第6章

1. 在数据结构上执行操作序列 $OP_1, \cdots, OP_n$.

当 $i=2^k$, OP_i 的代价为 i

当 $i \neq 2^k$, OP_i 的代价为 1

(a) 用聚集方法分析该操作序列的时间复杂度上界

(b) 用会计方法分析该操作序列的时间复杂度上界.

(a) 聚集: 总代价 $/n$.

当 $i=2^k$, $t_1 = \sum_{k=0}^{k=\lfloor \log_2 n \rfloor} 2^k = 2^{\lfloor \log_2 n \rfloor + 1} - 1 = 2n-1$

当 $i \neq 2^k$, $t_2 = \sum_{i \neq 2^k} 1 = n - (\lfloor \log_2 n \rfloor + 1)$

$T(n) = t_1 + t_2 = 3n - \log_2 n$

$A(n) = \frac{T(n)}{n} = 3 - \frac{\log_2 n}{n}$ 平摊代价为 $O(1)$

当 $n \to \infty$, $\frac{\log_2 n}{n} \to 0$. 所以 $A(n) \to O(1)$.

$T(n)$ 复杂上界为 $O(n)$.

(b) 会计法: 每个 cost(i) 付 3 美元, 1 美元用来付基本开销.

剩下 2 美元预支给 $i=2^k$ 的开销.

考察 $(2^{k-1}, 2^k]$, 此 2^k 代价 $< 2^{k-1}+1 = 2^k+1$

$\underbrace{2^{k-1}+1, \cdots, 2^k+2^{k-1}}_{共 2^{k-1} 个数字}$, 其中 $\underbrace{2^{k-1}+1, \cdots, 2^k+2^{k-1}-1}_{2^k \text{用钱都支付}}$ 实际代价为 1, 已被支付.

2. k位二进制计数器上，Increment，Reset 构成的长度为n的操作序列的时间复杂度上界为 O(n)

（基本代价定义0和1翻转次数，注意不是遍历的复杂度）

```
Increment(counter)
for i from 0 to k-1:
    if counter[i]==0:
        counter[i]=1
        break
    else
        counter[i]=0
```

```
Reset(counter)
for i from 0 to k-1:
    if counter[i]=1:
        counter[i]=0.
```

会计法分析：

每次一位被置1，付2美元.

- 1美元用于置1的开销
- 1美元存储在该"1"位上，用于支付其被置0时的开销

置0操作无需再付款（不论是Increment导致的1→0.
还是Reset导致的1→0.）

Cost(Increment&Reset) = 2. n个Increment代价 < 2n.

T(n) = O(n).