Homework5 Solution

洪方舟
2016013259
Email: hongfz16@163.com

2018 年 3 月 27 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)

```
1  let index = −1,result = −1
2  for i = 0 to k
3        do if A_i[0] ≥ x
4              then result =BINARY-SEARCH(A_i, x)
5                   if result ≠ −1
6                      then index = i
7                           BREAK
8  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)

```
1  if ⌈lg n + 1⌉ > k
2     then let A_k be new array
3  let i = 0
4  let B_0 be new array containing x
5  while A_i not empty
6        do B_{i+1} = MERGE(A_i, B_i)
7           let A_i be empty array
8           i = i + 1
9  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)
 1  let {index, result} = SEARCH(A, x)
 2  if index = −1
 3     then ERROR "Cannot find x in A"
 4  let i = 0
 5  while A_i is empty
 6        do i=i+1
 7  Delete A_index[result]
 8  Insert A_i[2^i − 1] in A_index so that A_index is still ordered
 9  let count = 0
10  for j = 0 to i − 1
11     then for k = 0 to 2^j − 1
12             then A_j[k] = A_i[count]
13                  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)
1  if k < x.key
2     then FIB-HEAP-DECREASE-KEY(H, x, k)
3  if k = x.key
4     then RETURN
5  FIB-HEAP-DELETE(H, x)
6  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.