算法分析与设计基础作业8

软件71 骆炳君 2017013573

17.4-3

设第i个操作是TABLE-DELETE，则有$num_i = num_{i-1} - 1$.

当$a_{i-1} < \frac{1}{2}$时，$\Phi(T) = T.size - 2 \cdot T.num$

若未触发收缩操作，$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + 2 = 3$

若触发收缩操作，则$num_{i-1} = \frac{1}{3} size_{i-1}$，$size_i = \frac{2}{3} size_{i-1}$，记$n = num_{i-1}$，$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = \frac{1}{3}n + (\frac{2}{3}n - 2(\frac{1}{3}n - 1)) - (n - \frac{2}{3}n) = 2$

当$a_{i-1} \geq \frac{1}{2}$时，$\Phi(T) = 2 \cdot T.num - T.size$

若$a_i \geq \frac{1}{2}$，$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = O(1)$

若$a_i < \frac{1}{2}$，则$num_{i-1} = \frac{1}{2} size_{i-1}$，$\Phi_i = size_i - 2num_i$，$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (size_i - 2num_i) - (2num_{i-1} - size_{i-1}) = 1 + 2 = 3$

综上，TABLE-DELETE的摊还代价的上界是一个常数.

17-2

a.

SEARCH操作需要遍历A_0至A_{k-1}这k个数组，在每个数组中进行二分查找，直到找到该元素为止，算法实现如下：

```
SEARCH(A,x)
for i=0 to k-1:
    m=BINARY_SEARCH(A[i],x)
    if m != -1:
        break
return i,m
```

最坏运行时间$T(n) = O(\sum_{i=0}^{k-1} \lg A_i.len) = O(\sum_{i=0}^{k-1} \lg 2^i) = O(k^2) = O(\lg^2 n)$

b.

算法描述

查找n最低位的0，记为n_i，则A_i为最小的空数组，将待插入的数据与A_0进行归并排序，再依次与$A_1 ... A_{i-1}$进行归并排序，将所得结果存入A_i中，并将$A_m(m = 0, \cdots, i)$设为空.

分析最坏情况运行时间

在最坏情况下，需要归并A_0到A_{k-1}的所有数组，因为一次归并排序为线性时间复杂度，所以最坏运行时间$T(n)=O(\sum_{i=1}^{k}2^i)=O(2^k)=O(n)$

分析摊还时间

使用核算法，记每个插入元素的摊还代价为$\lg n$，因为一个元素所经历的归并过程最多为$\lg n$个，而在每个归并过程中仅被访问1次，所以摊还代价可以支付其插入后归并排序的时间花费，摊还时间为$O(\lg n)$.

c.

算法描述

查找待删除元素所在的数组，记为A_k，查找n最低位的1，记为n_m，则有$k\geq m$，在A_k中删除该元素，并将A_m的最大值插入到A_k中，然后将A_m中剩余的2^m-1个元素按原顺序拆分为长度为$1,2,4,\cdots,2^{m-1}$的部分，移动到$A_0,A_1,\cdots,A_{m-1}$中.

运行时间分析

查找k和m的时间为$O(\lg n)$，插入操作的时间为$O(\lg n)$，分拆A_m的时间为$O(\lg n)$，所以最终的时间复杂度为$O(\lg n)$.

19-3

a.

当$k<x.key$时，直接调用FIB-HEAP-DECREASE-KEY，摊还时间为$O(1)$，当$k=x.key$时，摊还时间为$O(1)$，当$k>x.key$时，先调用FIB-HEAP-DELETE从堆中删除原有的x，再调用FIB-HEAP-INSERT插入k，摊还时间为$O(\lg n)$.

b.

将势能函数改为$\Phi(H)=t(H)+2m(H)+H.n$，与原来的势函数相比，仅在插入或删除（抽出）元素时有所不同，且变化量为常数，因此不会影响其他操作的摊还时间.同时将堆上的所有叶子节点通过一个双向循环链表连接起来.

在实现FIB-HEAP-PRUNE时，沿着连接叶子节点的链表删除q个节点，并在删除的同时将产生的新的叶子节点加入该链表中.

算法的实际代价为$O(q)$，摊还时间为$O(q)-q=O(1)$，因为在势的单位足够大时可以抵消$O(q)$.