《过去的集合》命题报告

厦门双十中学 汪文潇

2016 年 4 月 29 日

目录

- 答辩概要
- 问题大意
- 维护森林法
- 按秩合并的应用
- 倍增法的应用

答辩概要

不相交集合的合并及查询是一个经典问题，也即常说的并查集，以下简称为并查集问题。
在命题报告中，对该问题进行了一个简单的扩展，并介绍了一些能够适应扩展后情况的解法，达到了相对较优的效果。
这里，将选取其中较为重要的一部分内容进行介绍、解释，希望能带来一定帮助。

答辩概要

不相交集合的合并及查询是一个经典问题，也即常说的并查集，以下简称为并查集问题。
在命题报告中，对该问题进行了一个简单的扩展，并介绍了一些能够适应扩展后情况的解法，达到了相对较优的效果。
这里，将选取其中较为重要的一部分内容进行介绍、解释，希望能带来一定帮助。

问题大意

问题大意

有 n 个元素，最初每个元素属于单独的一个集合，有 m 次操作，每次操作都是以下两种之一：

1、将两个集合进行合并

2、询问在过去的某个时间点，某两个元素是否在同一个集合

简而言之，该问题即是一个要求支持对历史版本进行询问的并查集问题。

问题大意

有 n 个元素，最初每个元素属于单独的一个集合，有 m 次操作，每次操作都是以下两种之一：

1、将两个集合进行合并

2、询问在过去的某个时间点，某两个元素是否在同一个集合

简而言之，该问题即是一个要求支持对历史版本进行询问的并查集问题。

注：为了更好地表现该报告的想法和目的，这里的问题描述与题目《过去的集合》形式不尽相同。

维护森林法

维护森林法在并查集问题中是很常见的，在竞赛中经常使用的路径压缩和按秩合并的方法都是对其的优化。

维护森林法

维护森林法在并查集问题中是很常见的，在竞赛中经常使用的路径压缩和按秩合并的方法都是对其的优化。

维护森林法即是用森林来表示当前的集合状态，图中的每棵树就对应一个独立的集合。

维护森林法

维护森林法在并查集问题中是很常见的，在竞赛中经常使用的路径压缩和按秩合并的方法都是对其的优化。

维护森林法即是用森林来表示当前的集合状态，图中的每棵树就对应一个独立的集合。

存储一个森林的结构，可以直接存储每个节点的父节点。

询问操作只需要判断两个点所在树的根即可。

合并时就将一棵树的根的父节点设成另一棵树的根。

维护森林法

维护森林法在并查集问题中是很常见的，在竞赛中经常使用的路径压缩和按秩合并的方法都是对其的优化。

维护森林法即是用森林来表示当前的集合状态，图中的每棵树就对应一个独立的集合。

存储一个森林的结构，可以直接存储每个节点的父节点。

询问操作只需要判断两个点所在树的根即可。

合并时就将一棵树的根的父节点设成另一棵树的根。

如果不考虑询问历史版本，那么显然朴素实现单次询问的复杂度最坏是 $O(n)$ 的，合并的总复杂度最坏可以达到 $O(n^2)$。

分析性质

在实现对历史版本的询问以前，我们首先要考虑一些性质。

分析性质

在实现对历史版本的询问以前，我们首先要考虑一些性质。

Theorem (1)

过去任意时刻的森林的边集一定是当前边集的子集。

分析性质

在实现对历史版本的询问以前，我们首先要考虑一些性质。

Theorem (1)

过去任意时刻的森林的边集一定是当前边集的子集。

证明是显然的，因为操作过程中边不会被删除。

分析性质

分析性质

Theorem (2)

如果设当前存在的每一条边的权值为其出现的时刻，那么任意一个点到根的路径上边权都是单调增的。

Theorem (2)

如果设当前存在的每一条边的权值为其出现的时刻，那么任意一个点到根的路径上边权都是单调增的。

可以通过归纳法证明。
首先显然图中没有边时该性质是成立的。
假设当前局面该性质成立，再一次合并两棵树后，新加的边一定连接原来某两棵树的根节点。
由于新加的边是最晚出现的，其权值最大，且原来这两颗树内的所有点到根的路径上边权单调增，所以合并出的新树内每个点到根的路径上边权仍然单调增。
对于其他树内的点，显然仍然满足该性质。
这样就证明了定理 2。

按秩合并的应用

按秩合并的应用

按秩合并即给每棵树一个秩，只有一个点的树秩为 0：当合并两个树时，如果两棵树的秩不同，将秩较小的树的根作为另一棵树的根的子节点，新树的秩设为原来两棵树秩的较大值；如果两棵树的秩相同，任意将一棵的根作为另一棵树的根的子节点，新树的秩设为原来两棵树的秩加一。

按秩合并的应用

按秩合并即给每棵树一个秩，只有一个点的树秩为 0：当合并两个树时，如果两棵树的秩不同，将秩较小的树的根作为另一棵树的根的子节点，新树的秩设为原来两棵树秩的较大值；如果两棵树的秩相同，任意将一棵的根作为另一棵树的根的子节点，新树的秩设为原来两棵树的秩加一。

不难发现，秩其实等价于一棵树内离根最远的点到根的距离。

按秩合并的应用

按秩合并即给每棵树一个秩，只有一个点的树秩为 0；当合并两个树时，如果两棵树的秩不同，将秩较小的树的根作为另一棵树的根的子节点，新树的秩设为原来两棵树秩的较大值；如果两棵树的秩相同，任意将一棵的根作为另一棵树的根的子节点，新树的秩设为原来两棵树的秩加一。

不难发现，秩其实等价于一棵树内离根最远的点到根的距离。

通过归纳法可以得出，一棵秩为 i 的树，至少要有 2^i 个节点，因此任意一棵树的秩不超过 $O(\log n)$。相应的，树高最坏情况下是 $O(\log n)$ 的。

按秩合并的应用

按秩合并的应用

结合性质 2，为了实现对历史版本的询问，可以记录下每条边出现的时刻。

按秩合并的应用

结合性质 2，为了实现对历史版本的询问，可以记录下每条边出现的时刻。
询问时从询问点往根走，逐条边判断在指定版本是否存在，即可找到询问点在指定时刻所在树的根节点。
合并和询问的复杂度都是最坏 $O(\log n)$ 的。

倍增法的应用

倍增法的应用

结合性质 1、2，在询问时，所求的其实是给定点到根的路径上，满足出现的时刻早于询问时刻的边中，离根最近的那一条。
此时，可以应用倍增法在 $O(\log$ 树高 $)$ 的复杂度内解决。

倍增法的应用

考虑如何维护倍增所需的信息。

考虑如何维护倍增所需的信息。

对于每个 x 和 i，记录从点 x 向根走 2^i 步所到节点 $f_{x,i}$ 和 x 与 $f_{x,i}$ 连通的最早时刻 $t_{x,i}$。每次合并操作，实际上是给某个 $f_{x,0}$ 赋值。当某个 $f_{x,i}$ 被确定后，对于所有满足 $f_{y,i} = x$ 的 y，$f_{y,i+1}$ 也将被确定。对于每对 (x, i)，用链表将满足 $f_{y,i} = x$ 的 y 记录下来，当 $f_{x,i}$ 确定后，就可以根据这个链表更新出所有新确定的 $f_{x,i}$。与此同时，所有被确定的 $f_{x,i}$ 其对应的 $t_{x,i}$ 即为当前时刻。

考虑如何维护倍增所需的信息。

对于每个 x 和 i，记录从点 x 向根走 2^i 步所到节点 $f_{x,i}$ 和 x 与 $f_{x,i}$ 连通的最早时刻 $t_{x,i}$。每次合并操作，实际上是给某个 $f_{x,0}$ 赋值。当某个 $f_{x,i}$ 被确定后，对于所有满足 $f_{y,i}=x$ 的 y，$f_{y,i+1}$ 也将被确定。

对于每对 (x,i)，用链表将满足 $f_{y,i}=x$ 的 y 记录下来，当 $f_{x,i}$ 确定后，就可以根据这个链表更新出所有新确定的 $f_{x,i}$。与此同时，所有被确定的 $f_{x,i}$ 其对应的 $t_{x,i}$ 即为当前时刻。

显然对于任意一个 x，最多只有 $O(\log$ 树高 $)$ 种 i 是有意义的，因此有效的位置不超过 $O(n\log$ 树高 $)$，又由于每个有效位置最多被赋值一次，合并的总复杂度也不超过 $O(n\log$ 树高 $)$。

倍增法的应用

注意到按秩合并只在合并过程中进行了优化，而倍增法则只在询问过程中进行了优化。

倍增法的应用

注意到按秩合并只在合并过程中进行了优化，而倍增法则只在询问过程中进行了优化。

结合这两个算法，就得到了一个单次询问复杂度为最坏 $O(\log\log n)$，合并的总复杂度为 $O(n\log\log n)$ 的支持询问历史版本的并查集算法。

致谢

感谢我的父母、家人。

感谢曾艺卿老师。

感谢同学们。

感谢中国计算机学会。