浅谈开关点连通性
Dynamic Subgraph Connectivity

成都七中　蒋明润　党星宇

目　录

1 简介

简介

动态图连通性是动态图系列的经典问题。对于动态的边修改维护图连通性问题，OI界已经引入了通过分层维护在 PolyLog 时间复杂度内实现的做法，但是该做法难以拓展到开关点维护图联通性的问题上来。

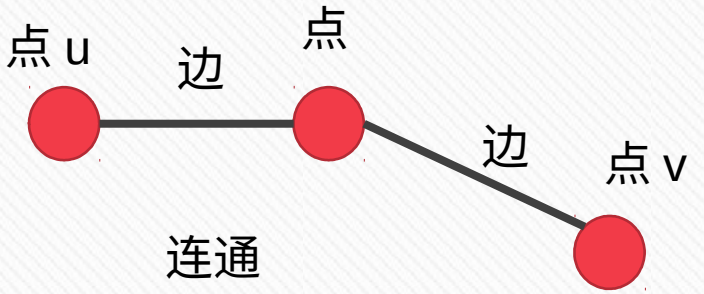
我们将介绍一个预处理时间复杂度$O(m^{\frac{4}{3}})$修改时间复杂度$\tilde{O}(m^{\frac{2}{3}})$的开关点连通性算法

前置知识

一些概念

$G = (V, E)$表示一个图，其中V为G中的点集，E为G中的边集，如果没有特殊说明，将用来表示点数和边数。$|V|, n$（若无特殊说明，接下来我们所讨论的图都认为是无向图）

两点在图中连通，当且仅当存在一个的节点序列$u_0, u_1, u_2, \dots, u_k = v$,使得$(u_0, u_1), (u_1, u_2), (u_2, u_3), \dots, (u_{k-1}, u_k)$均属于$E$



一些概念

$G=(V,E)$是连通图，当且仅当对于V中任意两点u,v，均有u,v在图G中连通。

是连通图，当且仅当对于任意两点，均有在图中连通。

是导出子图，当且仅当，且对于中任何一条边，若，则。

$G'=(V',E')$是$G=(V,E)$的导出子图，当且仅当$V'\subseteq V$且$E'\subseteq E$，且对于E中任何一条边(u,v)，若$u\in V',v\in V'$，则$(u,v)\in E'$。

对于任意一个点集，显然存在唯一的一个图满足是的导出子图，此时称为导出子图。

对于任意一个点集$V'\subseteq V$，显然存在唯一的一个图$G'=(V',E')$满足$G'=(V',E')$是$G=(V,E)$的导出子图，此时G'称为G的V'导出子图。

的极大连通导出子图称为的连通块或者的连通分量。

$G=(V,E)$的极大连通导出子图称为G的连通块或者G的连通分量。

中与节点相连的边的数量称为节点在图中的度数。

$G=(V,E)$中与节点u相连的边的数量称为节点u在图G中的度数。

动态图的两种修改

边修改：在图$G=(V,E)$中加入或者删除一条边

点修改：将一个节点加入某个点集$S \subseteq V$中或者从S中删除一个节点

查询时需要查询图G的S导出子图的信息。

- 边修改：在图中加入或者删除一条边
- 点修改：将一个节点加入某个点集中或者从中删除一个节点
- 查询时需要查询图的导出子图的信息

如果没有特殊说明，在本课件中动态图均指动态图连通性，即查询为给定两个节点u, v，查询u, v是否在图G的S导出子图中连通。

- 如果没有特殊说明，在本课件中动态图均指动态图连通性，即查询为给定两个节点，查询是否在图的导出子图中连通。

动态图的两种修改

其中需要支持边修改的动态图连通性已经可以做到 $O(\log n)$ 均摊时间复杂度。

本课件中讨论的 dynamic subgraph connectivity，实际上是需要支持点修改的动态图连通性。

3

修改的两类点连通性

$\tilde{O}(m^{\frac{2}{3}})$的开关点连通性

问题描述

给出一张图 $G=(V,E)$ 和一个点集 $S\subseteq V$，要求支持：

1. 将 V 中的一个节点加入 S 中，或者从 S 中删除
2. 查询两个节点 u,v 是否在 G 的 S 导出子图中连通

给出一张图和一个点集，要求支持：
1. 将中的一个节点加入中，或者从中删除
2. 查询两个节点是否在导出子图中连通

约定

我们约定，S 中的节点被称为打开的节点，在 S 外的节点被称为关闭的节点。这也是我们将 dynamic subgraph connectivity 翻译为开关点连通性的原因。

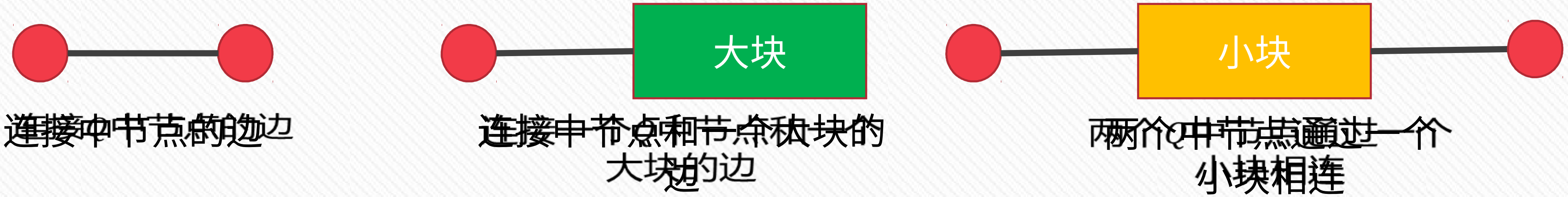
将点集划分成两个部分P和Q，其中只能包含打开的节点，是的子集，初始状态下，P是的子集，Q是P的补集，初始状态下，Q为空集。每次操作，就将中所有打开的节点移动到并重构整个数据结构，以保证中最多只有个打开的节点。除此之外，中节点不能移动到中。

将点集划分成两个部分P和Q，其中只能包含打开的节点，是的子集，初始状态下，P是的子集，Q是P的补集，初始状态下，Q为空集。每次操作，就将中所有打开的节点移动到并重构整个数据结构，以保证中最多只有个打开的节点。除此之外，中节点不能移动到中。

使用只需要支持边修改的动态图连通性来维护G的P的导出子图。对于P的导出子图中的每个连通分量，如果这个连通分量中所有点在图中的度数和大于D，则称其为大块。否则称其为小块。显然最多有$\frac{m}{D}$个大块。

使用只需要支持边修改的动态图连通性来维护G的P的导出子图。对于P的导出子图中的每个连通分量，如果这个连通分量中所有点在图中的度数和大于D，则称其为大块。否则称其为小块。显然最多有$\frac{m}{D}$个大块。

建立一张新图$G'=(V',E')$和点集S'，其中V'中的每个节点对应每个大块或者Q中的节点。E'中的边有3种。第一种是图G中的连接两个Q中节点的边，第二种是连接Q中节点和大块的边，第三种代表两个Q中的节点可以通过一个小块相连。S'包含所有对应大块的节点和Q中打开的节点。显然S'中只有$O(\frac{m}{D})$个节点。



修改操作

使用只需要支持边修改的动态图连通性来维护S'的导出子图。

由于S中只有$O(\frac{m}{D})$个节点，所以对Q中边进行修改时，可以暴力枚举S中的每一个节点，判断是否有边相连。时间复杂度为$\tilde{O}(\frac{m}{D})$。

修改操作

如果要对 P 中的节点进行修改，需要将 P 中的节点移动到 C 中。

如果这个节点是 P 中的节点，可以暴力枚举这次修改影响到的边，时间复杂度为 $O(n^2)$。

修改操作

如果这个节点是大块中的节点，这次修改会将大块分裂成若干个大块和小块。注意到分裂出的大块最多只有一个大块，因为大块中所有点在中的度数和会超过原来的大块的一半。用图G'中对应原来的大块的节点对应这个大块，然后暴力枚举与这次修改移动的节点或者其余的大块和小块相连的边来处理这次修改对G'的影响。

修改操作的时间复杂度分析

修改中的节点和修改小块中节点的情况的时间复杂度已经确定为，现在分析修改大块中的节点和重构整个数据结构的时间复杂度。

修改Q中的节点和修改小块中节点的情况的时间复杂度已经确定为$O(\frac{m}{D}+D^2)$，现在分析修改大块中的节点和重构整个数据结构的时间复杂度。

修改操作的时间复杂度分析

除了修改小块中节点的情况，对于图G中的一条边，只有三种情况下会考虑这条边对数据结构的影响：

情况1：这条边的一个端点从小块中移动到了Q中。这种情况只会发生$O(1)$次，只会影响到数据结构中的$O(1)$条边。

情况2：这条边相连的大块在T中对应的节点发生了变化。这种情况每次发生一次，为这条边相连的大块中所有点在T中的深度的变化量减去减少量，所以至多发生$O(\log n)$次，每次只会影响到数据结构中的$O(1)$条边。

情况3：这条边被修改后，这条边从一块相连块相连。这种情况只会发生$O(1)$次，只会影响到数据结构中的$O(1)$条边。

修改操作的时间复杂度分析

综上，如果没有重构整个数据结构，在预处理和修改大块中的节点的过程中，图G中的每条边总共只会对数据结构中的$O(D+\log m)$条边产生影响。由于图G中只有$O(m)$条边，数据结构中的每条边可以在$O(\log m)$的均摊时间复杂度内进行处理，总时间复杂度为$O(mD)$。

如果要重构整个数据结构，$O(mD)$的时间复杂度可以平摊到$O(\frac{m}{D})$次修改操作中，这样单次修改的时间复杂度仍然是$\tilde{O}(\frac{m}{D}+D^2)$。

查询操作

对于查询操作，如果查询的节点是Q中的节点或者大块中的节点，可以直接找到这个节点在C'中对应的节点然后在C'中进行查询。但是如果查询的节点是小块中的节点，需要在这个小块中 dfs 直到到达一个Q中的打开的节点或者确认无法到达，这需要$\tilde{O}(p)$的时间复杂度。

时间复杂度

综上，取$D \approx m^3$，可得修改时间复杂度为$\tilde{O}(m^{\frac{1}{3}})$，查询时间复杂度为$\tilde{O}(m^3)$，预处理时间复杂度为$\tilde{O}(m^{\frac{4}{3}})$。

另外，该数据结构实际上可以同时支持边修改和点修改，时间复杂度均为$O(m^{\frac{2}{3}})$。如果要加入或者删除一条边(u,v)，可以先将u和v移动到C中，然后在C中做相应的修改。

另外，该数据结构实际上可以同时支持边修改和点修改，时间复杂度均为。如果要加入或者删除一条边，可以先将和移动到中，然后在中做相应的修改。

4 总结

总结

虽然对于支持边修改的动态图连通性我们有很好的做法，但如果直接应用到支持点修改的动态图连通性上，会由于度数原因而产生高额复杂度。

因此我们采用了分块的方法，通过定期重构和按照度数分大小块分别处理的思想来获得一个较为优秀的算法。

这启示我们分块平衡也是处理很多问题的有力工具。

参考文献

Chan, Timothy M., Mihai P a ˇ traşcu, and Liam Roditty. "Dynamic connectivity: Connecting to networks and geometry." SIAM Journal on Computing 40, no. 2 (2011): 333-349.

Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. Journal of the ACM(JACM), 48(4):723–760, 2001.

谢谢您的聆听

THANKS YOUR LISTENING