算法分析与设计基础 第四次作业

徐浩博 软件02 2020010108

Problem 1

a. 我们假设数组序号从1—n，n是数据的总数，那么我们对任意一个下标i的节点，有：

父节点下标：

```
PARENT(i):
    return floor((i + d - 2) / d)
```

第k个子节点下标($1 \le i \le d$)：$di + k$

```
kth-SON(i, k):
    return (i - 1) * d + k + 1
```

b. 包含n个元素的d叉堆的高度是$\lfloor \log_d n(d-1) \rfloor$，规模是$\Theta(\log_d n)$.

c. EXTRACTMAX

```
EXTRACT-MAX:
    if heap-size[A] < 1:
        error "heap-underflow"
    max = A[1]
    A[1] = A[A.heap-size]
    A.heap-size = A.heap-size - 1
    MAX-HEAPIFY(A, 1)
    return max
```

```
MAX-HEAPIFY(A, i):
    MAX-SON = i
    for k = 1 to d:
        if(kth-SON(i, k) <= A.heap-size and A[kth-SON(i, k)] > A[MAX-SON]):
            MAX-SON = kth-SON(i, k)
    if MAX-SON != i:
        swap(A[i], A[MAX-SON])
        MAX-HEAPIFY(MAX-SON)
```

时间复杂度$O(d \log_d n)$.

d. INSERT

```
INSERT(key):
    A.heap-size = A.heap-size + 1
    A[A.heap-size] = key
    pos = A.heap-size
    while pos > 1:
        if A[pos] > A[PARENT(pos)]:
            swap(A[pos], A[PARENT(pos)])
            pos = PARENT(pos)
        else return
```

时间复杂度$O(\log_d n)$.

e. INCREASE-KEY

```
INCREASE-KEY(A, i, key):
    if A[i] > key:
        error "CANNOT-INCREASE"
    A[i] = key
    while i > 1:
        if A[i] > A[PARENT(i)]:
            swap(A[i], A[PARENT(i)])
            i = PARENT(i)
        else return
```

时间复杂度$O(\log_d n)$.

Problem 2

a. 考虑到数组中每个元素值都相同，那么随机取出任何一个数，数组中左右的数都会被分到PARTITION的左边，时间递推式可以写作：

$$T(n) = T(n-1) + \Theta(n)$$

那么则有$T(n) = \sum_{i=1}^{n} = \frac{n(n+1)}{2} = \Theta(n^2)$，时间复杂度为$\Theta(n^2)$.

b. PARTITION'(A, p, r)

```
PARTITION'(A, p, r):
    x = A[r]
    q = p - 1
    t = p - 1
```

```
    for i = p to r - 1
        if A[i] < x:
            q = q + 1
            t = t + 1
            swap(A[i], A[q])
            swap(A[i], A[t])
        else if A[i] == x:
            t = t + 1
            swap(A[t], A[i])
    t = t + 1
    swap(A[r], A[t])
    return q, t
```

c. RANDOMIZED-QUICKSORT'

```
RANDOMIZED-QUICKSORT'(A, p, r):
    if p < r:
        q, t = PARTITION'(A, p, r)
        RANDOMIZED-PARTITION'(A, p, q)
        RANDOMIZED-PARTITION'(A, t + 1, r)
```

d. 设原数组排好序后各个元素为$z_1 \cdots z_n$，对任意两个元素$z_i \leq z_j$，显然如果要排序出z_i，z_j的大小，$Z_{ij} = \{z_q \cdots z_i \cdots z_j \cdots z_t\}$中必然有一元素与$z_i$或$z_j$作过比较（$z_q$是等于$z_i$的下标最小的元素，$z_t$是等于$z_j$的下标最大的元素）. 他们被选为主元pivot是等可能的.

1)选择$z_x(z_q \leq z_x \leq z_t$，$x \neq i$，$j)$作为pivot，这种情况发生的可能性是$\frac{t-q-1}{t-q+1}$，如果$z_x$与$z_i, z_j$均不相等时，则$z_i, z_j$彼此无须比较；如果与$z_i, z_j$中的一个或两个相等时，根据RANDOMIZED-QUICKSORT'写法，z_i, z_j也无需比较.

2)选择z_i或z_j作为pivot，这种情况发生的可能性是$\frac{2}{t-q+1}$. 这种情况下z_i, z_j需要比较一次.

综合以上：$Pr\{z_i\text{与}z_j\text{进行比较}\} = \dfrac{2}{t-q+1} \leq \dfrac{2}{j-i+1}$，即比较可能性不大于7.4.2节结果，则总的期望比较次数也不大于7.4.2的结果，因此期望的运行时间不大于O(nlogn). 当重复元素较多，运行时间趋近于O(n)；重复元素较少，运行时间则趋近于O(nlogn).