第6章

1. 在数据结构上执行操作序列 $op_1, \dots, op_n$.

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 $i=2^k$，$t_1 = \sum_{k=0}^{N=\log_2 n} 2^k = 2^{N+1}-1 = 2 \cdot 2^{\log_2 n}-1 = 2n-1$

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

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

$A(n) = \frac{T(n)}{n} = 3 - \frac{\log_2 n}{n}$　　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) 当 $i=2^k$ 的代价，$2^{k-1} \sim 2^k$ 中分摊 2 个即可。$i \neq 2^k$，分

摊如是来看每个数字 i 最多承担 3 个代价.

$T(n)$ 复杂度上界为 $O(n)$

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)

```
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:
    counter[i]=0
```

用平摊代价分析思考：Increment，平摊代价为 O(1)

Reset，平摊代价为 O(k)

n次操作，假设执行了t次Reset，Reset平摊代价为k，支付tk。

每次Reset至少会执行k次Increment，所以有 $n \geq tk$

总的代价：Increment + Reset $= n + tk \leq 2n$.

$T(n) = O(n)$,