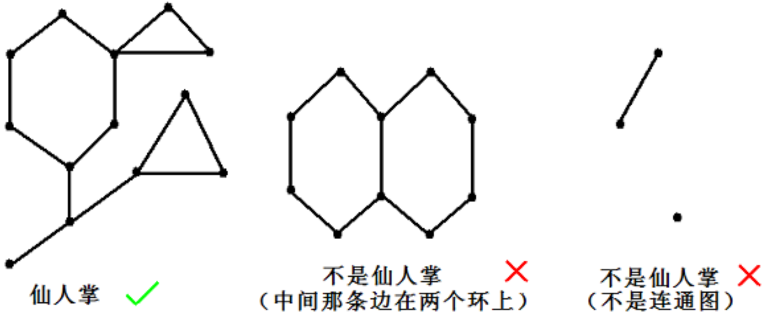
仙人掌

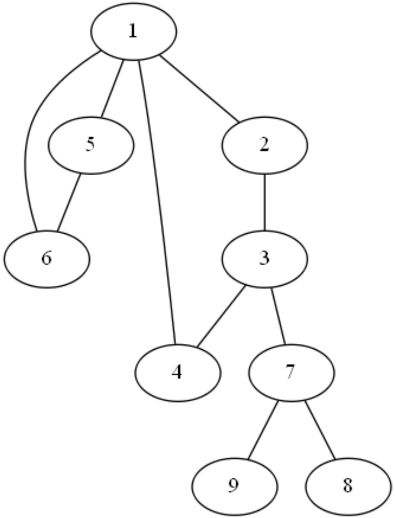
___debug

2018 年 1 月 4 日

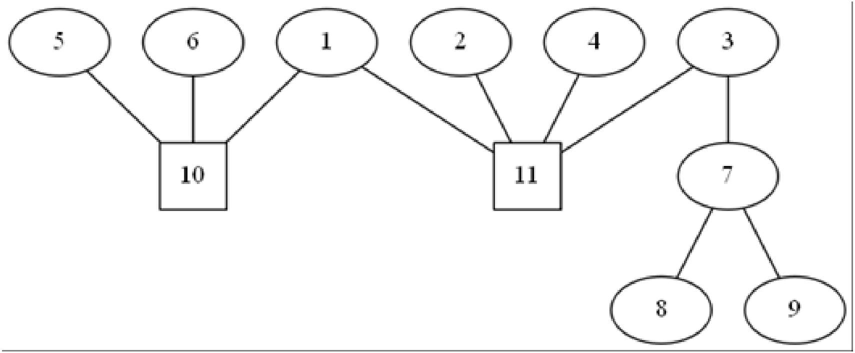
仙人掌: 定义



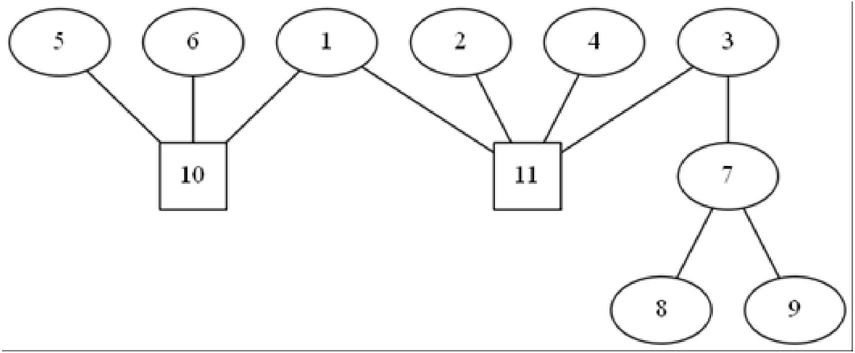
圆方树: 定义



圆方树: 定义



圆方树: 定义



可以用 tarjan 求点双的时候顺便求出来.

圆方树：性质

1. 形态与根的选取无关

圆方树: 性质

1. 形态与根的选取无关
2. 子树对应原图的子仙人掌

圆方树：性质

1. 形态与根的选取无关
2. 子树对应原图的子仙人掌
3. 方点和方点不会直接相连

圆方树: 应用

BZOJ 2125

仙人掌带权最短路.

圆方树: 应用

BZOJ 2125 - 题解

首先给圆方树定一个根, 这样每个方点一定有一个圆点父亲, 将这个圆点称为这个环的根.

首先给圆方树定一个根, 这样每个方点一定有一个圆点父亲, 将这个圆点称为这个环的根.

树上的边权怎么定?

首先给圆方树定一个根, 这样每个方点一定有一个圆点父亲, 将这个圆点称为这个环的根.

树上的边权怎么定?
对于每个环, 圆点到方点的权值为这个点到环根的最短距离 (顺时针/逆时针).

圆方树: 应用

BZOJ 2125 - 题解

首先给圆方树定一个根, 这样每个方点一定有一个圆点父亲, 将这个圆点称为这个环的根.

树上的边权怎么定?
对于每个环, 圆点到方点的权值为这个点到环根的最短距离 (顺时针/逆时针).

直接在圆方树上求距离还不行, 因为 LCA 为方点时会出问题.

首先给圆方树定一个根, 这样每个方点一定有一个圆点父亲, 将这个圆点称为这个环的根.

树上的边权怎么定?
对于每个环, 圆点到方点的权值为这个点到环根的最短距离 (顺时针/逆时针).

直接在圆方树上求距离还不行, 因为 LCA 为方点时会出问题.
不过也很好解决, 倍增求 LCA 的时候可以求出是从哪个点上来的.
一开始预处理出每个点在环里的位置和每个环的距离前缀和, 直接取个 $\min$ 就可以了.

圆方树: 应用

BZOJ 4316

仙人掌最大独立集.

圆方树：应用

BZOJ 4316 - 题解

类比树上最大独立集的做法.

圆方树: 应用

BZOJ 4316 - 题解

类比树上最大独立集的做法.

设状态 $dp(u, 0/1)$ 表示考虑了以 u 为根的子树, 并且 u 没选/选了, 此时的最大独立数. 对于圆方树上的情况, 我们类似地设状态, 只有在 u 为方点的时候需要特殊考虑.

圆方树: 应用

BZOJ 4316 - 题解

类比树上最大独立集的做法.

设状态 $dp(u,0/1)$ 表示考虑了以 u 为根的子树, 并且 u 没选/选了, 此时的最大独立数. 对于圆方树上的情况, 我们类似地设状态, 只有在 u 为方点的时候需要特殊考虑.
然后方点也是跑一个环上的 $f(i,0/1,0/1)$ 的 DP 就可以了.

圆方树: 应用

BZOJ 1023

仙人掌直径.

圆方树: 应用

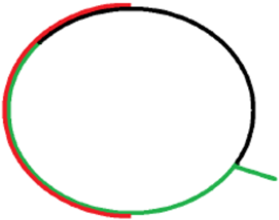
BZOJ 1023 - 题解

两次 bfs?

圆方树: 应用

BZOJ 1023 - 题解

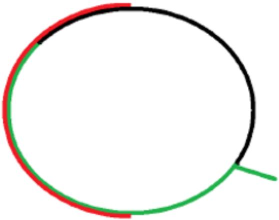
两次 bfs?
这个结论在仙人掌上显然是不对的...



圆方树: 应用

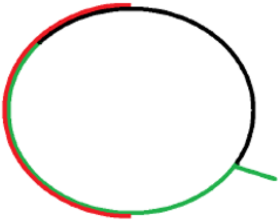
BZOJ 1023 - 题解

两次 bfs?
这个结论在仙人掌上显然是不对的...



不过可以利用另一种树上求直径的方法. 设 $dp(u)$ 表示从 u 能往下走的最大距离, 这个无论是圆点方点都很好维护.

两次 bfs?
这个结论在仙人掌上显然是不对的...



不过可以利用另一种树上求直径的方法. 设 $dp(u)$ 表示从 u 能往下走的最大距离, 这个无论是圆点方点都很好维护.
注意方点更新答案需要用到单调队列.

圆方树: 应用

UOJ 87

给出一个点集, 求点集中两两最短路的最大值.

圆方树: 应用

UOJ 87 - 题解

看完了之前的三道题, 想必大家对圆方树已经很熟悉了吧.

UOJ 87 - 题解

看完了之前的三道题, 想必大家对圆方树已经很熟悉了吧.
这道题只用建一个虚仙人掌就可以了.

圆方树: 应用

UOJ 87 - 题解

看完了之前的三道题, 想必大家对圆方树已经很熟悉了吧.
这道题只用建一个虚仙人掌就可以了.

注意可能会出现一个方点在虚树上的儿子不是真实儿子的情况, 这种情况需要特殊考虑.

圆方树: 应用

UOJ 87 - 题解

看完了之前的三道题, 想必大家对圆方树已经很熟悉了吧.
这道题只用建一个虚仙人掌就可以了.

注意可能会出现一个方点在虚树上的儿子不是真实儿子的情况, 这种情况需要特殊考虑.

不过也很好做, 只需在倍增的时候找出是从哪个点爬上来的就行了.

圆方树: 应用

UOJ 87 - 题解

看完了之前的三道题, 想必大家对圆方树已经很熟悉了吧.
这道题只用建一个虚仙人掌就可以了.

注意可能会出现一个方点在虚树上的儿子不是真实儿子的情况, 这种情况需要特殊考虑.

不过也很好做, 只需在倍增的时候找出是从哪个点爬上来的就行了.
然后就是和前一题类似的 DP 和单调队列.

圆方树: 应用

UOJ 23

求仙人掌上从 1 出发长度为 $1 \ldots n$ 的简单路径条数.

首先考虑一个特殊情况: 存在一个生成树是一条 $1...n$ 的链.

圆方树: 应用

UOJ 23 - 题解

首先考虑一个特殊情况: 存在一个生成树是一条 $1 \ldots n$ 的链.

因为仙人掌的性质, 所以非树边代表的区间一定是不相交的. 对于一条没有被非树边覆盖的边, 我们把它当作多项式 x; 否则, 如果是一个长度为 s 的环, 我们就当作是多项式 $x + x^{s-1}$.

首先考虑一个特殊情况: 存在一个生成树是一条 $1...n$ 的链.

因为仙人掌的性质, 所以非树边代表的区间一定是不相交的. 对于一条没有被非树边覆盖的边, 我们把它当作多项式 x; 否则, 如果是一个长度为 s 的环, 我们就当作是多项式 $x + x^{s-1}$.
然后直接分治一下就好了, 时间复杂度是 $O(n \log^2 n)$ 的.

圆方树: 应用

UOJ 23 - 题解

对于原问题, 我们考虑在圆方树上进行点分治.

圆方树: 应用

UOJ 23 - 题解

对于原问题, 我们考虑在圆方树上进行点分治.

设当前子树中重心为 c, 我们要求的是这棵子树的最高点 $top(c)$ 往下走的方案数的多项式.

圆方树: 应用

UOJ 23 - 题解

对于原问题, 我们考虑在圆方树上进行点分治.

设当前子树中重心为 c, 我们要求的是这棵子树的最高点 $top(c)$ 往下走的方案数的多项式. 显然包含 $top(c)$ 这一个子树需要分开考虑. 不妨先考虑 c 的其它子树:

圆方树: 应用

UOJ 23 - 题解

对于原问题, 我们考虑在圆方树上进行点分治.

设当前子树中重心为 c, 我们要求的是这棵子树的最高点 $top(c)$ 往下走的方案数的多项式. 显然包含 $top(c)$ 这一个子树需要分开考虑. 不妨先考虑 c 的其它子树:

- 如果 c 是圆点, 直接将子树的 f 求和即可

对于原问题, 我们考虑在圆方树上进行点分治.

设当前子树中重心为 c, 我们要求的是这棵子树的最高点 $top(c)$ 往下走的方案数的多项式. 显然包含 $top(c)$ 这一个子树需要分开考虑. 不妨先考虑 c 的其它子树:

- 如果 c 是圆点, 直接将子树的 f 求和即可
- 如果 c 是方点, 也只需考虑每个子树顺时针/逆时针两种情况, 直接求和即可

对于原问题, 我们考虑在圆方树上进行点分治.

设当前子树中重心为 c, 我们要求的是这棵子树的最高点 $top(c)$ 往下走的方案数的多项式. 显然包含 $top(c)$ 这一个子树需要分开考虑. 不妨先考虑 c 的其它子树:

- 如果 c 是圆点, 直接将子树的 f 求和即可
- 如果 c 是方点, 也只需考虑每个子树顺时针/逆时针两种情况, 直接求和即可

现在我们需要求出这样一个多项式:

圆方树: 应用

UOJ 23 - 题解

对于原问题, 我们考虑在圆方树上进行点分治.

设当前子树中重心为 c, 我们要求的是这棵子树的最高点 $top(c)$ 往下走的方案数的多项式. 显然包含 $top(c)$ 这一个子树需要分开考虑. 不妨先考虑 c 的其它子树:

- 如果 c 是圆点, 直接将子树的 f 求和即可
- 如果 c 是方点, 也只需考虑每个子树顺时针/逆时针两种情况, 直接求和即可

现在我们需要求出这样一个多项式: 起点为 $top(c)$, 终点为 c 的简单路径的方案数.

圆方树: 应用

UOJ 23 - 题解

这个可以利用之前的分治 $O(n\log^2 n)$ 求出来，但是还有一种更好的方法.

这个可以利用之前的分治 $O(n\log^2 n)$ 求出来, 但是还有一种更好的方法.

对于每个分治重心 u, 我们都维护 u 到 $top(u)$ 的简单路径方案数.

圆方树: 应用
UOJ 23 - 题解

这个可以利用之前的分治 $O(n\log^2 n)$ 求出来, 但是还有一种更好的方法.

对于每个分治重心 u, 我们都维护 u 到 $top(u)$ 的简单路径方案数. 然后考虑从 c 跳到 $top(c)$, 在途中顺便把多项式乘起来. 可以发现, 这一部分的时间复杂度为 $T(n) = T(\frac{n}{2}) + O(n\log n) = O(n\log n)$.

这个可以利用之前的分治 $O(n\log^2 n)$ 求出来, 但是还有一种更好的方法.

对于每个分治重心 u, 我们都维护 u 到 $top(u)$ 的简单路径方案数. 然后考虑从 c 跳到 $top(c)$, 在途中顺便把多项式乘起来. 可以发现, 这一部分的时间复杂度为 $T(n) = T(\frac{n}{2}) + O(n\log n) = O(n\log n)$.

注意由于方点的多项式跟环长有关, 所以需要求带权重心, 方点的权值为环的长度.

圆方树: 应用

UOJ 23 - 题解

这个可以利用之前的分治 $O(n\log^2 n)$ 求出来, 但是还有一种更好的方法.

对于每个分治重心 u, 我们都维护 u 到 $top(u)$ 的简单路径方案数. 然后考虑从 c 跳到 $top(c)$, 在途中顺便把多项式乘起来. 可以发现, 这一部分的时间复杂度为 $T(n) = T(\frac{n}{2}) + O(n\log n) = O(n\log n)$.

注意由于方点的多项式跟环长有关, 所以需要求带权重心, 方点的权值为环的长度.
总时间复杂度为 $O(n\log^2 n)$.

圆方树: 应用

还有一些应用:

- 仙人掌同构 $\rightarrow$ 圆方树同构

圆方树: 应用

还有一些应用:

- 仙人掌同构 $\to$ 圆方树同构
- 动态仙人掌 $\to$ 动态圆方树

圆方树: 应用

还有一些应用:

- 仙人掌同构 $\to$ 圆方树同构
- 动态仙人掌 $\to$ 动态圆方树
- …