Push the Flow! 解题报告

杭州学军中学 王逸松

1 试题来源

CODECHEF August Challenge 2014

Push the Flow! | CodeChef

PUSHFLOW.pdf

2 试题大意

给一棵n个点的仙人掌（每条边最多属于一个简单环的无向连通图），保证没有重边或自环，每条边有个容量。

有q个操作，每个操作是修改一条边的容量，或询问两个结点之间的最大流。

3 数据范围

$1 \le n \le 100000$

$0 \le q \le 200000$

4 算法介绍

4.1 算法一

首先考虑树上的情况，两个结点之间的最大流就是这两个结点之间路径上边的容量的最小值。

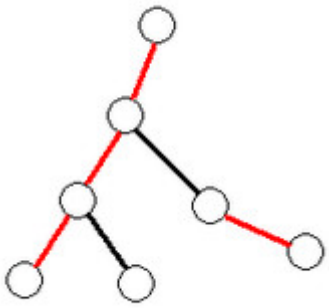
可以用link-cut tree来维护。

对于仙人掌上的问题，我们可以用link-cut cactus [1] 来维护。

link-cut tree维护的是树的一个链剖分，于是我们可以维护仙人掌的一个链剖分。

4.1.1 link-cut cactus的结构

如果没有环，就跟link-cut tree一样，每个结点有个preferred-child，用实边连起来，其他儿子用虚边连起来。



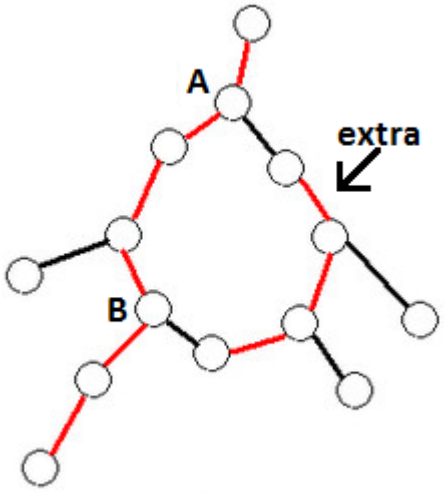
（红色的边为实边，黑色的边为虚边）

对于一个环，我们定义它的父亲为环上离仙人掌的根最近的结点，记为A。我们定义环的preferred-child为环上最后一次access到的结点，记为B。

我们将A和B的最短路用实边连起来。（在这道题里，两点之间的最短路是指经过边数最少的路径）

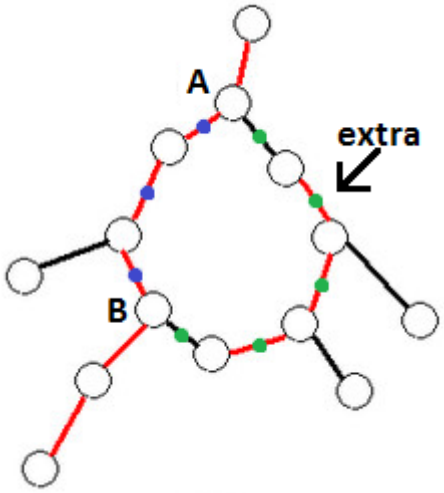
对于环上其他的结点构成的链，也用实边连起来，我们称这条链为**额外链**。

[1] 在vfleaking的博客里有较详细的介绍



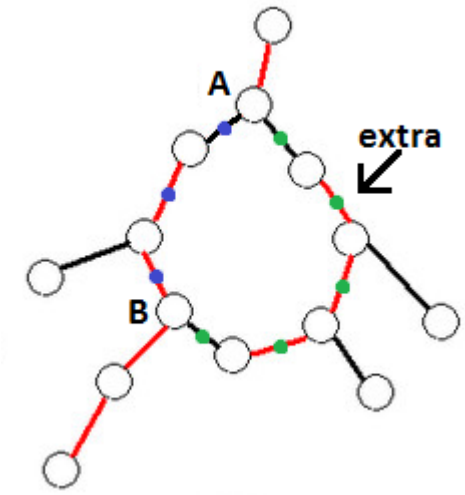
4.1.2 如何在splay上维护边的信息

在每条边上都加个点，如图所示：



注意环上额外链两段与A和B之间的边（黑色边），我们将它直接接在额外链的两端来维护。

4.1.3 一种特殊情况



如图，上一次access了结点A，这导致了AB链上与A相连的边变为虚边，但是表示这条边的splay结点还在AB链的splay上。

要注意特判这种情况。

4.1.4 access操作

类比link-cut tree的access操作。

假设当前access到结点x，我们先找到x在splay上的前一条边e。

如果e不在环上，就跟link-cut tree的一样处理。

否则先把这个环的AB链的左右两端断开（要注意考虑特殊情况），然后把AB链和额外链连接起来。

然后把x提到splay的根，选择较短的一边连回去作为新的AB链，并将另一边设置为额外链。

4.1.5 换根操作

类比link-cut tree的换根操作，在access后对根到该结点的路径打翻转标记来换根。

要注意这会导致这条路径经过的环的A和B结点互换，而且额外链方向不正确。

解决方法是对于access到的每一个环，检查A和B在splay中的前后顺序，如果反了，就交换A和B，并给额外链打翻转标记。

4.1.6 如何用link-cut cactus来解决这道题

对于每个环，A和B结点之间的最大流为AB链上的容量最小值加上额外链上的容量最小值，所以可以将额外链的容量最小值加到AB链的每一条边的容量上。

询问时只要换根再access就行了。

修改一条边的容量时，将根换成这条边的一个端点，然后access这条边的另一个端点，此时需要修改的边一定不会在某个环的额外链上，所以直接修改即可。

4.1.7 复杂度

类比link-cut tree，结点和环的preferred-child的切换次数是$O((n+q)\log n)$的，所以总复杂度是$O((n+q)\log^2 n)$的。

4.2 算法二

先考虑没有修改的情况。

对于某个环C，假设C上容量最小的边为e。考虑一个询问(S, T)，如果S到T的路径经过了环C，那么在环C上一定是有一条路径经过了e，另一条路径不经过e。只考虑环C上的部分，经过e的那条路径的容量最小值一定是e的容量。所以可以把e从环上断开，然后把e的容量加到环C上其他所有边的容量上。

然后我们维护的就是仙人掌的一棵生成树，用link-cut tree即可。

询问时只要在生成树上询问。

对于修改操作，如果修改的边在环上，我们在修改后重新找到环上容量最小的边，然后进行几次link和cut操作即可。

时间复杂度是$O((n+q)\log n)$。

注意如果把修改操作改为“给两点之间最短路径上所有边的容量增加一个值”，这个算法就没法做了。

5 参考程序

算法一：`PUSHFLOW_lcc.cpp`

算法二：`PUSHFLOW_lct.cpp`