4 Bulbasaur

4.1 题目来源

Petrozavodsk Summer 2019. Day 5. Radewoosh + mnbvmar Contest Problem B[4]

4.2 题目大意

有 $n \times k$ 个点排成 n 列，每列 k 个点，行和列都从 1 开始编号。对于所有 $i \in [1, n)$，给出所有从第 i 列到第 $i+1$ 列的有向边。没有其他的边。定义 $f(l, r)$ 表示以下问题的答案：

取出第 l 列到第 r 列之间的图。你需要求出最大的 p，使得存在 p 条不相交的长度为 $r-l$ 的简单路径。答案为 p。

你需要求出 $\sum_{i=1}^{n} \sum_{j=i+1}^{n} f(i, j)$。

4.3 数据范围

对于所有数据，满足 $1 \le n \le 4 \times 10^4$，$1 \le k \le 9$。

4.4 解题过程

把每个点拆成一个入点和出点，入点向出点连边。这样图变成 $2n \times k$ 个点，且 f 中的不相交路径的限制可以削弱成边不交（这是因为每个点的入度和出度至少一个为 1），且最终求的答案变为 $\sum_{i=1}^{n} \sum_{j=i+1}^{n} f(2i-1, 2j)$。

接下来的问题和算法全部在新图和新问题上讨论。

求最大的边不交路径集合其实是网络流问题。对于求 $f(1, n \times 2)$ 的问题，使用解决网络流的 FF 算法，每次增广需要走 $O(nk)$ 步，每步需要 $O(k)$ 的时间来遍历出边，需要从第一列的 k 个点各进行一次推流，最终时间复杂度为 $O(nk^3)$，已经十分接近时间限制，需要挤压算法的剩余空间来通过此题。

考虑求 $\sum_{i=2}^{n} f(1, 2i)$。仍然使用 FF 算法，不同的是，不再要求推流推到 $n \times 2$ 的位置，而是每次推到能推到的最远的位置。这样一定能够获得 k 条路径（因为第一列一定能到第二列）。为了求出 $f(1, 2i)$，把所有路径 $2i$ 后的部分截断，并忽略所有没有到达 $2i$ 的路径，剩余的路径就是子图 $[1, 2i]$ 的路径集合。下面证明这一点

把最终求出的 k 条路径进行排序。考虑如果需要计算 $f(1, 2i)$ 并已经求出了整个图的路径集合，所有长度超过 $2i-1$ 的路径都经过了 $2i$ 这一列，把 $2i$ 后的路径截断，求出这些路径的步骤和求子图 $[1, 2i]$ 的路径集合的算法是一样的，而对于所有没有到达 $2i$ 的路径，在求子图 $[1, 2i]$ 的算法中同样不会被考虑。而对于所有 i，上面的推理都正确。于是按照这个算法可以在和求 $f(1, 2n)$ 相同的复杂度中求出所有 $[1, 2i]$ 的路径集合。

这样，可以进行 $O(nk^3)$ 的推流之后 $O(nk)$ 每次枚举路径判断长度来求出 $\sum_{i=2}^{n} f(1, 2i)$。但是此时距离整个问题的答案还差一个求和，直接枚举 i 来求 $\sum_{j=i+1}^{n} f(2i-1, 2j)$ 是 $O(n^2k^3)$ 的，需要更多的优化。下面设目前算法中第 i 长的路径长度为 t_i。

[4]https://qoj.ac/problem/8085

首先考虑优化上面的求答案步骤。由于不需要求出每个 $f(i,j)$，就可以改变求和顺序，最终的答案就是 $\sum_{i=2}^{n}\sum_{j=1}^{k}[t_j\geq 2i-1]=\sum_{i=1}^{k}\lfloor\frac{t_i-1}{2}\rfloor$。

然后优化推流步骤。如果 $t_k\neq 1$，那么所有路径都到达了第 3 列。这样，把前两列删掉之后，从第 3 列开始推流的结果不会变化。当需要求 $\sum_{j=3}^{n}f(3,2j)$ 时，只需要简单地把第一列和第二列删掉即可。反映在 t 上，就是令所有 $t_l\leftarrow t_l-2$。沿用这样的思路来进行优化。

考虑枚举 i 并求 $\sum_{j=i+1}^{n}f(2i-1,2j)$。每次准备从 i 增加至 $i+1$ 时，先删去 $2i-1$ 和 $2i$ 两列。注意这有可能将一些路径完全删掉。如果没有完全删除任何路径，那么只需要维护 t 并直接统计答案即可。否则可以新开辟一些路径。直接从 $2i+1$ 这一列开始进行增广，找到能够到达的最远的位置，并推到那里。像这样推流得到的 $[2i+1,2n]$ 的路径集合是正确的，可以如下归纳地证明这一点。

在求出的 $[2i-1,2n]$ 的路径集合中，考虑改为先对所有长度超过 2 的路径进行推流，再进行长度为 1 的路径的推流。这是因为 FF 算法的推流顺序是任意的，根据证明的需要可以任意交换。现在删去图的 $[2i-1,2i]$ 部分相当于恰好删去了最后若干次推流的过程，那么从 $2i+1$ 开始进行推流正好相当于继续进行 FF 算法，和进行了一个完整的 FF 一样。

进行上面的操作即可求出最终的答案。现在需要分析时间复杂度的问题。

考虑一次增广的过程。从 $2p-1$ 层出发，进行搜索来求出最远能到的位置，设为 $2q$。然后进行一次推流，将路径上的边反向，并计入这条路径。其中搜索和推流都涉及到 $O((q-p)k)$ 个点，每个点有 $O(k)$ 条出边，这样的一次增广消耗了 $O((q-p)k^2)$ 的时间并新增了一条长度为 $2(q-p)$ 的路径。

考虑所有路径的长度和。这个数值从 0 开始，每增加 1 花费 $O(k^2)$ 的时间，i 每增加 1 就需要减 $O(k)$，最终需要变成 k。那么总共需要通过增广让它增加 $O(nk)$，总共需要 $O(nk^3)$ 的时间来进行增广。而其他部分并不占用瓶颈，于是总时间复杂度为 $O(nk^3)$，可以通过。

发现时间复杂度瓶颈在于总共 $O(nk^2)$ 次询问每个点的所有出边：只涉及到 $O(nk^2)$ 个点，却需要遍历出边来找到一条可行的边。由于这个算法每次访问一个点时，只需要求出任意一条可行的出边，那么可以使用一个长度为 $2k$ 的二进制数来存两侧的 $2k$ 条可能的出边是否存在，需要求出任意一条可行边的时候可以使用位运算来加速求出。这样每次对一个点操作是 $O(1)$ 的，于是总时间复杂度为 $O(nk^2)$，可以通过。

4.5 参考文献

无。