Homework5 Solution

洪方舟
2016013259
Email: hongfz16@163.com

2018 年 3 月 30 日

Problem 17.4-2

可以将势函数展开为如下形式：

$$\Phi(T) = \begin{cases} 2T.num - T.size & \alpha \geq \frac{1}{2} \\ T.size - 2T.num & \alpha < \frac{1}{2} \end{cases}$$

若 $\alpha_i \geq \frac{1}{2}$，则有 $T.num(i) = T.num(i-1) - 1$ 且 $T.size(i) = T.size(i-1)$，因此

$$\begin{aligned} \hat{c}_i &= 1 + 2T.num(i) - T.size(i) - 2T.num(i-1) + T.size(i-1) \\ &= 1 - 2 \\ &= -1 \end{aligned}$$

若 $\alpha_i < \frac{1}{2}$，如果此时发生表收缩，则 $T.num(i) = T.num(i-1) - 1$ 且 $T.size(i) = \frac{2}{3}T.size(i-1) - 1$，因此

$$\begin{aligned} \hat{c}_i &= T.num(i) + 1 + T.size(i) - 2T.num(i) - T.size(i-1) + 2T.num(i-1) \\ &= T.num(i-1) - \frac{1}{3}T.size(i-1) \\ &= \Theta(1) \end{aligned}$$

若 $\alpha_i < \frac{1}{2}$，如果此时未发生表收缩，则 $T.num(i) = T.num(i-1) - 1$ 且 $T.size(i) = T.size(i-1)$，因此

$$\begin{aligned} \hat{c}_i &= 1 + T.size(i) - 2T.num(i) - T.size(i-1) + 2T.num(i-1) \\ &= 3 \end{aligned}$$

综上，使用此策列，$TABLE - DELETE$ 操作的摊还代价的上届是一个常数。

Problem 17-2

a.

SEARCH(A, x)

let $index = -1$,$result = -1$
for $i = 0$ to k
    do if $A_i[0] \geq x$
        then $result =$ BINARY-SEARCH(A_i, x)
            if $result \neq -1$
              then $index = i$
                  BREAK
RETURN $\{index, result\}$

最坏情况下需要搜索所有 k 个集合，每个集合使用二分法复杂度为 $O(\lg n)$，则最坏情况下总的运行时间为 $O(\lg^2 n)$

b.

INSERT(A, x)

if $\lceil \lg n + 1 \rceil > k$
    then let A_k be new array
let $i = 0$
let B_0 be new array containing x
while A_i not empty
    do $B_{i+1} =$ MERGE(A_i, B_i)
        let A_i be empty array
        $i = i + 1$
$A_i = B_i$

最坏情况下，需要将 A_0 到 A_{k-1} 全部归并为 A_k，此时运行时间为 $\sum_{i=1}^{k-1} 2^i = \Theta(2^k) = \Theta(n)$
下面分析摊还时间，定义势函数为 $\Phi(D_i) = \lg n * b_i$，其中 b_i 为第 i 次操作之后装满元素的数组的个数，假设第 i 次操作一共归并了 t_i 个数组，则有 $b_i = b_{i-1} - t_i + 1$，因此有

$$
\begin{aligned}
\hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\
&= \lg n * t_i + \lg n * b_i - \lg n * b_{i-1} \\
&= \lg n * t_i + \lg n * (1 - t_i) \\
&= \lg n
\end{aligned}
$$

因此该插入算法的摊还时间为 $\lg n$

c.

DELETE(A,x)

```
let {index, result} = SEARCH(A, x)
if index = -1
    then ERROR "Cannot find x in A"
let i = 0
while A_i is empty
        do i=i+1
Delete A_index[result]
Insert A_i[2^i - 1] in A_index so that A_index is still ordered
let count = 0
for j = 0 to i - 1
    then for k = 0 to 2^j - 1
             then A_j[k] = A_i[count]
                  count = count + 1
```

调用 SEARCH 耗时 $O(\lg^2 n)$，找到第一个非空数组耗时 $O(\lg n)$，插入新的元素耗时 $O(n)$，重新整理数据耗时 $O(n)$，因此运行时间为 $O(n)$

Problem 19-3

a.

FIB-HEAP-CHANGE-KEY(H,x,k)

```
if k < x.key
    then FIB-HEAP-DECREASE-KEY(H, x, k)
if k = x.key
    then RETURN
FIB-HEAP-DELETE(H, x)
FIB-HEAP-INSERT(H, k)
```

当 $k > x.key$ 时，删除操作摊还时间 $O(\lg n)$，插入操作摊还时间 $O(1)$，则总的摊还时间为 $O(\lg n)$
当 $k = x.key$ 时，摊还时间为 $O(1)$
当 $k < x.key$ 时，减值操作摊还时间 $O(1)$

b.

对斐波那契堆的数据结构进行修改如下：将所有的叶子节点通过双向循环链表连接，如果剪切掉了一个叶子节点，需要考察其双亲结点，若双亲结点除了待剪切的结点外没有其他节点，那么就把双亲结点加入叶子节点的链表；
对势函数的修改如下：$\Phi(H) = t(H) + 2m(H) + c * H.size$，其中 c 为足够大的常数
下面说明删除结点的方法：任意找到一个叶子节点，将其从它双亲结点的孩子列表中删除，如果其双亲结点 $mark$ 为 $False$，则将其改为 $True$，并将其从叶子节点的链表中删除，删除方法同上所述；

这样删除一次的时间复杂度为 $O(1)$，删除 q 次的时间复杂度为 $O(q)$，下面使用势函数分析该算法的摊还时间

$$
\begin{aligned}
\hat{c_i} &= c_i + \Phi_i - \Phi_{i-1} \\
&= O(q) + t_i + 2m_i + c * H_i.size - (t_{i-1} + 2m_{i-1} + c * H_{i-1}.size) \\
&= O(q) - c * q + \Delta(t) + 2\Delta(m) \\
&= O(q) - c * q - O(q) + 2O(q) \\
&= O(q) - c * q \\
&= O(1)
\end{aligned}
$$

综上所述，该操作的摊还时间为 $O(1)$；