算法基础作业 HW7

PB20111686 黄瑞轩

1

记 c_i 为第 i 个操作的代价，由题可知

$$c_i=\begin{cases}i, & i=2^k\\ 1, & \text{other}\end{cases}\tag{1}$$

（1）**聚合分析**：总的代价

$$\begin{aligned}\sum_{i=1}^{n}c_i&=\sum_{k=1}^{\lfloor\lg n\rfloor}2^k+\sum_{i\neq 2^k}1\\&\le 2^{1+\lfloor\lg n\rfloor}-1+n\\&\le 2n-1+n\\&\le 3n-1=O(n)\end{aligned}$$

所以每个操作摊还代价为

$$\frac{O(n)}{n}=O(1)\tag{2}$$

（2）**核算法**：记 $\hat{c}_i$ 为摊还代价，若令 $\hat{c}_i=3$，则由（1）可知

$$\sum_{i=1}^{n}c_i<3n=\sum_{i=1}^{n}\hat{c}_i=O(n)\tag{3}$$

实际代价为 $O(n)$，每个操作的摊还代价为

$$\frac{O(n)}{n}=O(1)\tag{4}$$

（3）**势能法**：记 $\Phi(D_0)=0$，$\Phi(D_i)=2i-2^{1+\lfloor\lg i\rfloor},i>1$

当 $i=1$ 时

$$\hat{c}_1=c_1+\Phi(D_1)-\Phi(D_0)=1\tag{5}$$

当 $i=2^k,k>0$ 时

$$\begin{aligned}\hat{c}_i&=c_i+\Phi(D_i)-\Phi(D_{i-1})\\&=i+(2i-2^{1+k})-[2(i-1)-2^k]\\&=2\end{aligned}\tag{6}$$

当 $i\neq 2^k,k>0$ 时

$$\begin{aligned}\hat{c}_i&=c_i+\Phi(D_i)-\Phi(D_{i-1})\\&=1+(2i-2^{1+\lfloor\lg i\rfloor})-[2(i-1)-2^{1+\lfloor\lg(i-1)\rfloor}]\\&=3\end{aligned}\tag{7}$$

每个操作的摊还代价都是 $O(1)$。

2

假设元素互不相等，假设元素都是 int 类型。

```
class queue {
public:
    queue(); // 构造函数，略
    void ENQUEUE(int x) {
        data[tail] = x;
        // 维护min，如果新入队者更小，则修改min为新入队者下标
        if (x < data[min]) {
            min = tail;
        }
        // 维护tail
        if (tail == length) {
            tail = 0;
        }
        else {
            tail++;
        }
    }
    int DEQUEUE() {
        int ret = data[head];
        // 维护min，如果出队的是min，则重新获取min
        if (min == head) {
            min = _ITERATIVE_FIND_MIN();
        }
        // 维护head
        if (head == length) {
            head = 0;
        }
        else {
            head++;
        }
        return ret;
    }
    int FIND_MIN() {
        return data[min];
    }

private:
    unsigned tail; // 指示队尾
    unsigned head; // 指示队头
    unsigned min; //
    const unsigned length; // 指示队列长度
    int* data; // 队列数据存放

    // 迭代方式找最小元素指针
    unsigned _ITERATIVE_FIND_MIN() {
        unsigned _it = head + 1;
        unsigned _min = _it;
        while (_it != tail) {
            if (data[_it] < data[_min]) {
                _min = _it;
            }
            _it = (_it + 1) % length;
        }
        return _min;
    }
```

```
}
```

首先，如果不考虑维护 min 的话，DEQUEUE 的时间复杂度显然是 $O(1)$，在所有情况下 ENQUEUE、FIND_MIN 的时间复杂度都是 $O(1)$，下面分析加入维护 min 后的 DEQUEUE 的摊还复杂度。

_INTERATIVE_FIND_MIN 的时间复杂度为 $O(n)$，但是在整个队列的操作序列（假设数据分布是随机的）中执行的平均次数是 $O(1)$ 的，所以摊还复杂度为

$$\frac{O(1)O(n)+O(n)O(1)}{n}=O(1) \tag{8}$$

3

算法如下：

1. $T=$ 未被覆盖的元素集合，$T_{initial}=\cup_{i=1}^{m}S_i$
2. 选择 S_i，使得其在 T 中有最多的元素
3. 从 T 中去掉 S_i 中的元素

每次记录 i 的取值，重复上述 (2, 3) k 次。

证明：

设最优解（最大覆盖）的元素个数为 OPT，记 x_i 为第 i 次选择后新覆盖的元素个数，且

$$y_i=\sum_{j=1}^{i}x_j \quad z_i=\mathrm{OPT}-y_i \tag{9}$$

初始条件 $y_0=0, z_0=y_k=\mathrm{OPT}$。

算法第 2 步，考虑到最优解是 k 个集合的并集，所以必然存在一个集合，记 OPT 中剩余未被覆盖元素集合为 S，这个集合为 S'，满足

$$\frac{|S\cap S'|}{|S|}\geq\frac{1}{k} \tag{10}$$

根据上面的记号，即

$$x_{i+1}\geq\frac{z_i}{k} \tag{11}$$

所以

$$z_{i+1}=z_i-x_{i+1}\leq z_i(1-\frac{1}{k})\leq\mathrm{OPT}(1-\frac{1}{k})^{i+1} \tag{12}$$

所以

$$y_k=\mathrm{OPT}-z_k\geq\mathrm{OPT}[1-(1-\frac{1}{k})^k] \tag{13}$$

即近似比为

$$1-(1-\frac{1}{k})^k\geq 1-\frac{1}{e} \tag{14}$$