重边 a,b 对应的树上节点 v_a,v_b 的父亲设为 v'，然后将合并后的边对应的树上节点设为 v'。

3、缩 2 度点操作：建立一个新的节点 v'，将被删除点 x 和与其相连的两条边 a,b 对应的树上节点 v_x,v_a,v_b 的父亲设为 v'，然后将建立的新边对应的树上节点设为 v'。

这个串并联树中，每一个树上节点唯一对应图上的一个点或一条边，初始图 G 中每一个点或一条边唯一对应一个树上叶子节点，且每一个非叶子节点也对应一个收缩操作。

引理 6.2. *对于任意一个串并联树上非叶子节点 u，都可以在不进行其它收缩操作的情况下将所有在点 u 的子树内（包括点 u）的非叶子节点对应的收缩操作执行完毕。*

证明. 考虑反证法：找到任意一个深度最大的不满足条件的点 u。那么它子树内的操作都可以先执行，剩下点 u 对应的操作无法执行。

1、若 u 对应的是叠合重边操作，那么 u 的儿子对应需要被叠合的两条重边，u 子树内的操作执行完毕后这两条重边一定会出现，u 对应的操作一定可以执行，矛盾。

2、若 u 对应的是删 1 度点操作或缩 2 度点操作，如果 u 的儿子执行完毕后如果点 u 对应的操作无法执行，那么此时需要被收缩的点（设其为 x）的点度不符合条件，需要再执行一些其他操作才能满足条件。

能够影响 x 点度的有叠合重边操作和删 1 度点操作，但不在点 u 的子树内的删 1 度点操作会导致点 x 对应的树上节点不在点 u 的子树内，不在点 u 的子树内叠合重边操作会导致与点 x 相连的边对应的树上节点不在点 u 的子树内，与它们是点 u 的儿子矛盾。

由此，**引理 6.2** 得证。 □

6.3 算法五 询问点集大小为 2

对于一个串并联树上节点 u，我们定义它的**内部图** $G'_u(V'_u,E'_u)$ 如下：

定义 6.4. *串并联树上节点 u 的内部图 G'_u 为初始图 G 的一个子图。*

1、如果点 u 在图上对应的是一个点，那么对于初始图 G，如果图上的一个点或一条边对应的树上叶子节点在点 u 的子树中，那么这个点或边属于 G'_u。

2、如果点 u 在图上对应的是一条边，设这条边为 (x,y)，那么对于初始图 G，如果图上的一个点或一条边对应的树上叶子节点在点 u 的子树中，那么这个点或边属于 G'_u，且点 x 和点 y 也属于 G'_u。

定理 6.3. *对于一个树上节点 u 的内部图 $G'_u(V'_u,E'_u)$：$\forall(x,y)\in E'_u$ 有 $x\in V'_u,y\in V'_u$*

证明. 对于一条在 G'_u 的边 (x,y)，先执行点 u 子树内部的收缩操作，那么边 (x,y) 一定会被收缩掉。根据上述收缩操作，两个端点 x,y 要么是已经被收缩掉的点，要么（如果点 u 对应的是边）是点 u 的端点之一，也就一定属于 V'_u。 □

询问点集大小为 2 时等价于多次询问两个点之间的最短路径长度，设这两个点分别为 f,g 。

考虑在串并联树上进行树形动态规划：

如果 u 对应一个点，那么记该点为点 A ，点 B 不存在。如果 u 对应一条边，那么记与它相邻的两个点分别为点 A 和点 B 。

设计如下的 DP 数组（以下的最短路径长度均为只保留点 u 的内部图后的最短路径长度）：

$fa[u]$：点 f 到点 A 的最短路径长度（若点 f 不在 u 的内部图中，或 u 对应一条边且点 f 为点 u 对应边的端点，该值为 $+\infty$ ）

$ga[u]$：点 g 到点 A 的最短路径长度（细节同上）

$fb[u]$：点 f 到点 B 的最短路径长度（若点 f 不在 u 的内部图中，或 u 对应一条边且点 f 为点 u 对应边的端点，或点 B 不存在，该值为 $+\infty$ ）

$gb[u]$：点 g 到点 B 的最短路径长度（细节同上）

$fg[u]$：点 f 到点 g 的最短路径（若不满足上述任一条件，该值均为 $+\infty$ ）

然后就能在串并联树上进行树形动态规划来解决一组询问（具体转移较为繁琐且与本文内容无关，在此不详细列出）。

对于多组询问的处理，考虑分析上面动态规划的转移形式，发现它可以写成类似矩阵的形式，只是将加法换成取 max ，乘法换成加法。

由于加法，max 运算满足结合律，且加法对 max 运算满足分配率，所以这种新定义的矩阵依旧满足结合律[5]。

对于一组询问 f,g ，我们仅需要得到 f,g 对应串并联树上节点到它们的 LCA 的转移矩阵乘积和 LCA 到根的转移矩阵乘积，采用树上倍增算法即可解决。

该算法复杂度为 $O((n+q)\log n)$ ，可以通过子任务 6，期望得分 25 分。

6.4 算法六 一般情况

由于串并联树是一个三叉树，可以直接进行边分治，且边分治仅会将树分成两个部分，比起会将树分成多个部分的点分治来说需要考虑的情况要简单得多，因此这里采用边分治而非点分治。

边分治时每次找到一条边并将树按照这条边分开。

考虑这条边的两个端点，它们一定是儿子-父亲关系，设其中的儿子为点 u ，那么就变成了每次考虑 u 的子树内到 u 的子树外的答案。

再定义**外部图** 为：

定义 6.5. *一个串并联树上节点 u 的**外部图**为先进行所有在点 u 子树内的收缩操作后的图。*

那么有：

定理 6.4. 对于一个串并联树上节点 u，如果它对应一个点 x，那么其内部图和外部图点集的交恰为 x，如果它对应一条边 (x,y)，那么其内部图和外部图点集的交恰为 x,y。

证明.

1、如果 u 对应一个点 x，那么点 u 对应到一个缩一度点操作，点 u 子树中对应的非 x 节点都已经被缩掉，属于内部图，而其它没被缩掉的点属于外部图。

2、如果 u 对应一条边 (x,y)，那么点 u 子树中对应的节点都已经被缩掉，属于内部图，而其它没被缩掉的点属于外部图，且内部图额外加上了点 x 和点 y。 □

对于点 u 离线处理每一个询问：

我们称点 u 内部图和外部图点集的交为它的**分割点**，它有一到两个分割点，记为 p_1,p_2（p_2 可能不存在）。

首先 $O(m\log n)$ 预处理出分割点到图中所有点的最短路径长度，如果该询问点集中包含分割点，暴力统计其答案并删除，这样询问中的点要么属于内部图，要么属于外部图。

然后统计所有询问中所有满足 x 属于内部图且 y 属于外部图的点对 (x,y) 的贡献：

由**定理 6.4**，它们之间的路径至少会经过 p_1,p_2 其中之一，分类讨论，得出其最短路径长度为：

$dis(x,y)=\min\{dis(x,p_1)+dis(y,p_1),dis(x,p_2)+dis(y,p_2)\}$

（如果 p_2 不存在，那么 $dis(x,p_2)+dis(y,p_2)=+\infty$ ）

这个 min 取到哪一边和 $(dis(x,p_1)-dis(x,p_2))+(dis(y,p_1)-dis(y,p_2))$ 的正负性有关，因此可以将属于外部图的点 y 分别按照 $dis(y,p_1)+dis(y,p_2)$ 排序。

对于属于内部图的点 x。如果 $(dis(y,p_1)-dis(y,p_2))\geq -(dis(x,p_1)-dis(x,p_2))$，那么就有 $dis(x,y)=dis(x,p_1)+dis(y,p_1)$。这样就只需要知道所有满足 $(dis(y,p_1)-dis(y,p_2))\geq -(dis(x,p_1)-dis(x,p_2))$ 的相关信息和，就能算出点 x 的贡献。具体地：

1、对于第一种询问，点 y 的个数及其 $dis(y,p_1)$ 的和。

2、对于第二种询问，满足 $dis(y,p_1)\leq v-dis(x,p_1)$ 的点 y 的个数。

这两者都是能比较简单地通过前缀和和双指针扫描直接计算的。

$(dis(y,p_1)-dis(y,p_2))<-(dis(x,p_1)-dis(x,p_2))$ 的情况同理。

至此边分治的一步就完成了：我们解决了所有从内部图的点到外部图的点的贡献，也即 u 的子树内到 u 的子树外的贡献。

接下来我们就把剩下询问分成了两个部分：两者均属于外部图的点对之间的贡献和两者均属于内部图的点对之间的贡献。

由**引理 6.1**，外部图之间点对的最短路径长度与原图一致。

但对于内部图，两个点之间的最短路径是有可能经过不属于内部图的点的。

由**定理 6.4**，此时这条路径不属于内部图的部分一定是从 p_1 走到 p_2 或从 p_2 走到 p_1（且 p_2 一定存在），那么其长度也就一定是它们之间的最短路径 $dis(p_1,p_2)$。

只要在内部图中加上连接 p_1 和 p_2，长度为 $dis(p_1,p_2)$ 的无向边即可（如果 p_2 不存在就什么也不干）。

然后我们就可以将整张图拆分成内部图和外部图两张相互独立的图，且它们内部的点对之间的最短路径长度不变，这样原问题也就被分成了在这两个图上的子问题。

由定义可知，外部图对应的串并联树为原串并联树删除 u 的子树（不包括 u ）后的树，内部图对应的串并联树为 u 的子树（包括 u ）再加上点 u 和新加的边叠合重边。

设原图的串并联树结点个数为 N ，那么新图结点个数就分别为 $x+1$ 和 $N-x+2$ ($\frac{N}{4} \le x \le \frac{3N}{4}$) 。对于 $N \le 10$ 的图可以暴力处理所有询问。

该算法时间复杂度为 $O((n+K)\log^2 n)$ ，期望得分 100 分。

7 扩展

7.1 点/边分治的一个扩展

由上述的算法五我们可以得到一个点/边分治算法的推广：

如果能够把一张点数为 N 的图分成若干个子图，每个子图点数不超过 $aN+c$ 满足 a,c 为固定常数且 $0<a<1$ 。每一个点和每条边都至少属于其中的一个个子图，属于多个子图的点数量不超过某个较小的常数 k 。

那么我们可以把所有属于多个子图的点提取出来暴力处理，计算所有包含了这些点的路径的贡献，然后将这些点删除。每一个连通块往下分治，即可做到类似点/边分治的效果。

事实上，传统的点分治就是上述算法的 $k=1$ 时的情况。

例 1. *Giant Penguin*[2]

给定一张连通无向图 $G=(V,E)$ ，每条边长度均为 1 ，这张图满足每个点至多属于 k 个简单环。你需要依次执行 q 个操作，每种操作为以下两者之一:

1、给定 v ，标记顶点 v ，保证点 v 之前没有被标记。

2、给定 v ，输出所有被标记的点到点 v 的最短路径长度的最小值，保证此时至少存在一个点被标记。

$|V| \le 10^5, |E| \le 2\times 10^5, q \le 2\times 10^5, k \le 10$ 。

这题有一个重要性质：这张图满足每个点至多属于 k 个简单环，因此只需要任取这张图的一个生成树，在这个生成树上进行点分治。

对于重心 u ，将它作为生成树的根，它就将这颗树分成了若干个子树。考虑所有不在生成树上的边，如果它连接了 u 的两个不同的子树，那么这条边提供了一个包含重心 u 的简单环，因此这种边不超过 k 条。

这 k 条边端点的点集并不超过 $2k$ ，加上重心 u 就是不超过 $2k+1$ 个点，称这些点构成的点集为 S ，可以将 S 中的点一并删除，然后每个连通块往下递归。

[2]题目来源：300iq Contest 3

对于具体的操作，可以在提取出 S 后对每个 $x \in S, y \in V$ 求出 $dis(x,y)$，并建出点分治树，每个点分治树上结点 z 均有一个 S 集合 S_z，并记 f_i，初值均为 $+\infty$。

然后对于每一个具体操作：

1、对于 1 操作，找到满足的 $v \in S_z$ 点 z，依次考虑点 z 以及其在点分治树上的祖先。当我们考虑点 z' 时，对于每一个 $x \in S_{z'}$，更新 $f_x = \min\{f_x, dis(v,x)\}$。

2、对于 2 操作，记本次操作的答案为 ans，初值为 0。找到满足的 $v \in S_z$ 点 z，依次考虑点 z 以及其在点分治树上的祖先。当我们考虑点 z' 时，对于每一个 $x \in S_{z'}$，更新 $ans = \min\{ans, f_x + dis(v,x)\}$。

即可解决该问题，时间复杂度为 $O(k(|V|+q)\log|V|)$。

7.2 KDtree 上启发式合并维护动态规划

对于广义串并联图上的动态规划问题，常常转化为串并联树上动态规划。

由内部图和外部图的性质，广义串并联图上的动态规划常常需要维护两个点上的状态信息，也就是串并联树上点 u 上的动态规划状态常常是二维的，形如 $f[u][i][j]$。

而处理树上动态规划时有一个很好用的工具：线段树合并，我们希望它能被扩展到二维情况。

例 2. 树上偏序[3]

给定一棵 n 个结点编号为 $1,2,\cdots,n$ 的有根树 T 以及 m 个三元组 (a,x,y)，其中 a,x,y 均为正整数。称一个三元组 (a,x,y) 在结点 x 的子树中当且仅当树 T 中编号为 a_i 的结点在结点 x 的子树中。

求有多少个不同的非空三元组集合 S，满足对于每一个结点 x，要么不存在三元组 $(a,x,y) \in S$ 在 x 的子树中；要么存在一个三元组 $(a',x',y') \in S$ 在 x 的子树中且对于任意在 x 的子树中的三元组 $(a,x,y) \in S$，都有 $x \le x', y \le y'$，答案对 998244353 取模。

$n, m \le 50000$。

7.2.1 朴素算法

设计动态规划，记 $f[i][j]$ 为在 i 的子树中且结点 i 选取的三元组 (a',x',y') 为第 j 个三元组的方案数（若子树中不存在三元组则 $j=0$）。

合并两个点 u,v 的状态 $f[u][i], f[v][j]$ 到点 x 时：

1、如果 $x_i \ge x_j, y_i \ge y_j$ 或 $j=0$，那么转移: $f[x][i] += f[u][i] * f[v][j]$。

2、否则如果 $x_i \le x_j, y_i \le y_j$ 或 $i=0$，那么转移: $f[x][j] += f[u][i] * f[v][j]$。

3、若以上条件均不满足，那么这一对状态没有贡献。

对于一个结点 u 的状态，先合并它所有子树的状态作为 $f[x][j]$。

[3]题目来源：原创

然后对于所有三元组 (a_i, x_i, y_i) 满足 $a_i = u$，相当于点 x 的状态再合并上

$$f[n+i][j] = \begin{cases} 1; i \in 0, j \\ 0; otherwise \end{cases}$$

该算法时间复杂度为 $O(m^2)$，效率低下。

7.2.2 KDtree 上启发式合并

维护二维情况下的矩形信息一般使用 KDtree，于是考虑使用 KDtree 合并。二维上的合并十分复杂，可以使用启发式合并。

N 个点的静态 KDtree 单次查询复杂度为 $O(\sqrt{N})$，插入可以考虑采用分块重建：插入后每次查询都暴力求新插入的点的贡献，如果插入的点数超过 $\sqrt{N}$，那么 $O(N)$ 暴力重构整个 KDtree，这样单次插入及询问复杂度均为均摊 $O(\sqrt{N})$。

启发式合并时每个点每次被合并其所在 KDtree 大小都会翻倍，设总点数为 n，那么对于一个点，其被合并的代价不超过 $\sqrt{n} + \sqrt{\frac{n}{2}} + \sqrt{\frac{n}{4}} + \cdots$，这是 $O(\sqrt{n})$ 级别的，那么总复杂度为 $O(n\sqrt{n})$。

此时合并两个点 u, v 的状态 $f[u][i], f[v][j]$ 到点 x 时：

1、我们对于每一个 $f[u][i]$，设向量

$$g[u][i] = \begin{bmatrix} f[u][i] & 0 \end{bmatrix}$$

2、接下来对于每一个 $f[v][j]$，对于所有位于 (x_j, y_j) 右下角的 (x_i, y_i)，$g[u][i]$ 都右乘上矩阵

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

3、然后对于每一个 $f[v][j]$，求出所有位于 (x_j, y_j) 左上角的 (x_i, y_i) 的 $f[u][i]$ 的和 x'（也就是左上角 $g[u][i]$ 和 $[x' \ y']$ 中 x' 的值），然后在 KDtree 中加入

$$g[u][j] = \begin{bmatrix} 0 & x' \times f[v][j] \end{bmatrix}$$

4、转移所有完毕后将所有 $g[x][i]$ 右乘上矩阵

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

将其变回 $[f[x][i] \ 0]$ 的形式。

上述操作均可以在点数为 N 的 KDtree 中单次均摊 $O(\sqrt{N})$ 实现，因此总时间复杂度为 $O(m\sqrt{m})$，可以通过该例题。

7.3 带源汇串并联图计数

广义串并联图存在一个子集：**串并联图**：

定义 7.1. 如果一张图 G 可以仅通过缩 2 度点操作和叠合重边操作将 G 变成仅有两个点，且这两个点之间有且仅有一条边连接的图，则图 G 为**串并联图**。

引理 7.1. 一个点双连通的广义串并联图为串并联图。

证明. 考虑反证法。若一个广义串并联图不是串并联图，那么它的收缩操作中存在删 1 度点操作，且删除后点的数量不为 1 。设这个收缩操作对应的树上结点为 u ，那么 u 的内部图与外部图的点集的交仅包含一个点，这个点为割点，与原图点双连通矛盾。 □

引理 7.2. 任意广义串并联图的任意串并联树上结点 u 的内部图均连通。

证明. 考虑进行归纳，首先叶子结点的内部图显然连通。对于一个树上结点 u ，假设其儿子的内部图均连通，那么点 u 的内部图点集为儿子内部图点集的并。点 u 的儿子中至少有一个对应的是边，且对应边的儿子的内部图点集与其它儿子内部图点集均有交。因此点 u 的内部图连通。 □

我们定义串并联图的**源汇**为：

定义 7.2. 对于一张串并联图 G ，如果 G 可以仅通过缩 2 度点操作和叠合重边操作将 G 变成仅有两个点 S 和 T ，且这 S,T 之间有且仅有一条边连接的图，则无序点对 (S,T) 为串并联图 G 的一组**源汇**。

引理 7.3. 对于一个串并联图 G 和一组源汇 (S,T) ，如果图 G 的进行收缩操作的最后一步为缩 2 度点操作，那么图 G 存在一个点 $x \notin S,T$ 使得若删除点 x ，S,T 不连通；否则无论删除图中任意一个点 $x \notin S,T$ ，S,T 均连通。

证明. 如果图 G 的进行收缩操作的最后一步为缩 2 度点操作，那么如果删除被它缩掉的点（显然不为 S 或 T ），S,T 就不连通。否则图 G 的进行收缩操作的最后一步为叠合重边操作，那么设根结点的两个儿子（也是被叠合的一对重边对应的点）为 u,v ，如果删除任意一个点 $x \notin S,T$ ，如果 x 在 u 的内部图中，S,T 就可以通过 v 的内部图连通，反之同理。 □

例 3. 带源汇串并联图计数[4]

有 n 个点的简单图（无重边无自环）$G=(V,E)$ ，编号为 $1,2,\cdots,n$ ，和图 G 的一组源汇 (S,T) 组成的三元组 (G,S,T) 。求共有多少个不同的三元组，对 998244353 取模。

两个三元组 (G,S,T) 和 (G',S',T') 不同当且仅当无序点对 (S,T) 与 (S',T') 不同，或图 G 中存在一条边 (x,y) 而图 G' 中不存在，或图 G' 中存在一条边 (x,y) 而图 G 中不存在。

$2 \le n \le 10^5$

[4]题目来源：原创

由**引理 7.3** 可得，不存在三元组 (G,S,T) 使得图 G 的进行收缩操作的最后一步既可以为缩 2 度点操作，也可以为叠合重边操作。

设 f_i 为 $i+2$ 个点（即不包括源汇 i 个点），最后一步为缩 2 度点操作，源汇为 $(1,2)$ 的三元组数 $(G,1,2)$，g_i 为 $i+2$ 个点，最后一步为叠合重边操作的三元组数 $(G,1,2)$。

特别地，我们设 $f_0=0,g_0=1$，也就是说仅有两个点，且这两个点之间有且仅有一条边连接的图归属于 g。

记它们的指数型生成函数：

$$F(x)=\sum_{i=0}^{+\infty}\frac{f_ix^i}{i!},G(x)=\sum_{i=0}^{+\infty}\frac{g_ix^i}{i!}$$

对于转移，考虑将这张图“不断展开”，形式化地说：

1、对于 f_i，其最后一步为缩 2 度点操作。首先对最后缩成的边 x 执行“不断展开”，将它“展开”成收缩前的样子，它会展开成两条新的边的一个新的点，设两条新的边分别为 x_1,x_2。若 x_1 也是由缩 2 度点操作产生的，那么对它继续执行“不断展开”操作，递归下去；否则将 x_1 原样保留。对 x_2 的操作同理。这样“不断展开”后原图就变成了 $j+1$ 个点和 j 条边串成的链（j 为任意大于 1 的正整数），其中每一条边都是初始就存在的或是由叠合重边操作产生的。

2、对于 g_i，其最后一步为叠合重边操作。也是类似操作：首先对最后缩成的边 x 执行“不断展开”，将它“展开”成收缩前的样子，它会展开成一对新的重边，设它们分别为 x_1,x_2。若 x_1 也是由叠合重边操作产生的，那么对它继续执行“不断展开”操作，递归下去；否则将 x_1 原样保留。对 x_2 的操作同理。这样“不断展开”后原图就变成了 2 个点和 j 条重边形成的图，其中每一条边都是初始就存在的或是由缩 2 度点操作产生的。

继续分析不断展开后这 j 条边的内部图，这些内部图都是串并联图，更具体地：

若 f_i 展开成了 $j+1$ 个点和 j 条边，除源汇外有 $j-1$ 个点。其中每一条边都是初始就存在的或是由叠合重边操作产生的，也就是说其中每一条边的内部图的源汇都已经固定为这条边的端点，其他部分对应一个 g_x，且这些内部图是有序拼接的。更具体地，有：

$$\begin{aligned}F(x)&=\sum_{j=2}^{+\infty}x^{j-1}G^j(x)\\&=\frac{xG^2(x)}{1-xG(x)}\end{aligned}$$

若 g_i 展开成了 2 个点和 j 条边，这这两个点分别为 a,b。其中每一条边都是初始就存在的或是由叠合重边操作产生的，也就是说其中每一条边的内部图的源汇都为这 2 个点，其他部分对应一个 f_x，但有可能有一条边对应初始就存在的边 (a,b)。这些内部图是无序拼接的。更具体地，有：

$$\begin{aligned} G(x) &= 1 + \sum_{j=1}^{+\infty} \frac{F(x)^j}{j!} + \sum_{j=2}^{+\infty} \frac{F(x)^j}{j!} \\ &= \sum_{j=0}^{+\infty} \frac{2F(x)^j}{j!} - F(x) - 1 \\ &= 2e^{F(x)} - F(x) - 1 \end{aligned}$$

利用上述两个式子进行多项式牛顿迭代即可得出 $F(x), G(x)$ 的前 n 项，而最终答案即为 $\binom{n}{2} \times (f_{n-2} + g_{n-2}) = \frac{n!}{2} \times ([x^{n-2}]F(x) + [x^{n-2}]G(x))$ 。

时间复杂度 $O(n \log n)$ 。

8 命题思路

吴作同在 IOI2019 的候选队论文《〈公园〉命题报告》中引入了“广义串并联图”的概念，他给出的一种处理方法是通过收缩操作不断缩小问题规模并进行动态规划。同时，为了支持快速修改参数，也提到了建立表达式树[5]来将动态规划转化为树形结构然后链分治的方法。但是这种做法存在一定的局限性：其只能用于处理较为简单的规划类问题。于是我想，广义串并联图是否存在更好的性质。

我观察了收缩操作的逆操作的性质，发现将收缩过程中的一个点或一条边进行逆操作后会展开成一个子图，而缩 2 度点操作就是将两张这种图“串联”起来，叠合重边操作就是将两张这种图“并联”起来。进一步地，每一个串并联树上结点都对应着一个“内部图”，而这个“内部图”与外界有连接的部分至多仅有两个点，这一性质与树上的“子树”的性质十分相似。上述的动态规划本质上是将整个内部图的信息压缩到这两个点上。

这个新的发现不仅能够在设计动态规划时给予更多的启示，让动态规划的状态更加直观，还能使得对于广义串并联图的分析不再局限于规划类问题。

接着我分析了一些树上和仙人掌上的经典问题，考虑它们是否能通过这个性质扩展到广义串并联图上，发现多源最短路和点分治类问题能够很好地被这个新的发现解决，点分治类问题是强于多源最短路的，于是我将点分治作为最终的解法，将多源最短路作为子任务 6 ，命制出了《逛公园》。

在子任务设置方面，子任务 1,2,4 是常见的朴素算法，而子任务 3,5 则给选手提供一个解决问题的关键方向 ———— 点分治。子任务 6 则是弱于正解的多源最短路问题，也为选手提供另一个方向 ———— 收缩操作和动态规划。最终结合两个算法及内部图和外部图的相关性质即可得到子任务 7 的正解，获得满分。

9 致谢

感谢中国计算机学会提供学习和交流的平台。

[5]即上文的“串并联树”

感谢福州一中的陈颖老师给予的关心和指导。
感谢国家集训队教练高闻远提供的指导与帮助。
感谢福州一中的各位学长带给我的启发和指导。
感谢陈俊锟学长，吴作同学长为本文提供思路。
感谢福州一中的各位学长带给我的启发和指导。
感谢福州一中信息组的同学们为本文审稿。
感谢父母的养育之恩。

参考文献

[1] 吴作同，《〈公园〉命题报告》, IOI2019 中国国家候选队论文

[2] 陈俊锟, 《〈神奇的子图〉命题报告及其拓展》, IOI2017 中国国家候选队论文

[3] Wikipedia, Series-parallel graph

[4] Wikipedia, Biconnected component

[5] 机械工业出版社, 《线性代数》