仙人掌相关问题

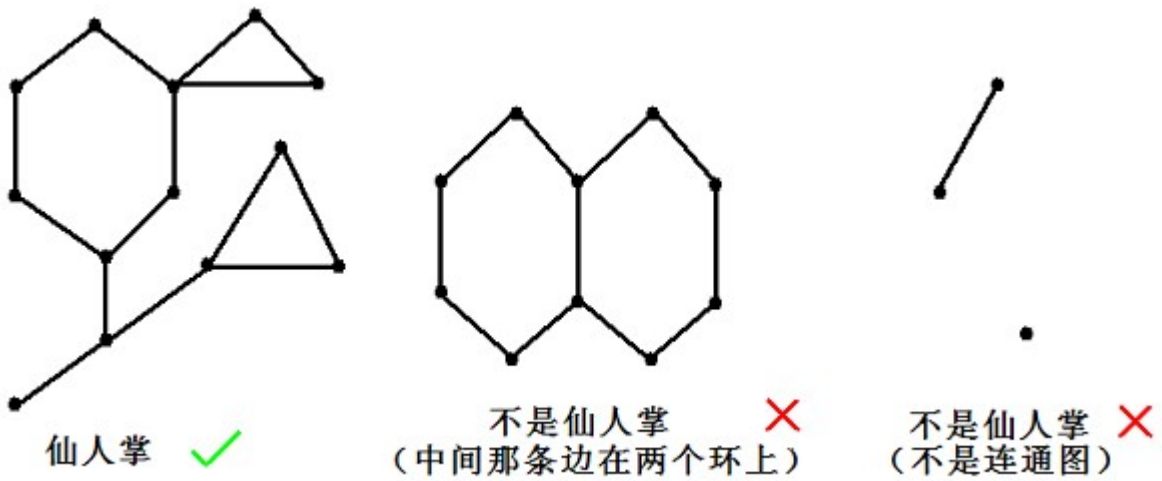
成都七中　nzhtl1477

什么是仙人掌

- 仙人掌分为点仙人掌和边仙人掌

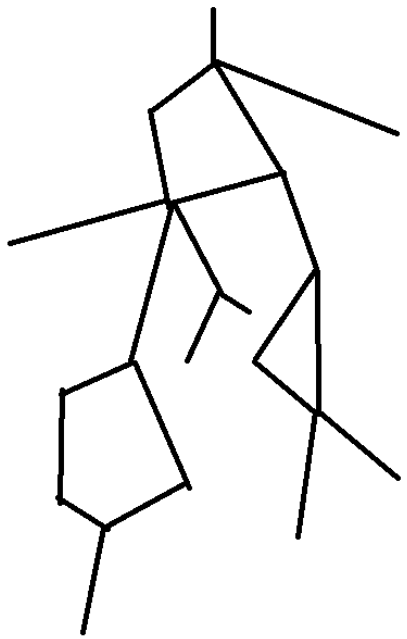
边仙人掌

- 如果一个无向连通图的任意一条边最多属于一个简单环，且不存在自环，我们就称之为仙人掌。



点仙人掌

- 如果一个无向连通图的任意一个点最多属于一个简单环，且不存在自环，我们就称之为点仙人掌。
- 可以发现点仙人掌的限制比边仙人掌的限制强很多



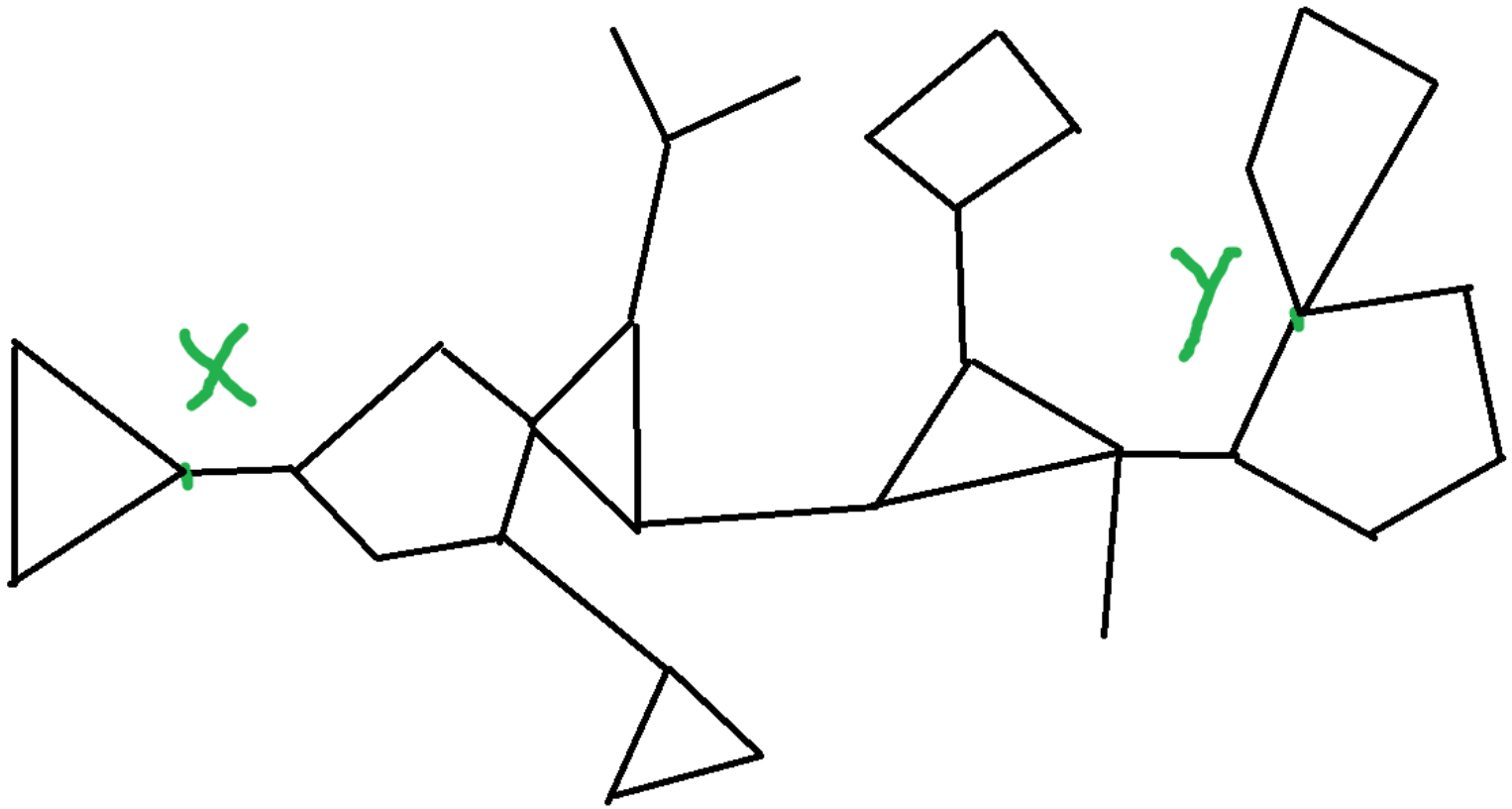
沙漠

- 沙漠对于仙人掌就和森林对于树一样，去掉了前面关于连通性的限制

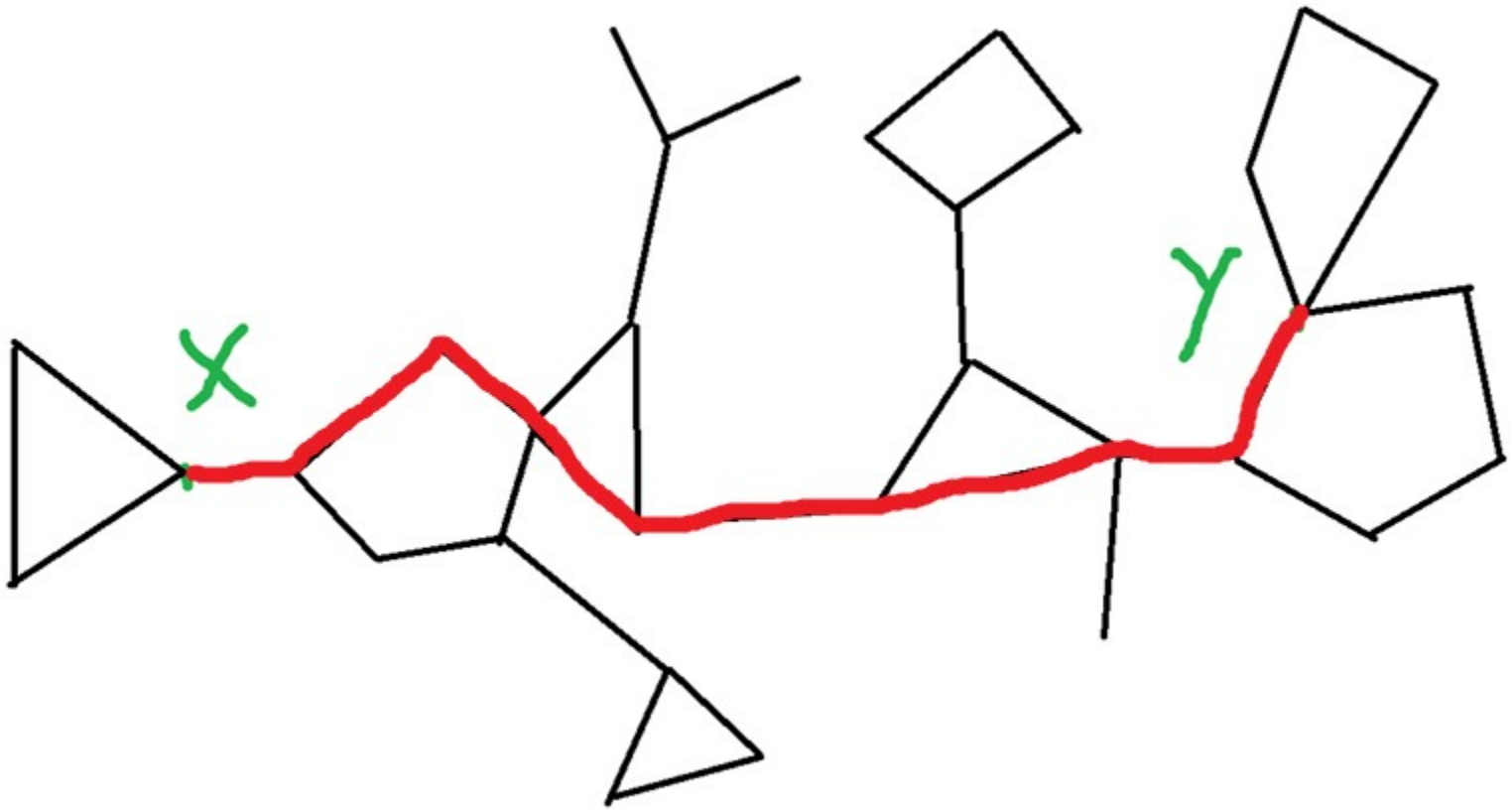
仙人掌上路径

- 一般有三种定义
- 1. 从 x 到 y 的最短简单路径
- 2. 从 x 到 y 的最长简单路径
- 3. 从 x 到 y 的简单路径的并集
- 以下按简单路径是一条不包含重复点的路径的定义

仙人掌上路径

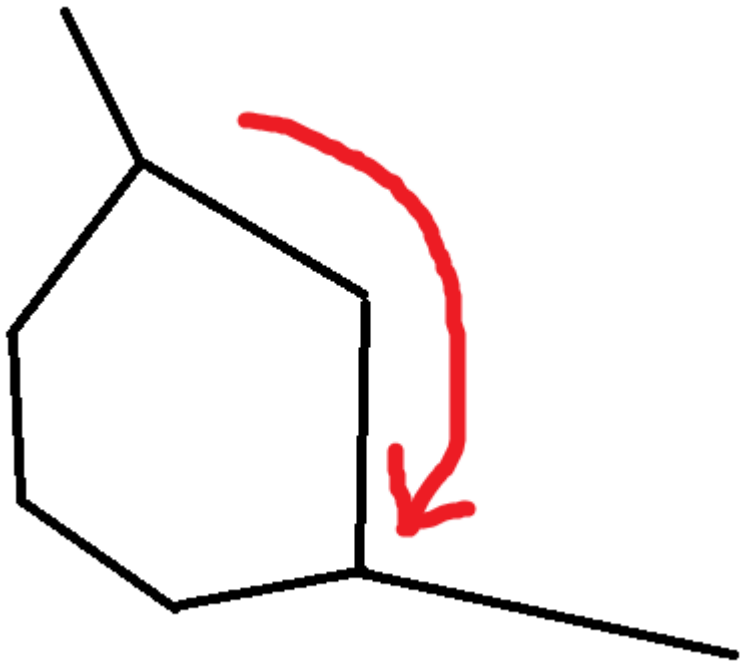


x 到 y 的最短简单路径



x 到 y 的最短简单路径

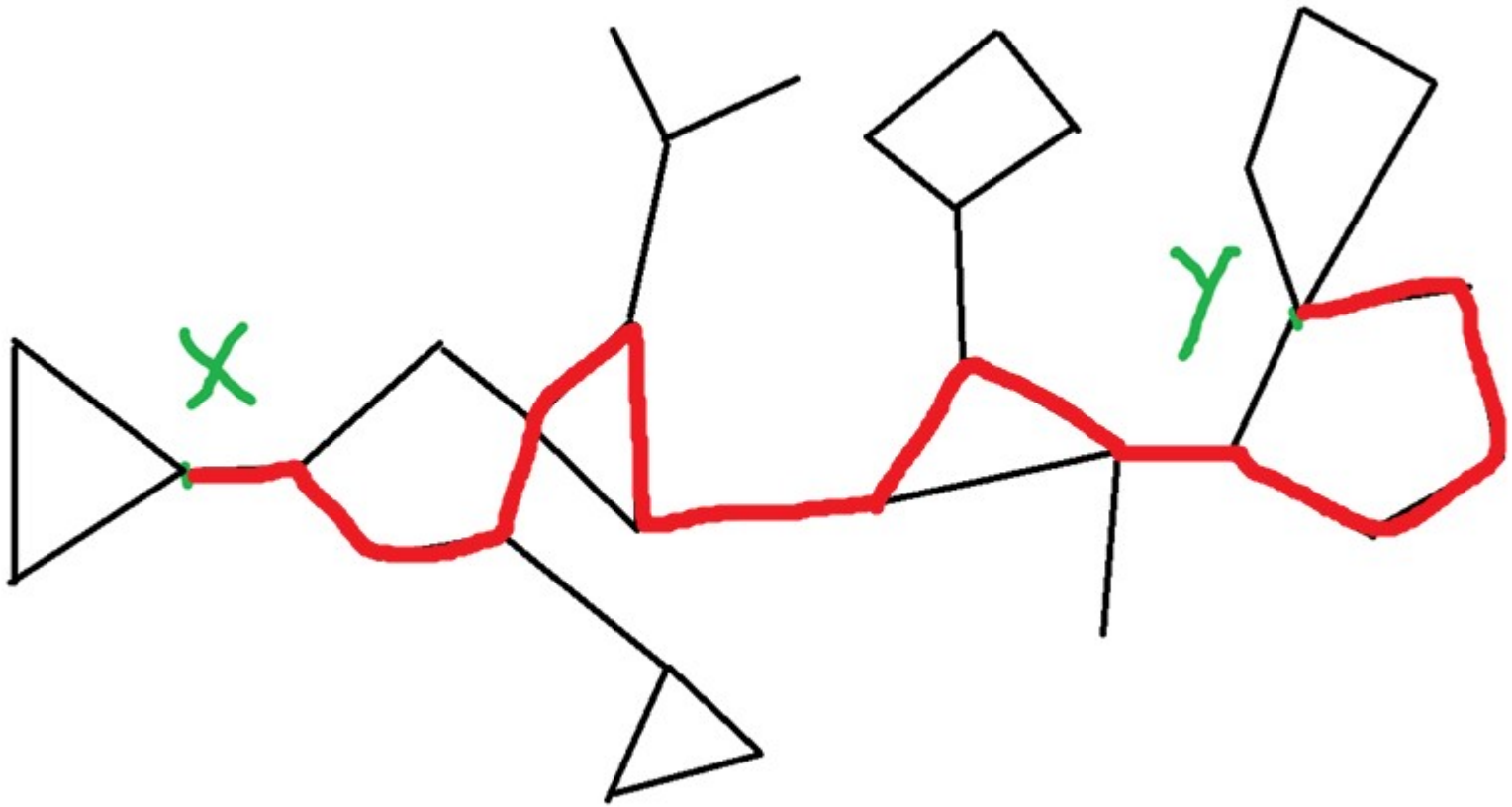
- 即所有 x 到 y 必须经过的割边，加上每个必须经过的环上较短的一段



x 到 y 的最短简单路径

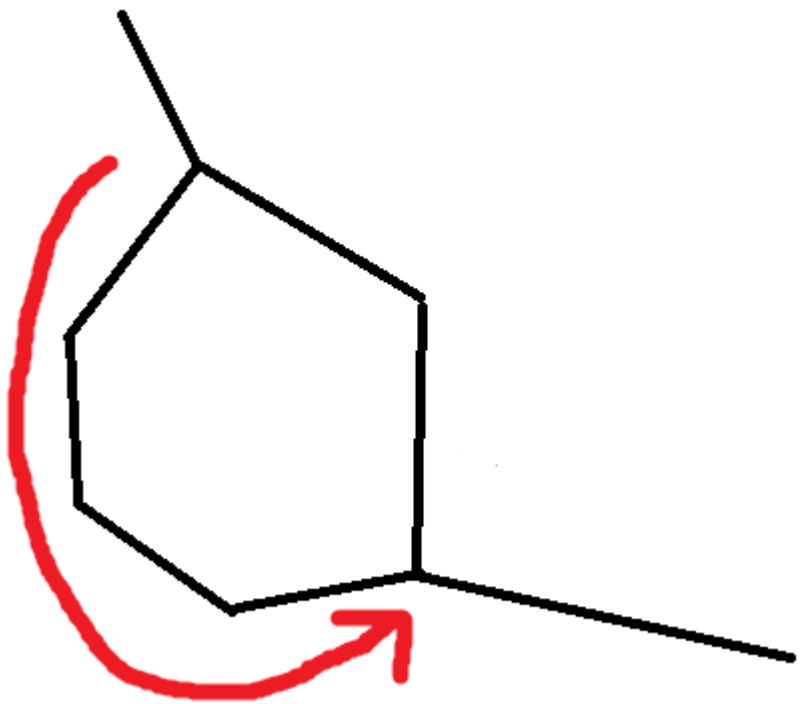
- 如果端点在环上，则取环上较小的一段

x 到 y 的最长简单路径



x 到 y 的最长简单路径

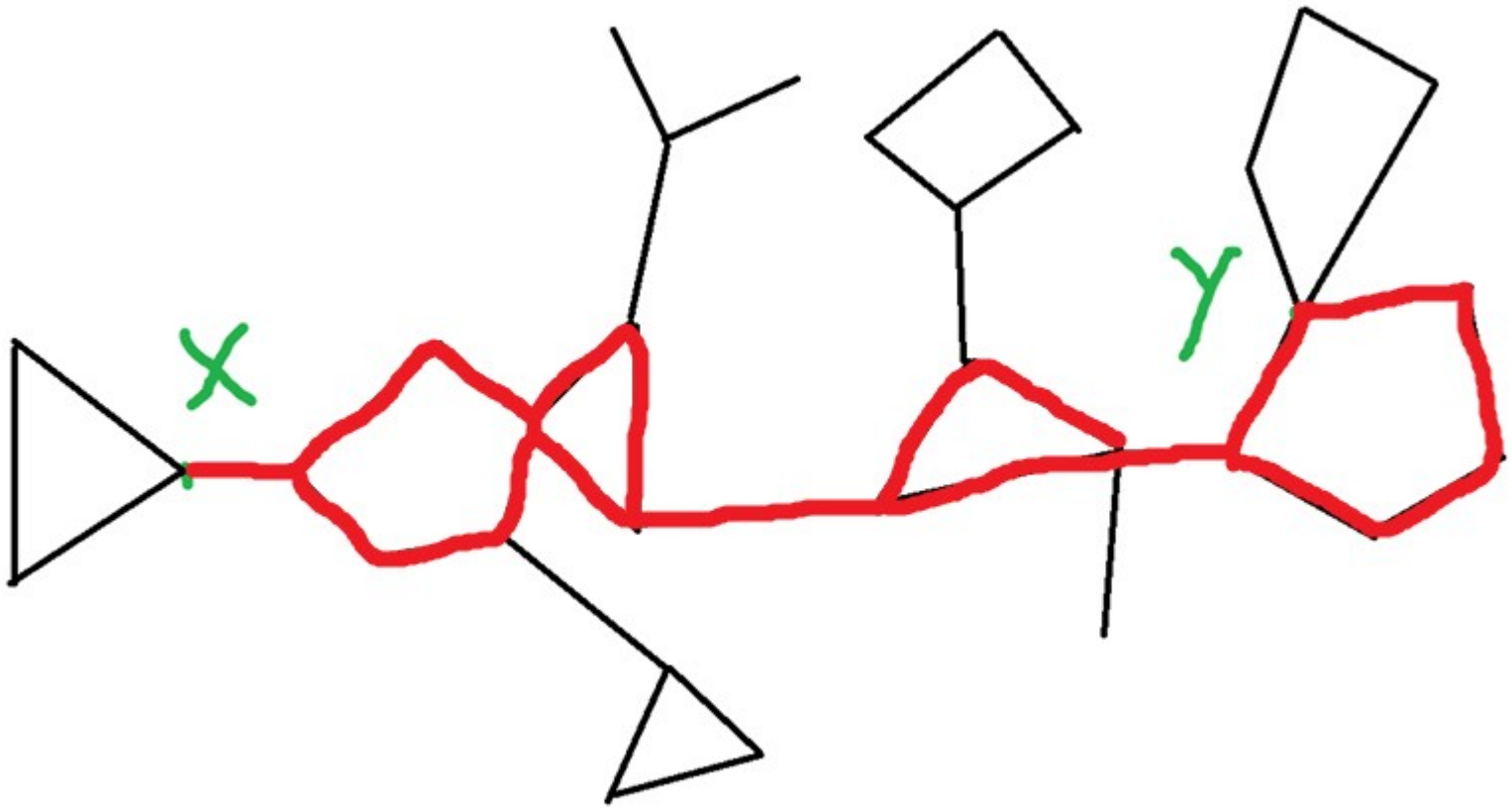
- 即所有 x 到 y 必须经过的割边，加上每个必须经过的环上较长的一段



x 到 y 的最长简单路径

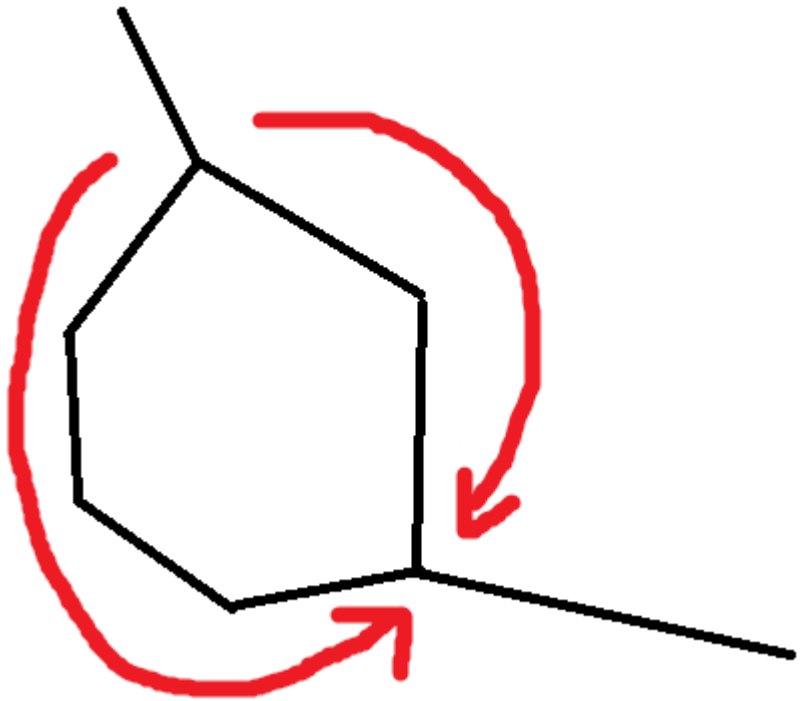
- 如果端点在环上，则取环上较大的一段

x 到 y 的简单路径的并



x 到 y 的简单路径的并

- 即所有 x 到 y 必须经过的割边，加上每个必须经过的环

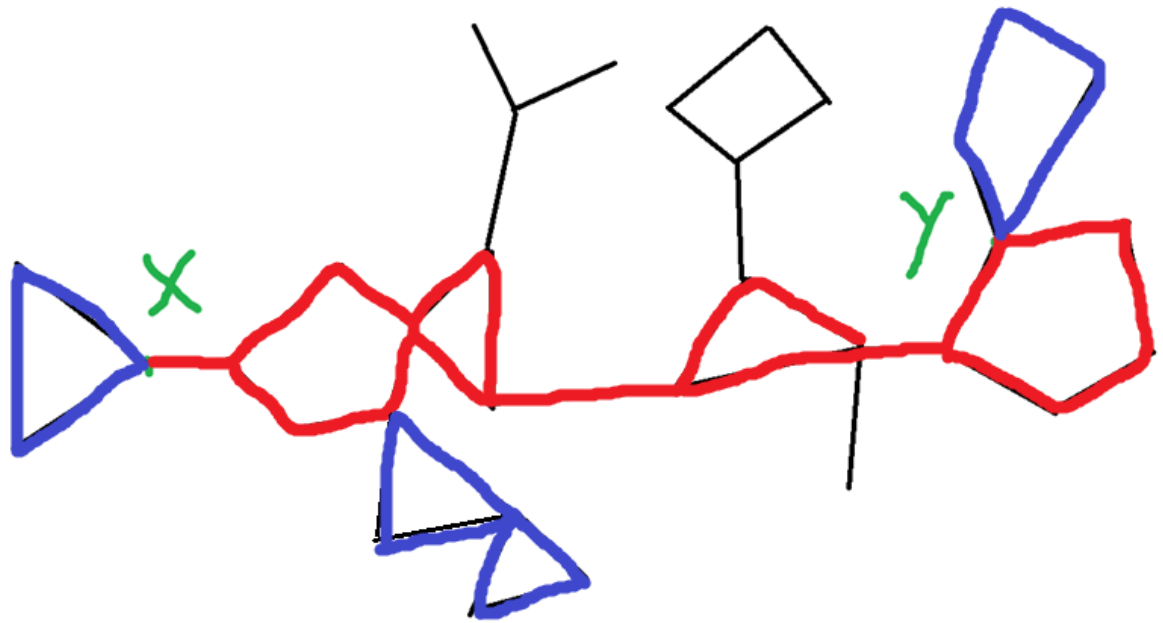


x 到 y 的简单路径的并

- 如果端点在环上，则取整个环

Notice

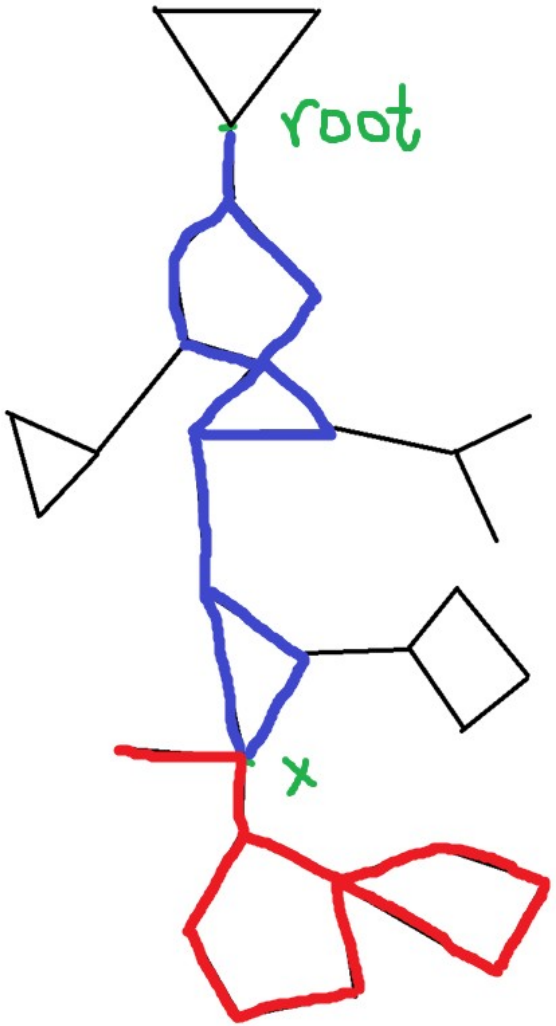
- 注意到这里点和边所定义的简单路径有不同的性质：
- 比如如果是按边定义的简单路径，则如图所示 x 所在的左边那个环是可以取到的，路径上点被经过的环全部都是可以取到的



子仙人掌

- 点 x 的子仙人掌定义为：删除从根到点 x 的所有简单路径上的所有边后，点 x 所在的连通块。

子仙人掌



仙人掌最短路

- 即将仙人掌看做一个普通的无向图的两点间最短路

// 仙人掌邻域

- // 仙人掌上，点 x 的 y 邻域即 {z|dist(x,z)<=y}，这里 dist(x,z) 表示 x 和 z 的仙人掌最短路长度

仙人掌 DP

洛谷 4129 [SHOI2006] 仙人掌

- 仙人掌图（cactus）是一种无向连通图，它的每条边最多只能出现在一个简单回路（simple cycle）里面。从直观上说，可以把仙人掌图理解为允许存在回路的树。但是仙人掌图和树之间有个本质的不同，仙人掌图可以拥有多个支撑子图（spanning subgraph），而树的支撑子图只有一个（它自身），我们把仙人掌图的支撑子图的数目称为“仙人数”。你的任务就是计算给定图的“仙人数”。

Solution

- 考虑每条割边，如果去掉了之后图便不连通，显然不是一个支撑子图
- 考虑每个环上，如果去掉了超过一条边，则图不连通，显然不是一个支撑子图
- 如果每个环上只去掉最多一条边，则这个图是原图的支撑子图
- 答案即为每个环大小 +1 的乘积
- 找环可以用 tarjan 算法来实现，总时间复杂度 O(n)

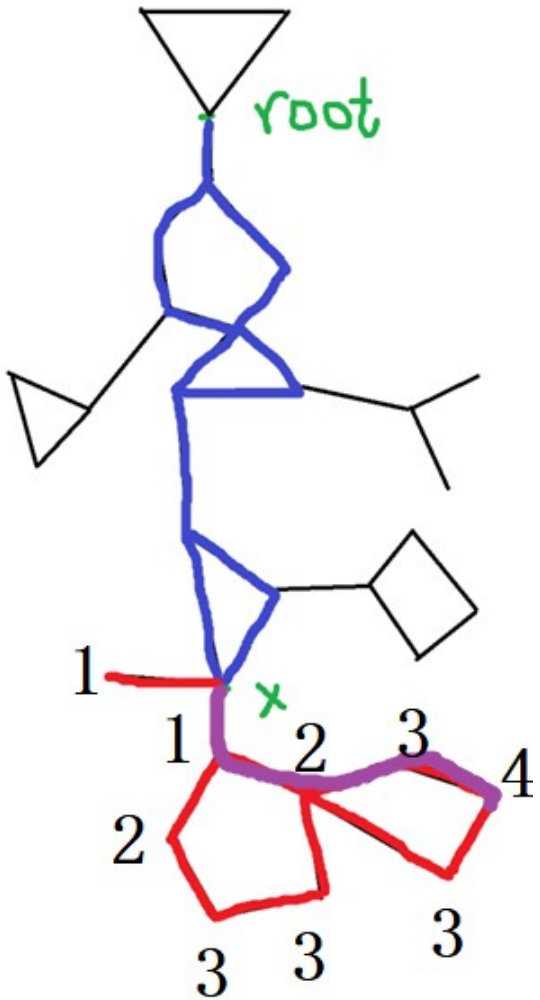
洛谷 4244 [SHOI2008] 仙人掌图 II

- 求一棵边仙人掌的直径
- 定义一个图的直径为这张图相距最远的两个点的距离。

Solution

- 先随便定一个根 1，以此为基础开始 dp
- 定义一个点 x 的深度为 dist(1,x)
- 定义 f[i] 为 i 的子仙人掌的高度，即从 i 的子仙人掌里面深度最大的点的深度减去 i 的深度

Solution

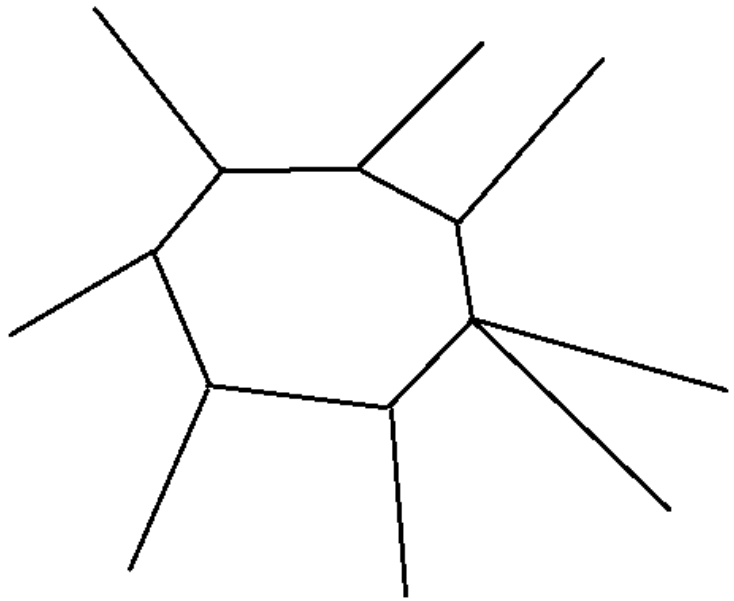


Solution

- 对于一个非环上节点的 i
- f[i]=max(f[j]+1),i 和 j 之间有边，且 j!=fa[i]
- 更新答案： ans=max(ans,f[j]+f[k]+2),i 和 j,k 都相邻且 j!=fa[i],k!=fa[i]

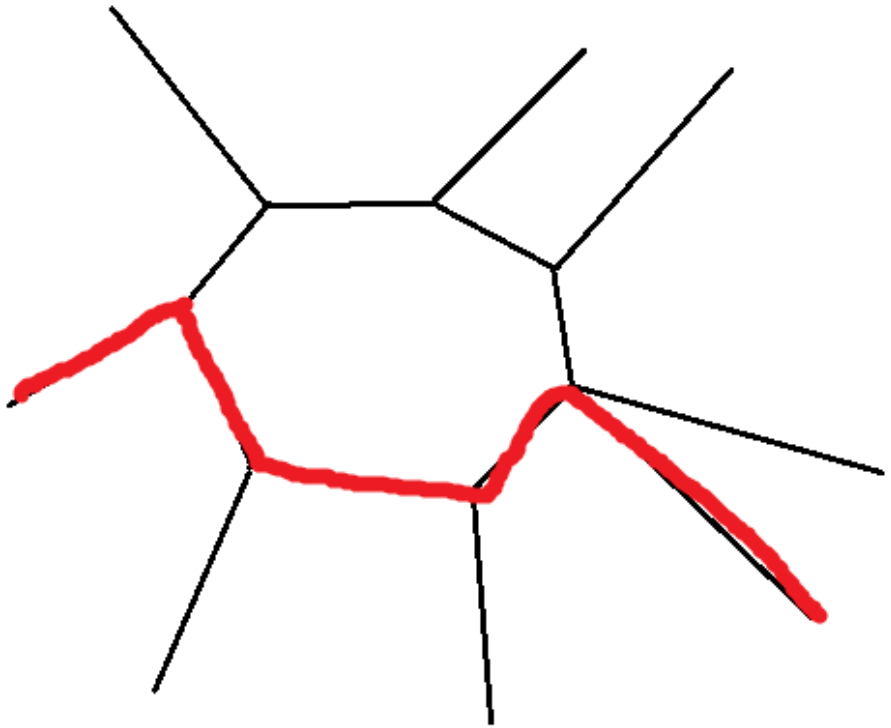
Solution

- 对于一个环上节点的 i
- 我们对这个环一起处理
- 以这个环为基环，可以发现是一个基环树，不用考虑其他节点构成的形态



Solution

- 即最大化 `f[x]+f[y]+dist(x,y)`
- 将环倍长，维护 `dist(x,y)` 构成的简单路径在 `x` 的逆时针方向，然后跑单调队列维护最大值即可



Solution

- 做完之后，再对环上深度最小的那一个点 x 算出 f[x]=max(f[y]+dist(x,y))， x 和 y 在同一个环上
- 由于除了 x 以外，所有节点的子仙人掌都不包含这个环，所以 f 不变
- 这里我们对一个有多个环包含着的点需要对每个环依次处理

- 总时间复杂度 O(n)

洛谷 4410 [HNOI2009] 无归岛
bzoj4316: 小 C 的独立集

- 求一棵边仙人掌的点最大独立集

Solution

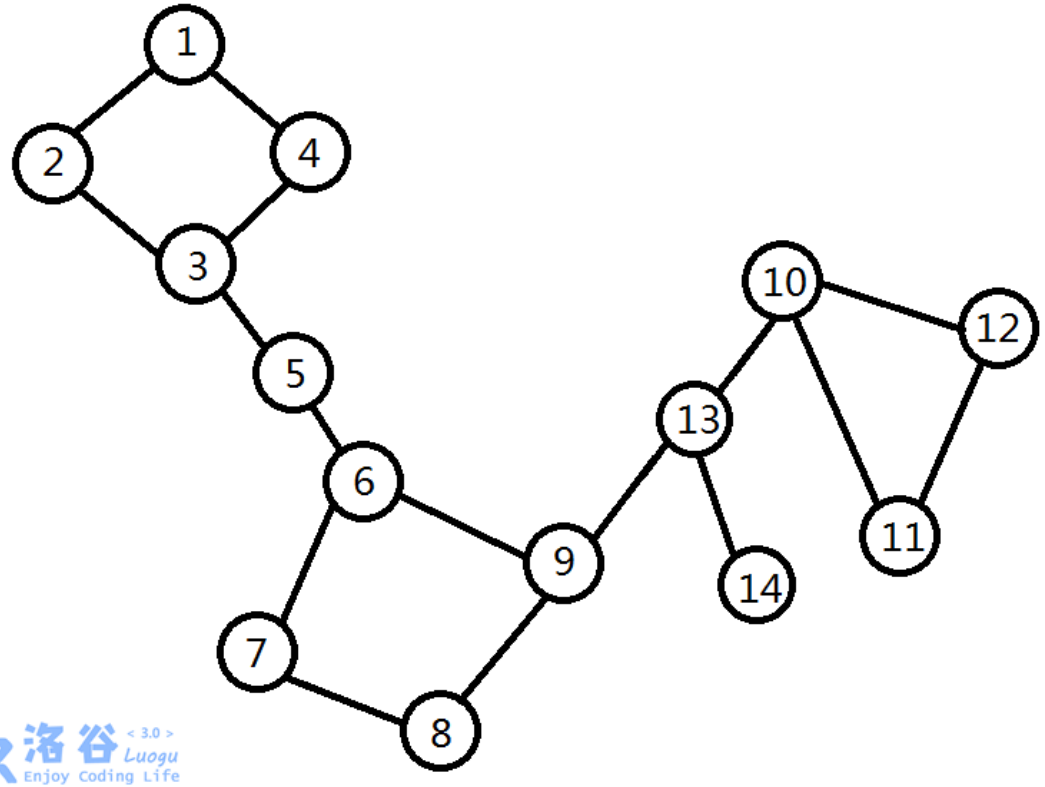
- 类似前面的那个题，定义 f[i][0/1] 表示在 i 的子仙人掌加上点 i 所构成的子图中，是否选 i 这个位置的子仙人掌的最大独立集
- 然后和树上的最大独立集类似的方法转移即可
- 也可以在 DFS 树上直接合并，由于此题信息特殊，具体做法不讲了

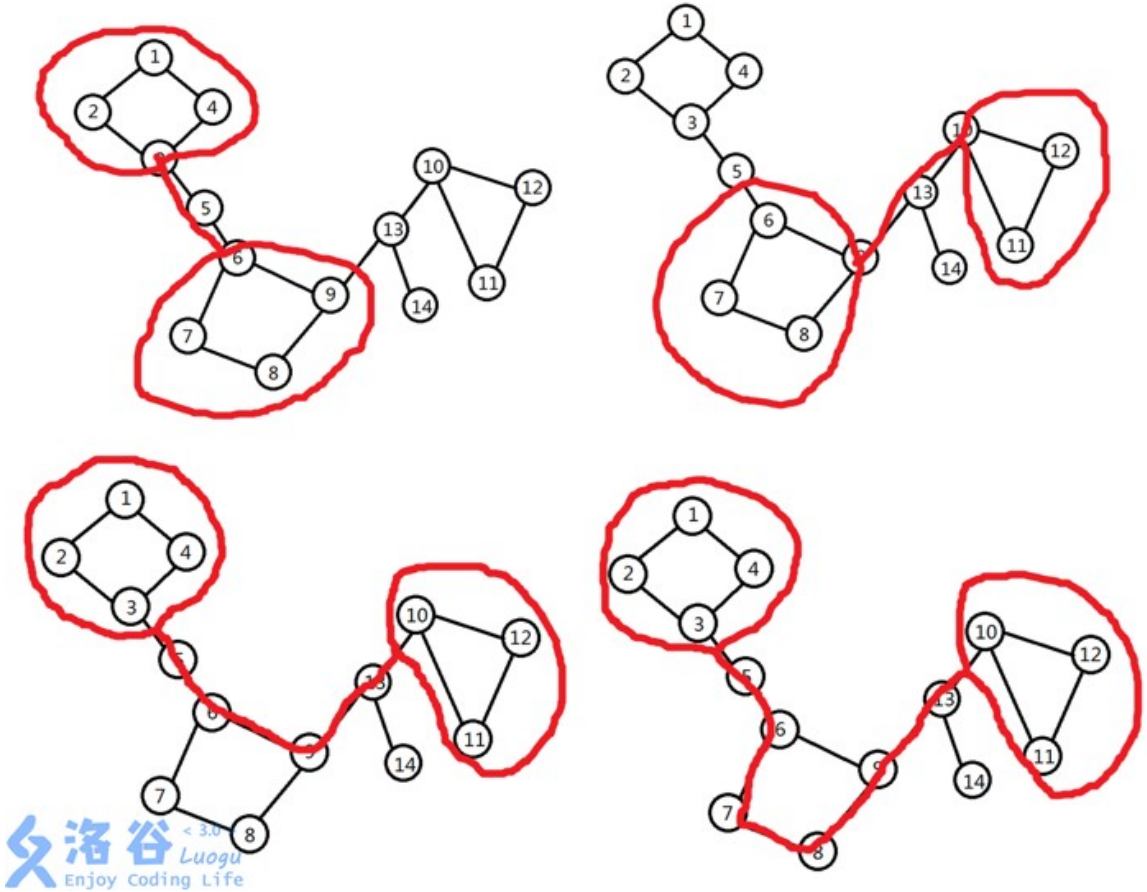
- 总时间复杂度 O(n)

我自己出的一个题

- 给你一个图，保证每个点最多属于一个简单环，每个点度数最多为 3，求这个图有多少“眼镜图形个数”其中“眼镜图形个数”，定义为三元组 (x,y,S)，其中 x 和 y 表示图上的两个点，S 表示一条 x 到 y 的简单路径，而且必须满足:
- 1.x 和 y 分别在两个不同的简单环上
- 2.x 所在的简单环与路径 S 的所有交点仅有 x，y 所在的简单环与路径 S 的所有交点仅有 y。
- (x,y,S) 与 (y,x,S) 算同一个眼镜
- 答案对 19260817 取模

我自己出的一个题





我自己出的一个题

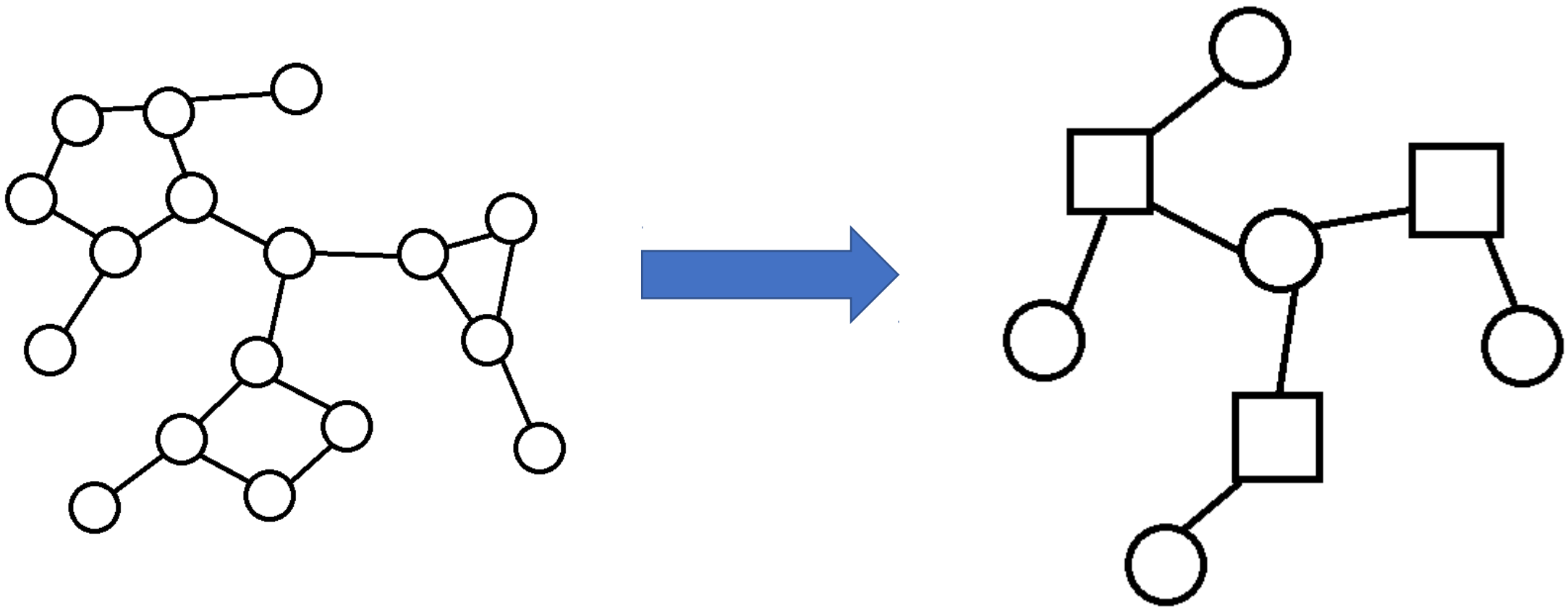
测试点编号	n的范围	m的范围	特殊性质
测试点1	n <= 10	m <= 20	
测试点2	n <= 20	m <= 40	
测试点3	n <= 20	m <= 40	
测试点4	n <= 2000	m <= 4000	
测试点5	n <= 2000	m <= 4000	
测试点6	n <= 1000000	m <= 2000000	简单环个数 <= 2000
测试点7	n <= 1000000	m <= 2000000	简单环个数 <= 2000
测试点8	n <= 1000000	m <= 2000000	
测试点9	n <= 1000000	m <= 2000000	
测试点10	n <= 1000000	m <= 2000000	

Solution

- 30 分做法：爱咋暴力咋暴力

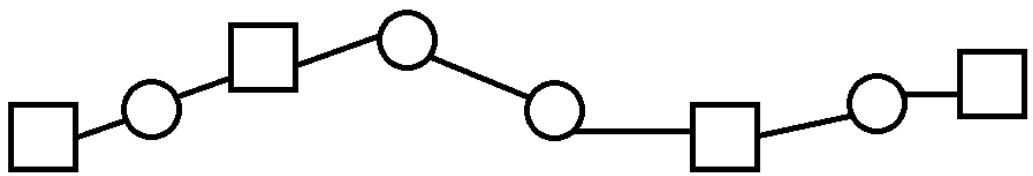
Solution

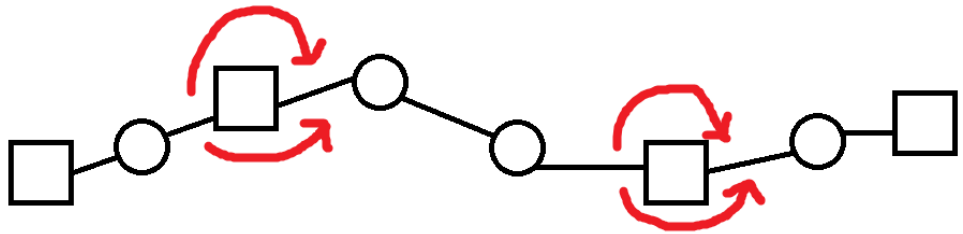
- 50 分做法：我们考虑把环给缩掉，缩了之后的点叫做方点，然后本来树上的点叫做圆点



Solution

- 然后想想怎么用这个新图来数眼镜
- 首先眼镜的两端都必须是一个方点
- 然后可以发现如果一条两端都是方点的路径上总共有 x 个方点，则这两个端点可以构成 2^(x-2) 个眼镜（每次可以走两端）





Solution

- 如图这构成了 4 □ 🕶个眼镜
- 于是我们可以 O(n^2) 缩掉所有的环，然后对于每个方点 DFS 一遍
- 总复杂度 O(n^2)

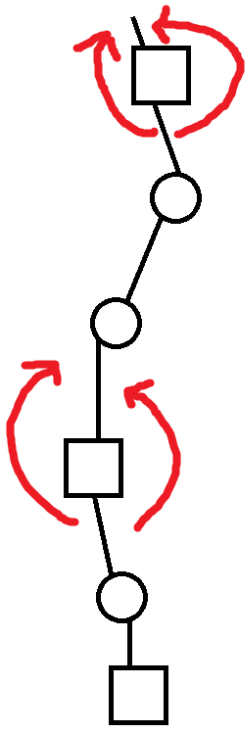
Solution

- 70 分做法：如果我们 DFS 或者用 tarjan 算法找环，然后重构出这个图，这部分复杂度 O(n)

- 然后注意到只有 2000 个简单环，所以我们把一条链上连续的圆点给删掉，这个图会变成一个 4000 个节点左右的新图，然后暴力对于每个方点 DFS 即可

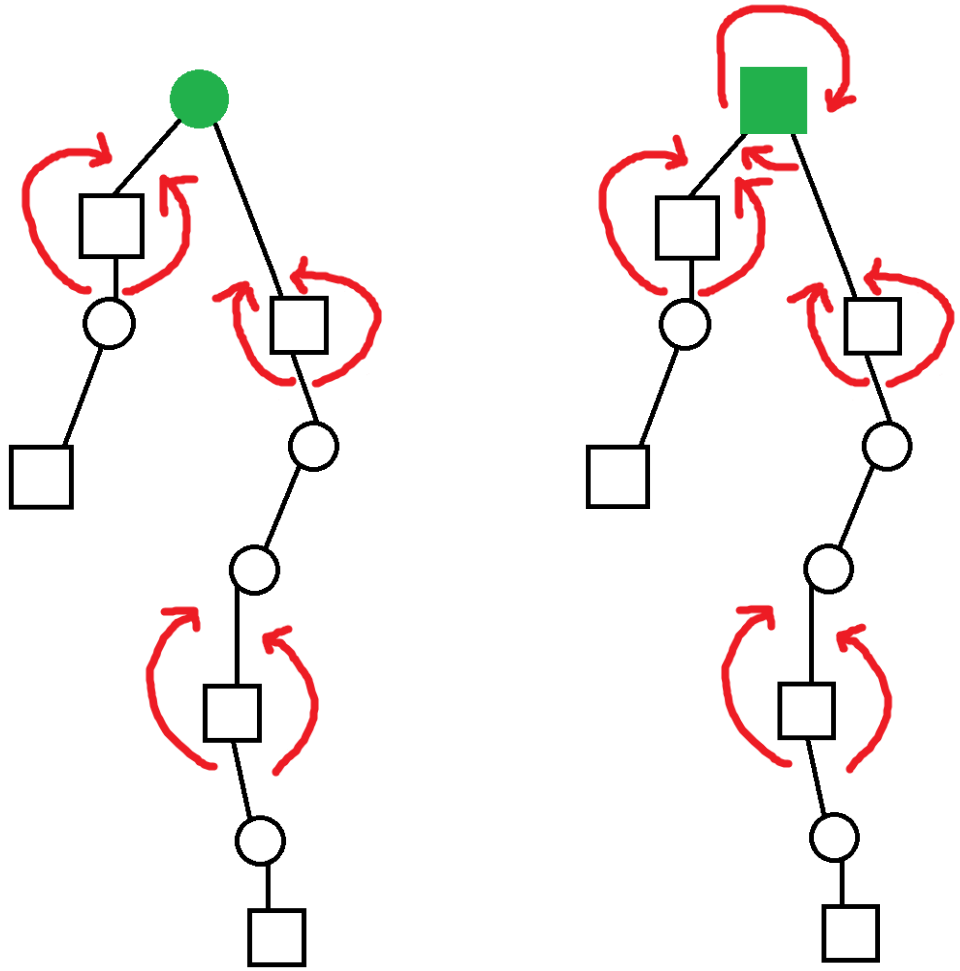
Solution

- 100 分做法：
- 考虑对于生成的这个树进行树形 DP
- 用 f[x] 表示以 x 为根的子树，到 x 构成的“一半的眼镜”的数量
- 也就是说这样的：



Solution

- DP 到这个点的时候，可以把子树中两个“半眼镜”在这个点的位置拼起来



Solution

- 半眼镜维护的方法就是
- `f[x] = Σ( f[ son[x] ] );`
- 如果 `x` 是圆点，则 `f[x]` 不变
- 如果 `x` 是方点，则 `f[x] = f[x] * 2 + 1`
- 因为下面上来的每个半眼镜都可以走两个方向，然后这个点也可以作为一个半眼镜的端点
- 然后合并答案的时候直接合并就可以了
- 也要分当前的点是圆点还是方点讨论一下
-
- 总时间复杂度：`O( n )`

仙人掌最短路

洛谷 5236 【模板】静态仙人掌

- 给一棵边仙人掌，有边权，多次查询两点间最短路径

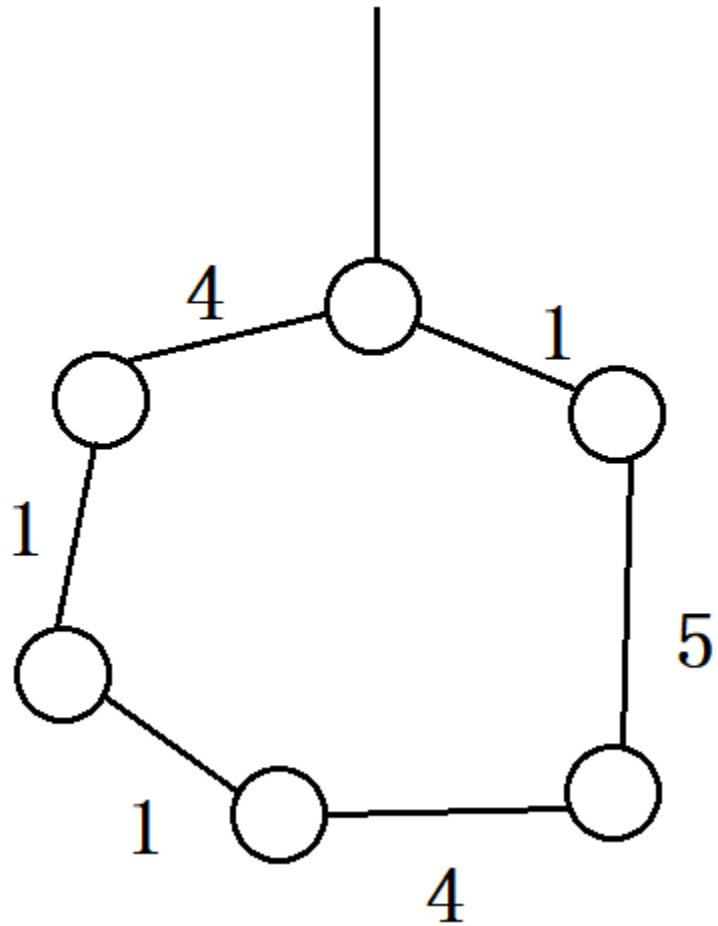
Solution

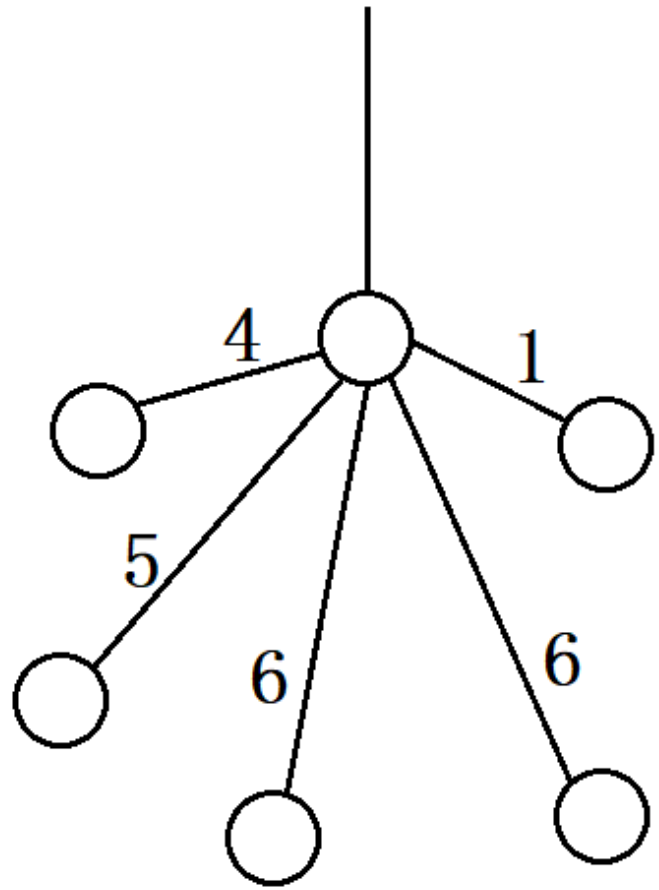
- 建一棵静态仙人掌的树，有两种方法

Solution

- 第一种是我们先随便找一个根，然后对于每个环，将所有环上的点 y 向其深度最小的点 x 连边，边权为 dist(x,y)，并且删掉原始环
- 第二种是我们对每个环建一个虚点，虚点向每个环上节点 y 连边，边权为 dist(x,y) ，并且删掉原始环

Solution



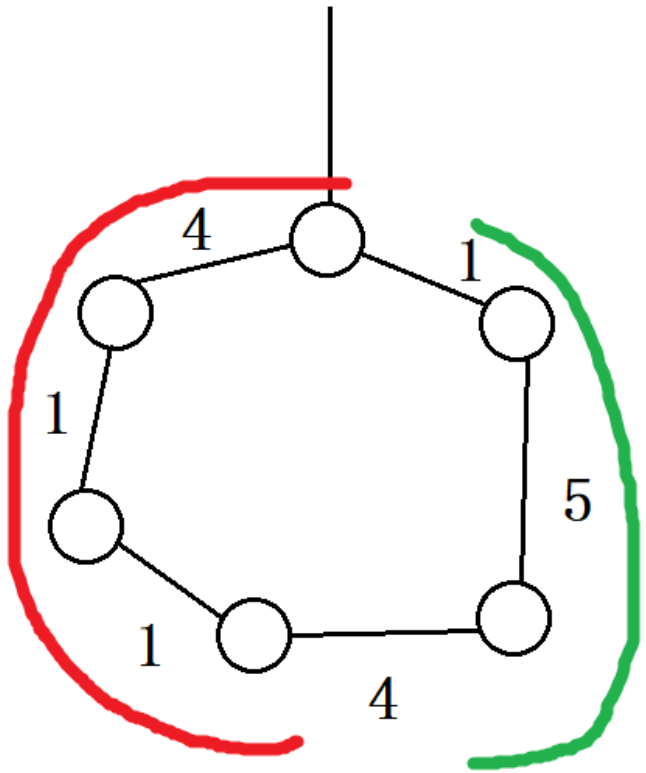


Solution

- 可以发现这样我们把仙人掌变成了一棵树
- 定义一个环由两个半环构成:
- {y|x 到 y 的顺时针距离 >x 到 y 的逆时针距离 }
- {y|x 到 y 的顺时针距离 <=x 到 y 的逆时针距离 }
- 每次查询两点 x,y 间距离
- 如果 x 和 y 在同一个环上，设环上深度最小的节点为 z ，环长为 len ，直接判断他们是否在同一个半环
- 如果在同一个半环，则为 |dist(x,z)-dist(y,z)|
- 如果不在同一个半环，则为 min(dist(x,z)+dist(y,z),len-dist(x,z)-dist(y,z))

Solution

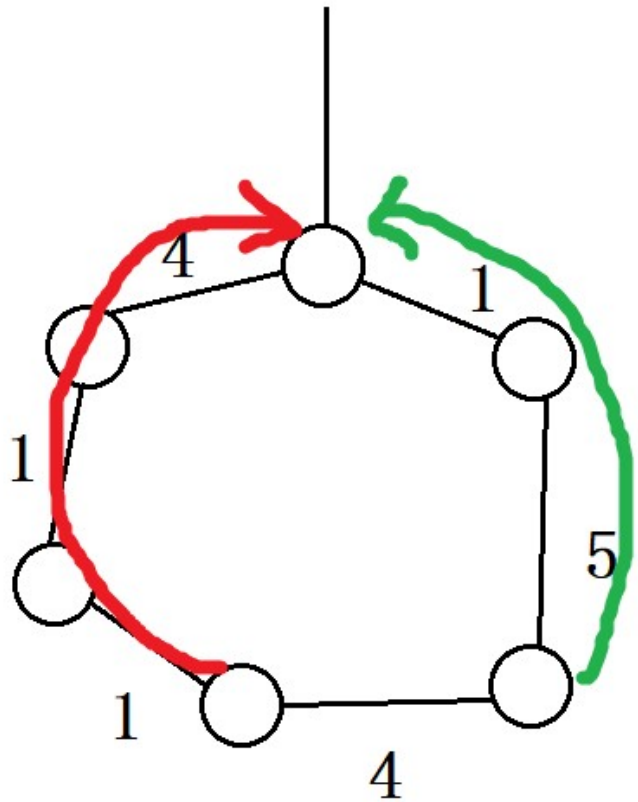
- 两个半环

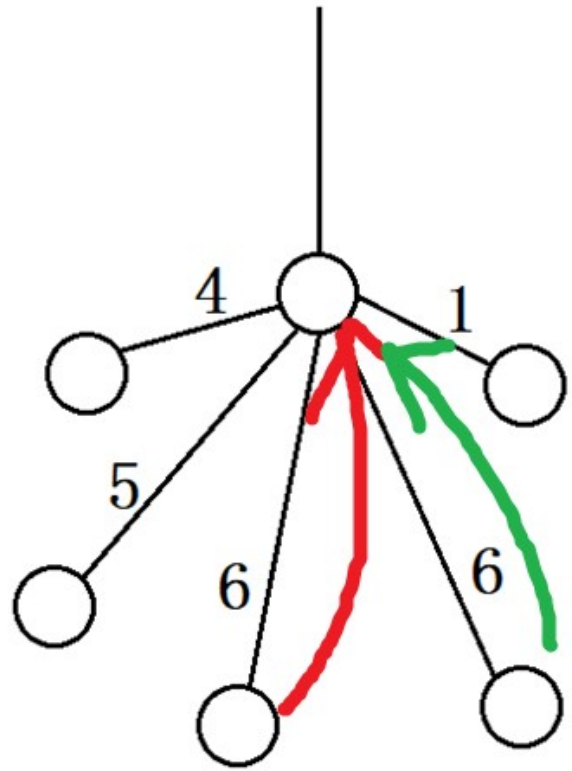


Solution

- 如果 x 和 y 不在同一个环上，我们一定会先让 x 和 y 走到其所在环中最浅的点上，这个相当于我们在刚才建的那个树上 x 和 y 都走向了 father
- 然后可以发现我们 x 和 y 每次在建的那个树上面走向 father，相当于走过了一条割边，或者沿着最短路径走过了一个环

Solution





Solution

- 然后这个和求 lca 类似，最后会走到同一个点 z ，如果不存在一个环，使得 z 是该环上深度最小的点，那我们 dist(x,y) 已经求完了
- 如果是的话，记住上一步走过来的 x 和 y ，判断一下 x 和 y 是否在同一个环上，如果否的话平凡，否则变成了前面 x 和 y 在同一个环里面的情况，套用即可

- 由于 lca 和建树都是线性的，所以总时间复杂度 O(n)

Codechef : Push the Flow!（弱化版）

- 给一棵边仙人掌，边权，多次查询，每次求两点间的最大流

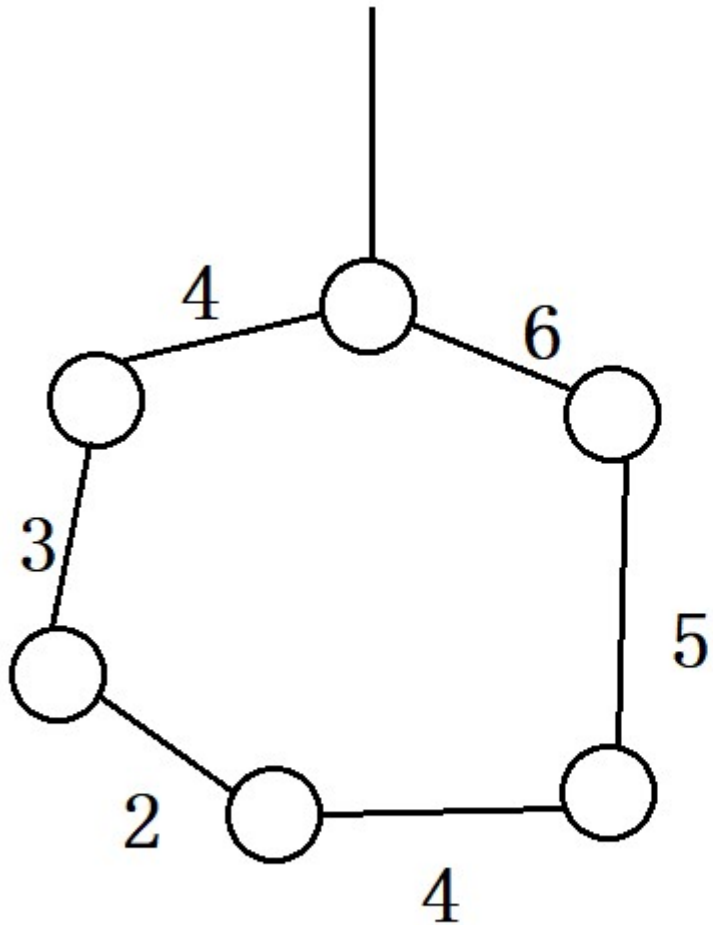
Solution

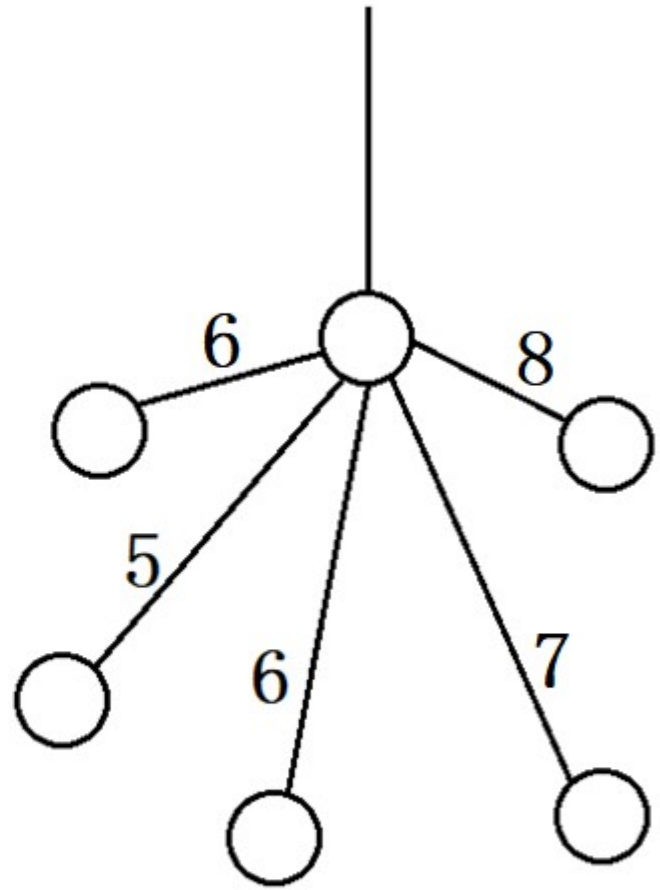
- 仙人掌两点间最大流即
- 初始流量 =inf
- 如果从一个环上经过，流量 min= 环上两边的 min 的和
- 如果从一条割边上经过，流量 min= 割边权值

Solution

- 同样建立一棵树，方法稍微改改，每个点 y 到环上深度最小的 x 点连边，边权为 x 和 y 间的最大流，即从 x 和 y 切开两个半环的 min 的和

Solution





Solution

- 如果 x 和 y 在同一个环上，把环倍长为一个序列，即变成查询 O(1) 次 rmq，然后比较一下
- 如果 x 和 y 不在同一个环上，和前面的最短路题类似的方法处理就行了

- 总时间复杂度 O(n)

Break

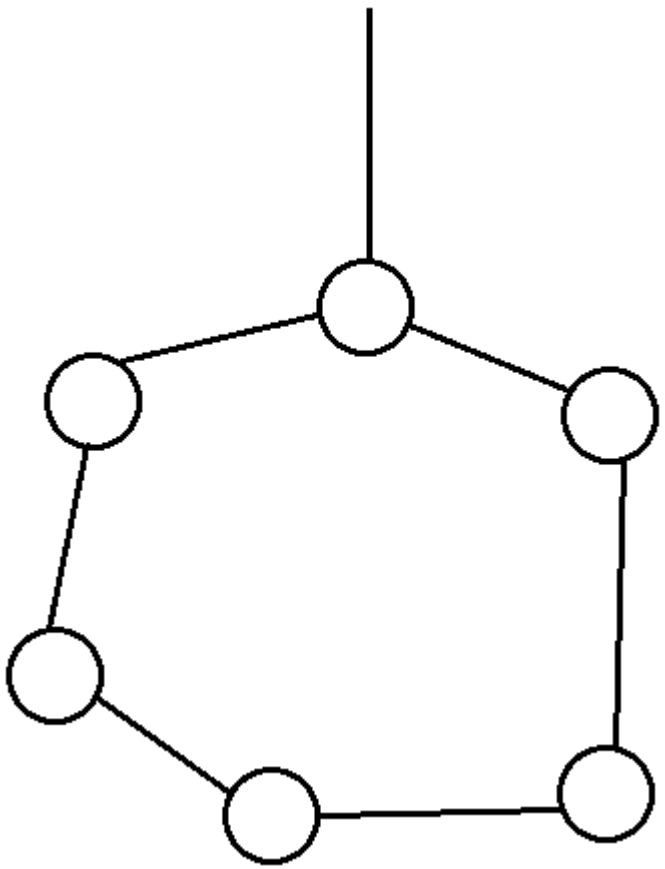
- 课间休息

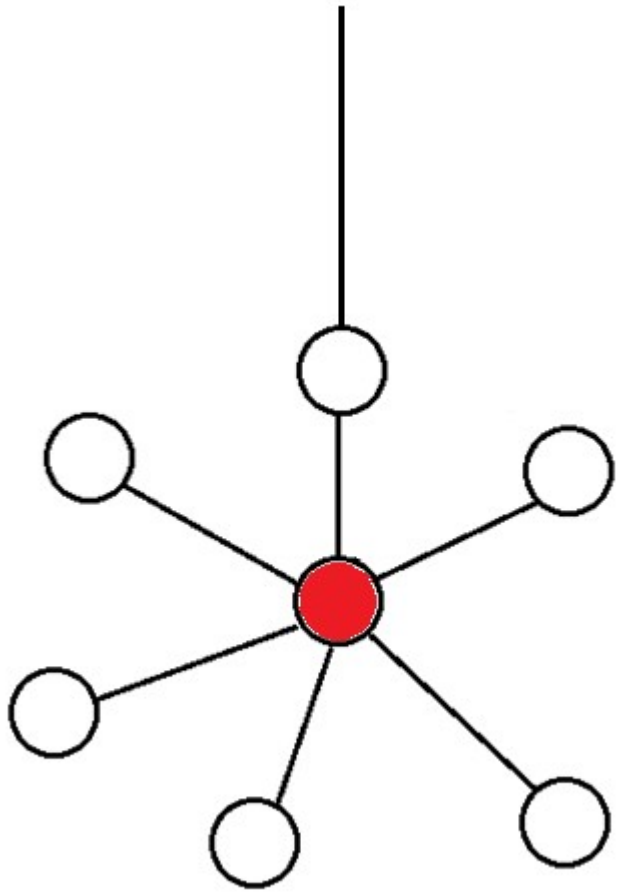
静态仙人掌分治

仙人掌点分治

- 还是考虑建一棵保留有仙人掌一定性质的树出来
- 我们对每个环建一个虚点，虚点向每个环上节点 `y` 连边，边权为 `dist(x,y)` ，并且删掉原始环

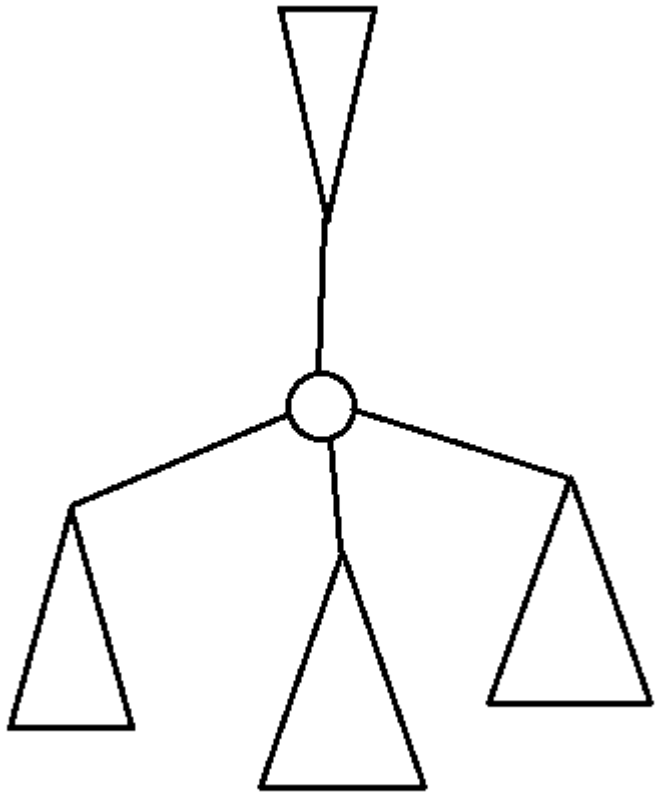
仙人掌点分治





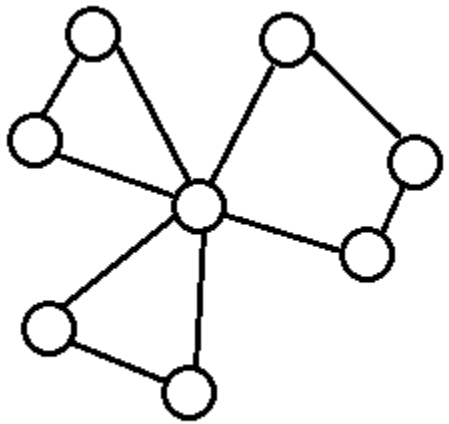
仙人掌点分治

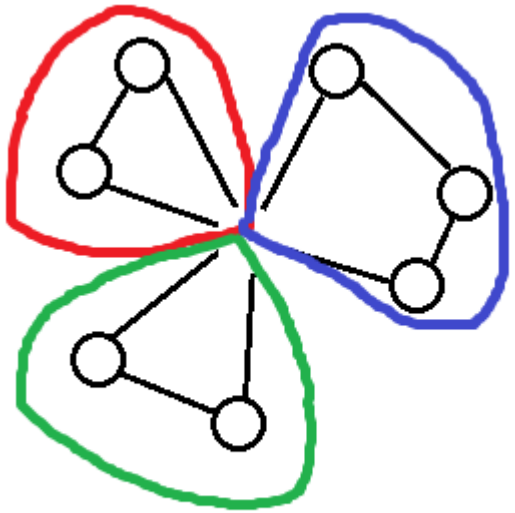
- 然后对这个建的树点分治，每次还是找一个分治重心
- 如果分治重心是一个普通的点，则和普通的点分治一样



仙人掌点分治

- 如果分治重心是一个环上的点，则相当于删掉了一些环上的一个点，这样分治下去合并上来的时候需要把这个点和其在的所有环上的信息合并

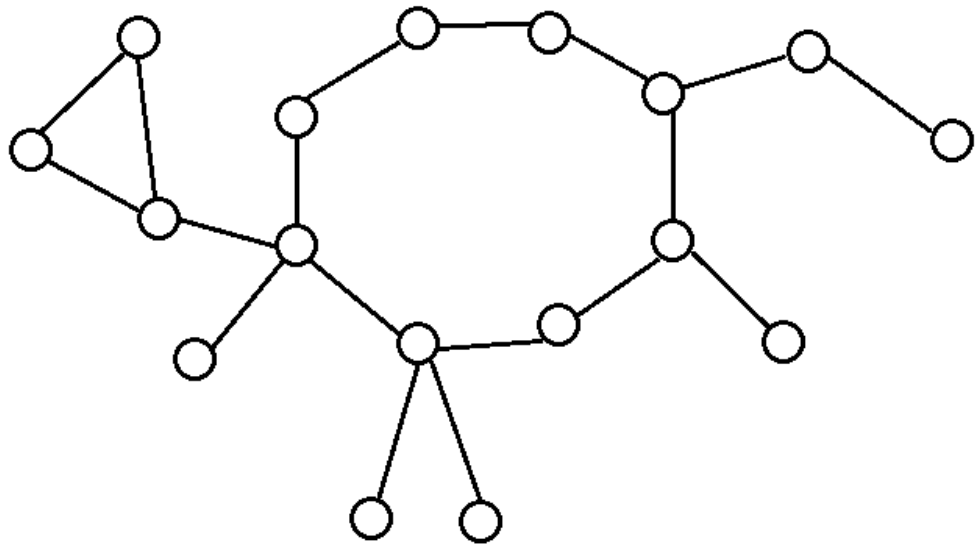


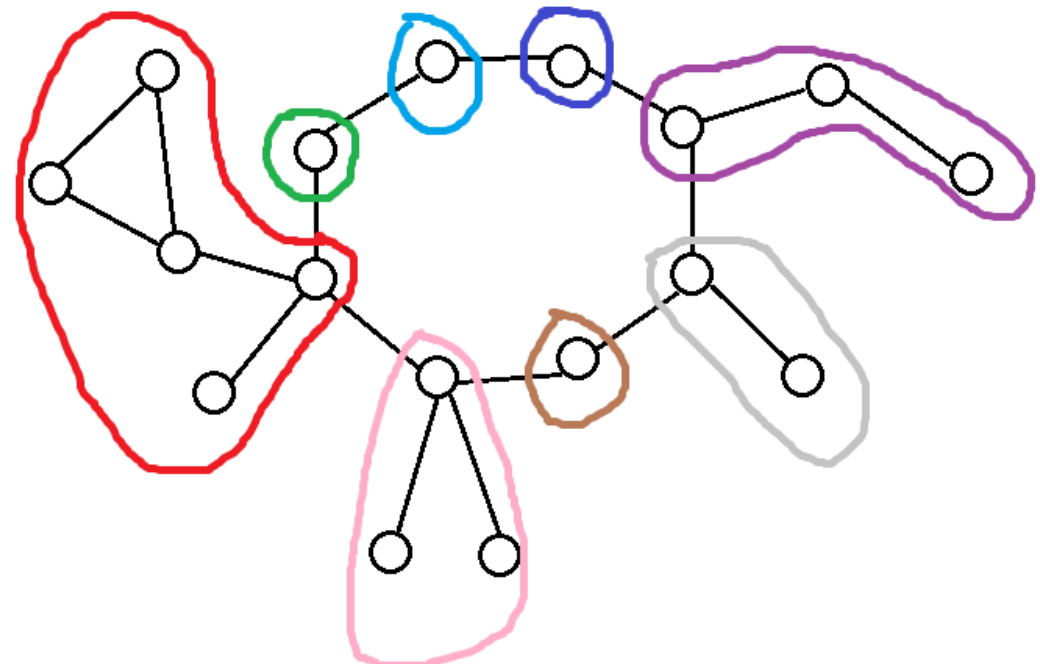


仙人掌点分治

- 如果分治重心是一个新建的虚点，则相当于我们把一个环拆开，每个点分别分治下去，然后再合并上来
- 拆开环的过程可以看做对一个序列进行分治，和线段树差不多
- 当然我们一般也可以直接把环上每个点拆出来，然后一个一个合并起来

仙人掌点分治





仙人掌分治

- 仙人掌分治的复杂度和点分治一样，是O(nlogn)的

// 仙人掌边分治

// 仙人掌链分治

仙人掌分治

- 使用仙人掌分治可以简单地解决前面讲到的那些仙人掌上 dp 的题目，不过复杂度可能多个 log

UOJ23. 【UR #1】跳蚤国王下江南

- 给你一棵 n 个节点的边仙人掌
- 对于所有的 $L(1\le L\le n-1)$，求出有多少不同的从节点 1 出发的包含 L 条边的简单路径。
- 简单路径是指不重复经过任意一点。

Solution

- 仙人掌分治，发现信息是一个多项式，每次合并的时候相当于做一个卷积
- 使用重量平衡的仙人掌分治结构可以做到 O(nlog^2n)，如果结构不好可能退化到 O(nlog^3n)

动态仙人掌

经典的动态树问题

- 给一棵树，支持
- Link
- Cut
- Modify
- 链信息查询

经典的动态仙人掌问题

- 给一棵仙人掌，支持
- Link
- Cut
- Modify
- 链信息查询
- 这里的链的定义一般有三种，就是最开始讲的那三种，其中两种基本相同

动态仙人掌

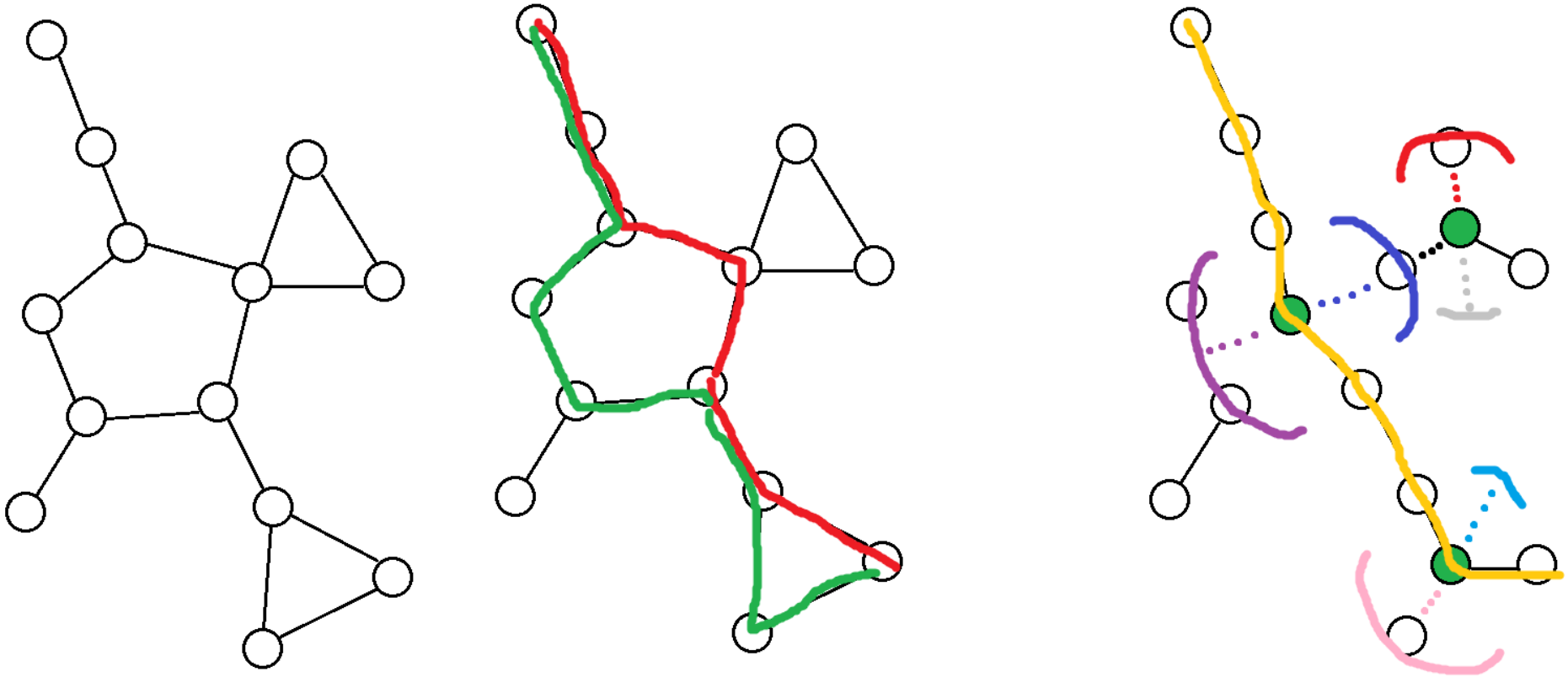
- 主要讲讲如何使用 `lct` 来维护动态仙人掌
- 相比于动态树问题，动态仙人掌问题复杂在环

基于点的动态仙人掌

- 在 lct 上，对于每个环，我们开一个新的虚点来维护
- 虚点向两个半环连一条特殊的边
- access 的时候如果经过了这个环，则我们用这个虚点来连接重链
- access 会导致半环发生变化
- 此时这个虚点上的信息变成我们这次 access 后这个环中统计入答案部分的信息

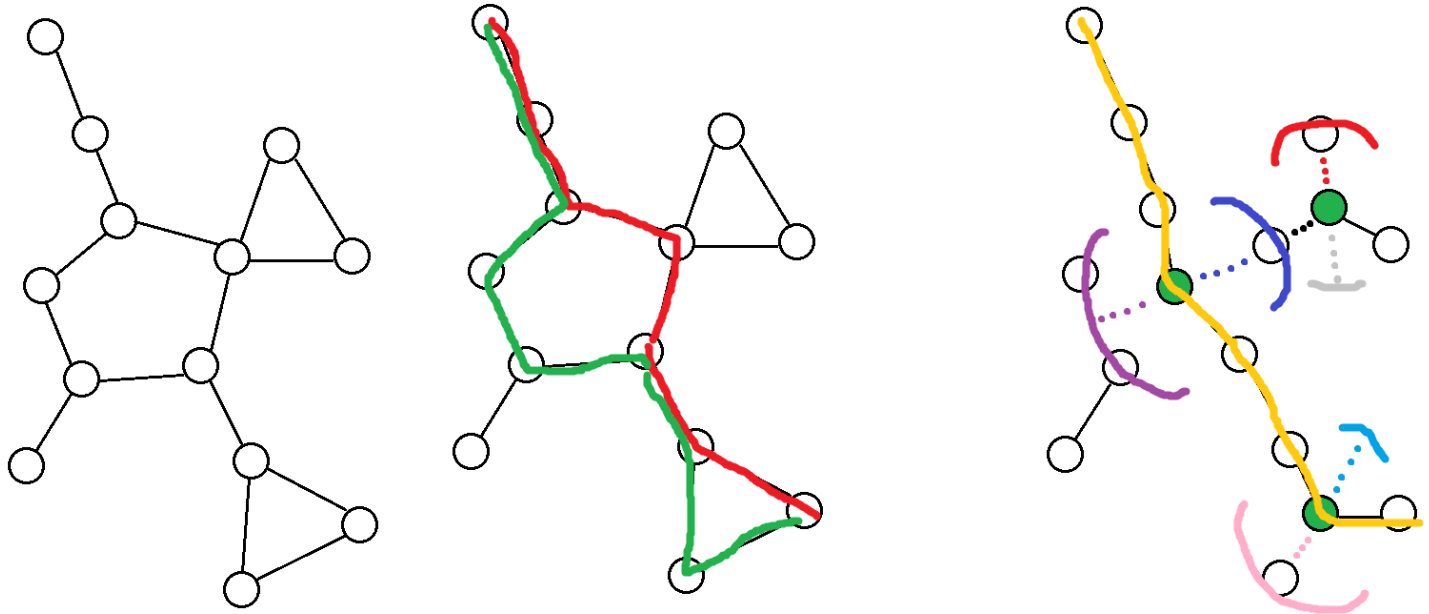
基于点的动态仙人掌

- 这里红色的部分是两点间最短简单路径，绿色的部分是两点间最长简单路径，二者的并是两点间所有简单路径的并



基于点的动态仙人掌

- 最短简单路径即黄色部分合并深蓝部分合并浅蓝部分
- 最长简单路径即黄色部分合并紫色部分合并粉色部分
- 简单路径的并即黄色部分合并深蓝部分合并浅蓝部分合并紫色部分合并粉色部分



// 基于边的动态仙人掌

复杂度

- 一般的动态仙人掌做法只能证明 O(log^2n) 均摊复杂度上界，但存在一些写法达到了 θ(logn) 确界
- 现实中二者的速度一般也差不到两三倍，所以不用担心被卡
- // 都出动态仙人掌了还卡常数?

UOJ65 动态仙人掌 III

- 有一天，VFleaKing 到森林里游玩，回来之后跟 pyx1997 说，我发现好多棵会动的树耶！pyx1997 说，这有什么好稀奇的，我用手指头就能维护每棵树的形态。
- 于是又过了几天 VFleaKing 到沙漠里游玩，回来之后跟 pyx1997 说，我发现好多棵会动的仙人掌耶！pyx1997 说，这有什么好稀奇的，我用脚丫子就能维护每棵仙人掌的形态。
- 于是 VFleaKing 很郁闷，他向你求助，请帮帮他吧。
- 如果一个无向连通图的任意一条边最多属于一个简单环，我们就称之为仙人掌。
- 如果一个无向图的每个连通块都是个仙人掌，且不存在自环，我们就称之为沙漠。

UOJ65 动态仙人掌 III

- 为了证明你确实能够维护仙人掌，我们给你 n 个结点，从 1 到 n 标号。初始时没有任何边。每次进行如下操作之一：

1. link $v\ u\ w_a\ w_b$：在结点 v,u 间连一条权值A为 w_a、权值B为 w_b 的边。
   - $1 \le v,u \le n$ 且 w_a,w_b 为正整数。
   - 如果连边完成后图仍为沙漠，则输出 "ok"（不含引号）。
   - 否则操作非法，撤销此次操作并输出 "failed"（不含引号）。
2. cut $v\ u\ w_a\ w_b$：在结点 v,u 间删掉一条权值A为 w_a、权值B为 w_b 的边。
   - $1 \le v,u \le n$ 且 w_a,w_b 为正整数。
   - 如果存在这样的边则输出 "ok"（不含引号）（如果有多条权值A为 w_a、权值B为 w_b 的边删去任意一条）。
   - 否则操作非法，不进行操作并输出 "failed"（不含引号）。
3. distance? $v\ u$：查询结点 v 到结点 u 的按权值A计算的最短路信息。
   - $1 \le v,u \le n$。
   - 输出两个用空格隔开的整数 l_m,w_m。
   - l_m 代表按权值A计算的最短路的长度，w_m代表最短路上的边的权值B的最小值。
   - 如果 $v=u$ 则 $l_m=0,w_m=2147483647$。
   - 如果没有路可到达则 $l_m=-1,w_m=-1$。
   - 如果最短路不唯一则 $w_m=-1$。
4. add $v\ u\ d$：把结点 v 到结点 u 的按权值A计算的最短路上的每一条边的权值B都加上 d。
   - $1 \le v,u \le n, v \ne u$ 且 d 为正整数。
   - 如果有路可到达且最短路唯一，则输出 "ok"（不含引号）
   - 否则操作非法，不进行操作并输出 "failed"（不含引号）。

Solution

- 裸题
- 注意到链加可能导致一个环中的最短路发生变化，不过用 /* 我 */ccz 用的那个写法就好写了，每次判一下把最小的一个半环的贡献加进去即可

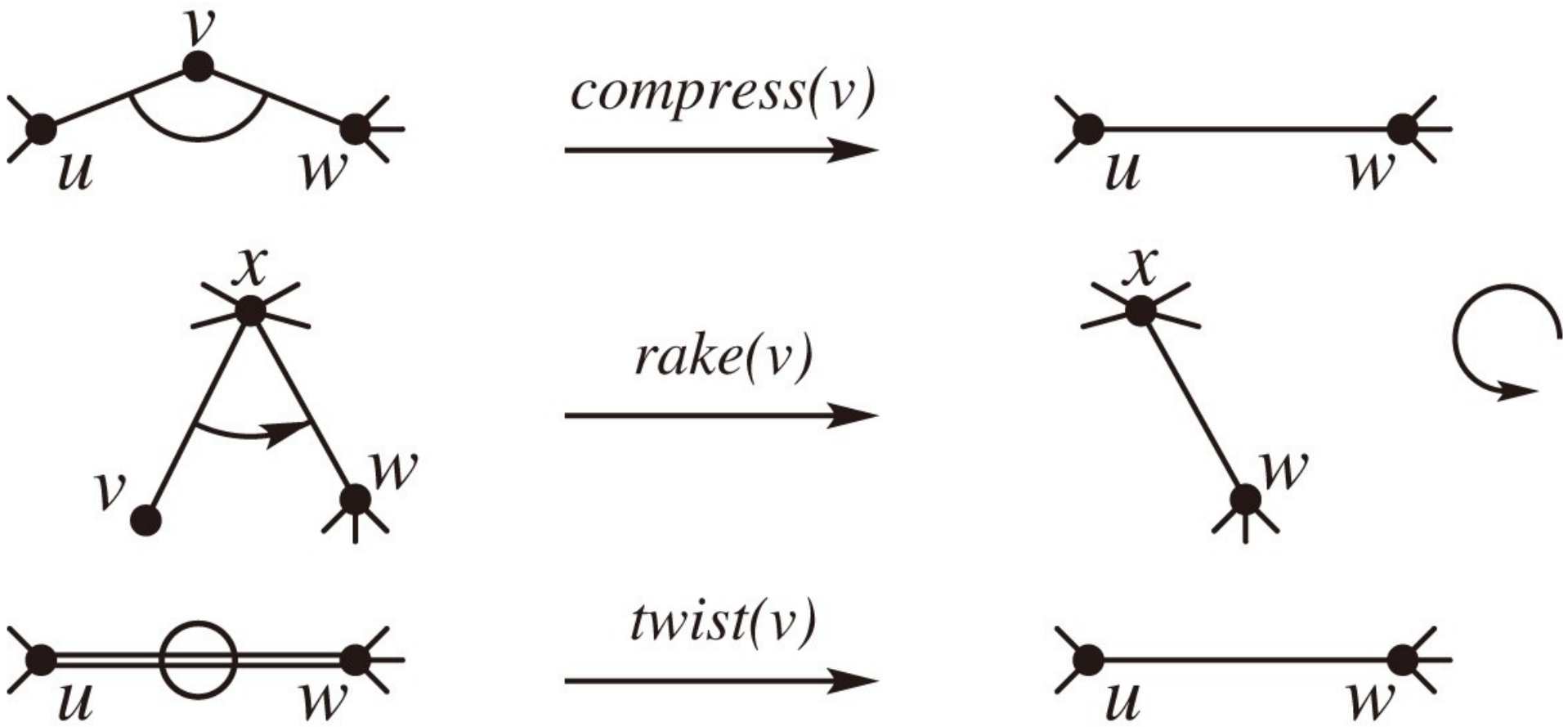
Top Cactus

- 以下部分摘抄自 negiizhao 的小文章

Top Cactus

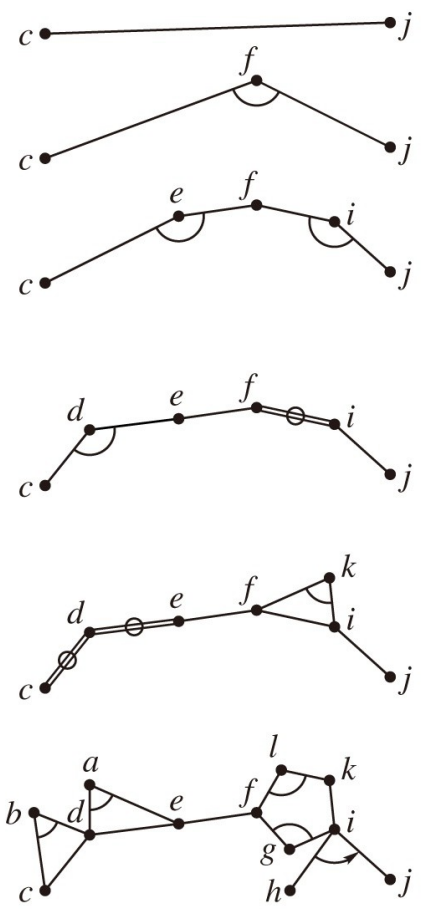
- 我们考虑将 top tree 推广用于“动态仙人掌问题”，即维护若干个不相交仙人掌组成的图，支持加边、删边、修改 / 查询两点之间 [最短路径 / 最长路径 / 所有路径的并] 或整个仙人掌上的信息.
- 既然想用在仙人掌上，就得先定义“仙人掌收缩”. 显然树收缩的操作是不够的，因为无法解决环的问题.
- 我们需要引入第三种操作，把两条端点相同的边合并为一条. 我把这操作称为 twist

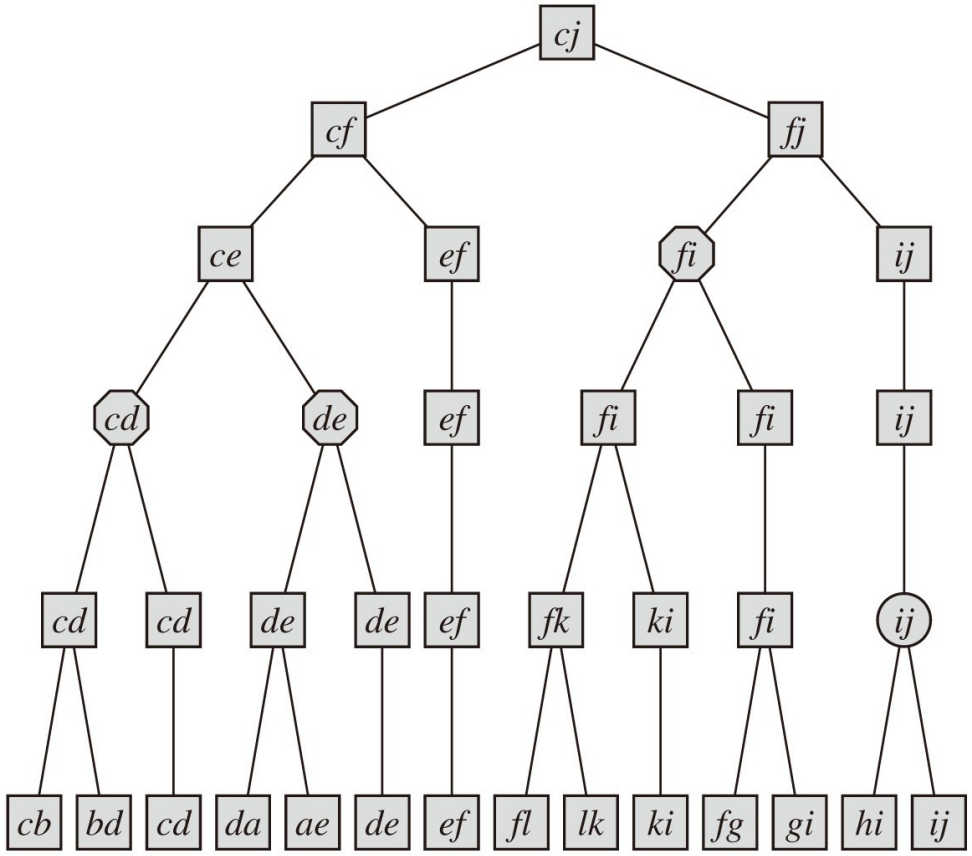
Top Cactus



Top Cactus

- 相应地，top tree 中也会出现一种结点：twist 结点，两个孩子表示原来的





“Top Cactus”

- 容易知道，这样仍能保证存在 O(logn) 层的收缩．很多信息仍然能在 twist 结点上统计，于是只要能用 O(logn) 时间完成 link 、 cut 、 expose ，那么很多动态树上的操作仍然可以在动态仙人掌进行．

- 下面给出一个 self-adjusting 版本的实现．
- **和 Tarjan 的论文一样，下面默认边按照某种顺序在点周围，并需要维护这个顺序．．实际中需要这个顺序的可能并不多，以下的某些东西可以大大简化．**

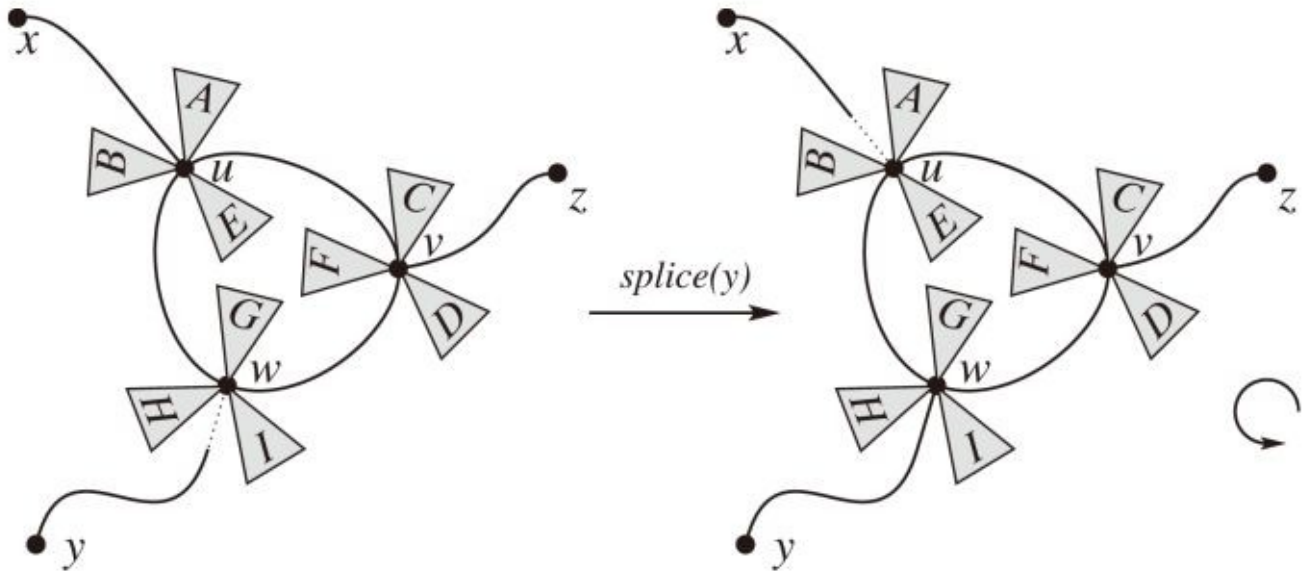
“Self Adjusting Top Cactus”

- 类似用于树的 self-adjusting top trees，我们对仙人掌进行剖分，但不是剖分为路径，而是某两点间所有路径的并.
- 对于这样“带环的路径”（这东西有没有什么名字啊），我们把每个环上离路径某个端点最近的点称为这个环的端点，环的两个端点把环分为两条边不相交的路径.
- 我们对环的两条路径分别进行 compress （用一个 compress tree 表示），再 twist 为一条边，这样整条“带环的路径”就被收缩为一条路径，直接用 compress tree 维护即可.

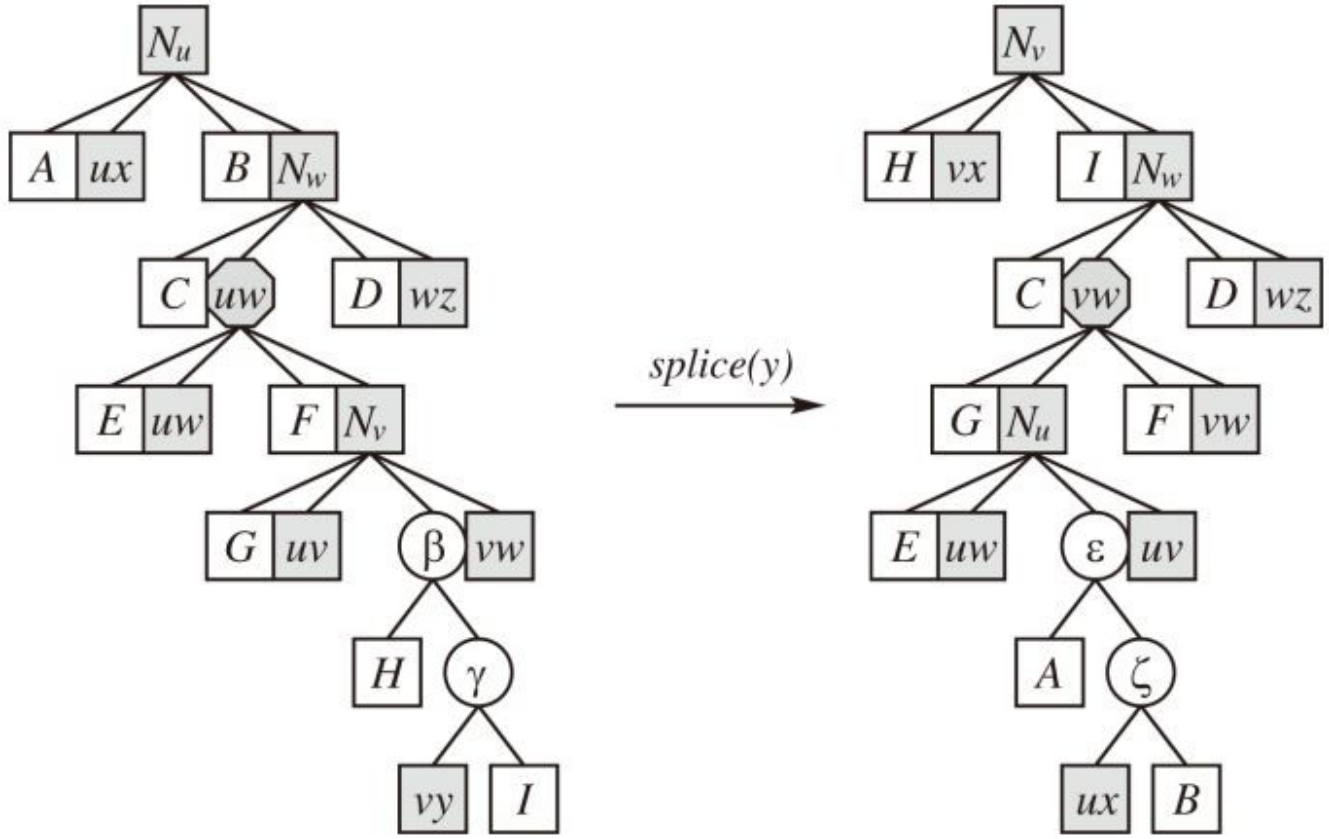
“Self Adjusting Top Cactus”

- 注意到如果要维护边的顺序，还需要处理环的两个端点在环内的子树．类似 compress，这可以在 twist 之前进行 rake，并成为 twist 结点的两个 foster child．然而，并不能 expose 环内和环外的顶点，因为 twist 的两条边之间不能有其他边，这只能通过将这些边 rake 到环上做到．
- Expose 操作和树上的情形相似，只需要再考虑如何解决 splice 到环上的情形．这种情形会改变环的端点，与非环上的 splice 相比，我们还需要先对环的两个原端点进行 local splay．

“Self Adjusting Top Cactus”



“Self Adjusting Top Cactus”



“Self Adjusting Top Cactus”

- 同样我们还需要考虑怎么进行和环相关的 link/cut 操作．显然可以先进行 soft_expose(u,w)
- 对于 link(v,w) 加入一个 twist 结点 (u,v) 和一个 base 结点 (u,v) ，对于 cut(v,w) 则相反．

- Splice 次数仍然是均摊 O(logn) 的，splay 的复杂度分析和树上的情形相似．
- 每次操作仍花费均摊 O(logn) 时间，常数略大一些，实际应用中通常也是不满的．

UOJ106. 动态仙人掌 IV

1. link $v\,u\,w$：在结点 v,u 间连一条权值为 w 的边。
    - $1\le v,u\le n$ 且 w 为正整数。
    - 如果连边完成后图仍为沙漠，则输出 "ok"（不含引号）。
    - 否则操作非法，撤销此次操作并输出 "failed"（不含引号）。
2. cut $v\,u\,w$：在结点 v,u 间删掉一条权值为 w 的边。
    - $1\le v,u\le n$ 且 w 为正整数。
    - 如果存在这样的边则输出 "ok"（不含引号）（如果有多条权值为 w 的边删去任意一条）。
    - 否则操作非法，不进行操作并输出 "failed"（不含引号）。
3. query1 $v\,u$：查询结点 v 到结点 u 的最短路信息。
    - $1\le v,u\le n$。
    - 输出两个用空格隔开的整数 $\min,\sigma$，分别代表最短路上点权的最小值、和。
    - 如果没有路可到达则 $\min=-1,\sigma=-1$。
    - 如果最短路不唯一则 $\min=-2,\sigma=-2$。
4. query2 $v\,u$：查询以结点 v 为根，子仙人掌 u 的信息。
    - $1\le v,u\le n$。
    - 以结点 v 为根，子仙人掌 u 的定义是，删掉 v 到 u 之间的所有简单路径上的边之后，u 所在的连通块。
    - 输出两个用空格隔开的整数 $\min,\sigma$，分别代表子仙人掌 u 中点权的最小值、和。
    - 如果 v,u 不连通则 $\min=-1,\sigma=-1$。
5. add1 $v\,u\,d$：把结点 v 到结点 u 的最短路上的每一个结点的权值都加上 d。
    - $1\le v,u\le n$ 且 d 为正整数。
    - 如果有路可到达且最短路唯一，则输出 "ok"（不含引号）
    - 否则操作非法，不进行操作并输出 "failed"（不含引号）。
6. add2 $v\,u\,d$：把以结点 v 为根，子仙人掌 u 的每一个结点的权值都加上 d。
    - $1\le v,u\le n$ 且 d 为正整数。
    - 如果 v,u 在同一个连通块里，则输出 "ok"（不含引号）
    - 否则操作非法，不进行操作并输出 "failed"（不含引号）。

Solution

- 裸题

UOJ158. 【清华集训 2015 】静态仙人掌

为了证明你确实能够维护仙人掌，我们给你 n 个结点，从 1 到 n 标号，其中 1 号点是仙人掌的根。它有 m 条边，第 i 条边连接了结点 u_i 与 v_i。

每个结点有一个颜色（黑或白），初始时均为黑色。现在有 q 次操作，每次操作格式为 $op\ x$（$1 \leq op \leq 3, 1 \leq x \leq n$）：

- 若$op = 1$，表示将点x到根的最短路径上的所有点的状态取反（黑变白，白变黑）；
- 若$op = 2$，表示将点x到根的最长简单路径上的所有点的状态取反；
- 若$op = 3$，表示询问点x的子仙人掌中的黑点数目。点 x 的子仙人掌定义为：删除从根到点 x 的所有简单路径上的所有边后，点 x 所在的连通块。

Solution

- 直接在静态的仙人掌分治上打标记即可维护，这里需要对最长路和最短路分别维护标记
- 然后由于信息可减，所以我们可以用 lct 维护子树的方法来维护子仙人掌信息，这里可以用静态 lct

- 总时间复杂度 O(n + mlogn)

例题讲解

成都七中　nzhtl1477

bzoj5473: 仙人掌

- 有一个 n 个点，m 个边的仙人掌。所谓仙人掌，就是任何一个点至多属于一个环。
- 每个边有 1/2 的概率被删掉。问期望剩下多少个边联通块。
- 所谓边联通块，就是问剩下的边，构成多少个联通块，单独一个点不算做联通块。
- 输出答案乘以 2^m 之后 mod 1000000007 的结果。

Solution

- 考虑如果仙人掌是一棵树的情况
- 如果考虑一个点也算一个连通块的话
- 此时考虑如果一条边都没有的话，答案会是 2^m*n
- 每加上一条边可以让连通块数量 -1，这部分的贡献是 -2^(m-1)*m
- m 条边，每条边必须选，其他 m-1 条边，每条边可以选和不选，
- 所以是 -m*2^(m-1)

Solution

- 对于仙人掌，环的出现导致其和树性质不同
- 考虑对于一个大小为 len 的环，如果环上所有边都选了，那其只减少了 (len-1) 个连通块
- 让这 len 条边必须选，剩下的 m-len 条边可以选或者不选，这部分是 2^(m-len) 的贡献
- 发现环上这样做会导致多减，加回来就可以了

Solution

- 然后我们还多算了大小为 1 的连通块的贡献
- 这里一个大小为 1 的连通块出现当且仅当其的边表中所有边都没选，对于一个度数为 deg 的点，这部分的贡献是 -2^(m-deg) 的
- 即我们这 deg 条边必须不选，其他边随便选
- 于是统计每个环大小，每个点度数即可

- 总时间复杂度 O(n)

洛谷 3180 [HAOI2016] 地图

- 这个都市有 n 个建筑，编号从 1 到 n，其中市中心编号为 1，这个都市有 m 条双向通行的街道，每条街道连接着两个建筑，其中某些街道首尾相连连接成了一个环。
- 这个都市的两个特点：1. 从市中心出发可以到达所有的建筑物。2. 任意一条街道最多存在与一个简单环中。每个建筑物都会有拉面售卖。拉面有很多不同的种类，但对于 rin 而言只有油腻程度的不同，因此我们把油腻程度相同的拉面看做同一种拉面。
- 由于不同建筑物的拉面的油腻程度可能不同，我们用一个正整数来表示拉面的油腻程度。
- 现在 rin 想知道，如果她正在编号为 x 的建筑物，那么在从市中心到 x 的所有简单路径经过的街道都被堵死的情况下，rin 可以品尝到的拉面中（注意没有出现的拉面是不能算在里面的）：1. 油腻程度 <= y 且品尝次数为奇数次的拉面有多少种？2. 油腻程度 <= y 且品尝次数为偶数次的拉面有多少种?

Solution

- 不知道为什么这题一堆人在写莫队，明明是 1log 的
- 这个就是要求一个子仙人掌的信息
- 考虑树上该如何维护
- 我们可以使用静态链分治来简单做出来
- 维护子树的一个数据结构，支持：
- 插入，查询出现奇数次的 <=x 的数和，查询出现偶数次的 <=x 的数和

Solution

- 这样的数据结构使用一个平衡树就可以了，对于出现次数奇数和偶数的数分别维护
- 每次合并上来的时候就按照静态链分治的方法，保留重儿子的数据结构，将轻儿子的数据结构启发式合并进来
- 由于平衡树启发式合并的复杂度是 O(nlogn) 的，所以总时间复杂度 O((n+m)logn)

Solution

- 在仙人掌上，我们可以对仙人掌的 DFS 树采取类似的方法做，每次子仙人掌就是 DFS 树的一个子树加上 O(1) 个节点

- 总时间复杂度 O((n+m)logn)

洛谷 3687 [ZJOI2017] 仙人掌

- 现在九条可怜手上有一张**无自环无重边的无向连通图**，但是她觉得这张图中的边数太少了，所以她想要在图上连上一些新的边。同时为了方便的存储这张无向图，图中的边数又不能太多。经过权衡，她想要加边后得到的图为一棵仙人掌。
- 不难发现合法的加边方案有很多，可怜想要知道总共有多少不同的加边方案。
- 两个加边方案是不同的当且仅当一个方案中存在一条另一个方案中没有的边。

Solution

- 我们先考虑只有一棵树如何处理．
- 仙人掌可以看做若干环的集合． 特别的， 对于一条没有环的边， 可以加上重边， 那么这个边和它的重边构成一个环．
- 对于树来说， 问题就可以转化为求加上若干条边， 使树上的每一条边在且仅在一个环内的方案数．
- 去掉加的边， 也就是说求用若干条边不相交的链将整个树覆盖的方案数．

Solution

- 考虑树形 dp.
- 设 f[i] 表示考虑 i 的子树与 i 连向父亲的边，用若干条边不相交的链覆盖的方案数；
- g[i] 表示一个点连出 i 条边，用若干条边不相交的链覆盖的方案数，也就是说，将 i 条边两两匹配或者单独留下的方案数.
- 考虑最后一条边是否匹配，我们可以得出 g[i] 的递推公式：
- g[i] = g[i - 1] + g[i - 2] * (i - 1)

Solution

然后求 f_i:

对于非根的点 i, 它连出了 $|child(i)|+1$ 条边. 可以考虑将 f_j 连向父亲的边两两匹配或者单独留下, 根据乘法原理, 有

$$f_i = \prod_{j \in child(i)} f_j \cdot g_{|child(i)|+1}$$

对于$i = rt$, 它没有连向父亲的边, 因此

$$f_i = \prod_{j \in child(i)} f_j \cdot g_{|child(i)|}$$

答案即为 f_{rt}.

- 总时间复杂度 O(n)

Solution

- 最后考虑其他的图怎么做：
- 如果不是仙人掌， 答案为0;
- 如果图是仙人掌：
- 对于仙人掌的一个环上的两点 p 和 q, 显然不能再加边使它们在环外联通． 因此， 我们可以去掉所有的环， 对于剩下的每棵树分别求出答案， 对答案相乘即可．

- 总时间复杂度O(n)

UOJ87. mx 的仙人掌

- 现给定一棵仙人掌，每条边有一个正整数权值，每次给 k 个点（可以存在相同点），问从它们中选出两个点（可以相同），它们之间最短路的最大值是多少。

Solution

- 可以使用动态仙人掌，或者静态的仙人掌分治结构，每次插入需要的那些点，维护的信息是最短的两点间路径的分治信息
- 也可以每次建出来这些点的虚仙人掌，这里和虚树类似，考虑一下如何保持仙人掌原来的结构就可以了
- 总时间复杂度 $O(n + m\log n)$

Thanks for listening

// 成都七中 nzhtl1477