数据结构与算法 I 作业 19

2019201409 于倬浩

2020 年 12 月 15 日

19.4-1

由于每次 `Decrease_Key` 都会从原堆中砍掉以该节点为根的一棵子树，因此考虑使用此操作构造定长的链。

首先，假设我们能构造出一条长度为 k 且每个节点都没被打标记的链，我们如果可以再通过同样的方法，构造出来一条性质完全相同，但是链顶部节点的键值小于当前链顶节点的键值，那么我们可以通过随便插入一个值更小的节点，然后调用 `Extract_Min` 来删除这个值更小的节点，顺便引发一次 `Consolidate` 来调整树的形态，得到的结果就是一颗由两条链构成的树，最长的一条链的长度为 $k+1$，另一条链长度为 k，那么我们通过 `Decrease_Key` 砍掉另一条链上的 $k-1$ 个节点，并通过 $k-1$ 次 `Decrease_Key` 和 `Extract_Min` 删掉这 $k-1$ 个节点，即可得到一个长度为 $k+1$ 且根节点被标记的链。

接下来，我们继续采用同样的方法，构造出另一条长度为 $k+1$ 的链，但是根节点被标记。再次随便插入一个更小的值，调用一次 `Extract_Min`，触发一次 `Consolidate`，拼成一颗两条链构成的树，最长链长度为 $k+2$，另一条链长度为 $k+1$，但是树根和次长链方向的节点（共两个节点）有标记。因此，我们对次长链方向有标记的节点使用 `Decrease_Key`，由于 `Cascading_Cut` 操作，原树根和当前节点都被切掉。

最后，只需要使用 $k+1$ 次的删除操作，切掉多余的节点，即可得到一个长度为 $k+1$ 的链，且每个节点都没有标记。

对于边界情况，长度为 1 的链，只需要插入 3 个节点，使用一次删除操作除掉最小节点即可得到。

梳理一下思路，我们实际上是通过数学归纳法，如果存在构造长度为 k 的无标记节点构成的链的方法，那么就可以构造出长度为 $k+1$ 的无标记节点构成的链，且对于边界情况 $k=1$ 仍然可行，因此我们可以构造出任意由 k 个节点构成的链。注意到这种方法在构造的过程中，已经证明了正确性。

19-1

- a.

  由于每个结点的度数都是 $O(lgn)$ 级别的，而不是 $O(1)$ 级别的。将 x 的儿子直接放到根

链表后，还需要逐个更新儿子们的父亲指针，因此需要节点度数次运算，因此第七行操作的分析不正确，实际运行时间为 $O(lgn)$（或认为是 $O(x.degree)$）。

- b.

由于每次 `Cascading-Cut` 的运行时间为 $O(1)$，共需运行 c 次，且第七行代码的运行时间为 $O(x.degree)$，因此实际运行时间为 $O(c + x.degree)$

- c.

考虑势函数的增量。

该次操作执行了 c 次 `Cascading_Cut`，最多切出 $c-1$ 棵树放进根链表，最多将 1 个节点由未标记变成标记，`Pisano_Delete` 的第 7 行会增加 $x.degree$ 棵树，因此操作过后，树的总数最多为 $t(H)+c-1+x.degree$，被标记的节点总数最多增加 1，但是一定会减少 c 个。因此，形式化地表示如下：

$$
\begin{aligned}
t(H') &\le t(H) + c - 1 + x.degree \\
m(H') &\le m(H) - c + 1 \\
\therefore \Phi(H') &\le t(H) + 2m(H) + x.degree + 1
\end{aligned}
$$

- d.

和原算法一样，最终的摊还代价依旧是 $\Theta(x.degree)$，而由于树本身的性质不变，$\Theta(x.degree) = \Theta(lgn)$ 这一性质不变，因此算法摊还代价渐进意义上和原来保持一致。