17.1-2 考虑初始计数器为 10⋯00（共 k 位），假设操作序列为：DECREMENT, INCREMENT, DECREMENT,⋯交替，那么每次操作都会反转 k 位，故花费时间开销为 O(kn)。

17.3-1 考虑Φ’(Di)= Φ(Di)- Φ(D0)，则势函数Φ’将数据结构 Di 映射到实数Φ’(Di)上，且有式Φ’(D0)=0，Φ’(Di)= Φ(Di)- Φ(D0)>=0 成立；
由于Φ’(Di)- Φ’(D(i-1))= [Φ(Di)- Φ(D0)]- [Φ(D(i-1))- Φ(D0)]= Φ(Di)- Φ(D(i-1))，故摊还代价和原势函数相同。

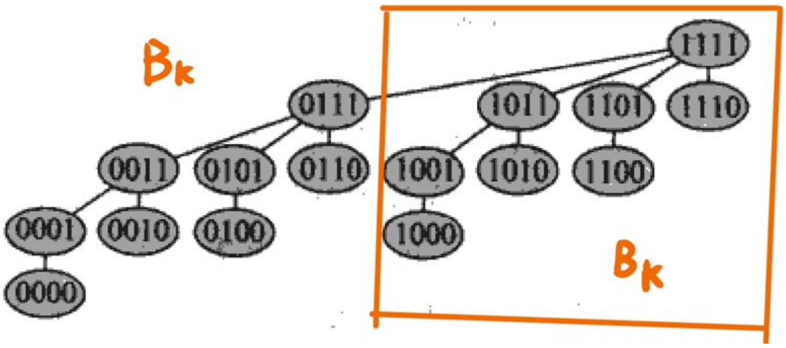
17.3-4 仍然将势函数定义为其中对象的数目，由于初始包含 S0 个对象，结束时包含 Sn 个对象，而两者并没有大小比较关系，所以不能保证Φ(Di)>= Φ(D0)的成立，但是仍然有：

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) \ \sim \ O(n)$$

然而Φ(D0)=S0，Φ(Dn)=Sn 成立，故有总实际代价上界为：

$$\sum_{i=1}^{n} c_i \ \sim \ O(n) + S_0 - S_n$$

19.1-3 先证明在 x 的二进制表示中共有 j 个 1，归纳证明。
首先对于只有二项树 B0 以及 B1，均可以进行验证是正确的。
假设对于二项树 Bk 是正确的，则 B(k+1)可以看成是两个二项树 Bk 根节点相连形成，如图：

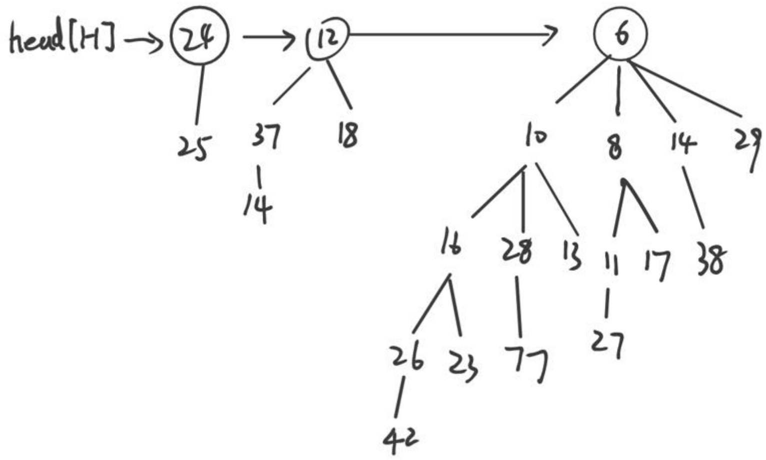


对于左边的 Bk 由后序遍历是排在最前面的，所以现在的序号相当于在之前二项树 Bk 的二进制序号前填上了一个 0，而右边的二项树排序位于左二项树之后，所以相当于在之前二项树 Bk 的二进制序号前填上了一个 1，由二项树的性质可知，右边的二项树恰好比左边的二项树是高 1 层的，再利用归纳假设，所以恰好使得每一层结点的二进制表达的 1 的数目相同。又有左二项树 j= k-i= (k+1)-(i+1)，所以归纳对于 k+1 情况成立，所以归纳成立。

恰好包含 j 个 1 的二进制 k 串共有：C_k^i

不难发现，按照后序遍历，排在结点 x 子树森林前的树的大小至少是 x 子树森林中最大子树大小四倍，又 x 子树森林所有树的大小为 2^0，2^1⋯2^(deg(x)-1)，即 x 的序号二进制的后部全部是连续的 1，一一对应于 x 子树森林所有树的大小，最右边的 0 即分割了 x 子树森林和结点 x 子树森林前的树，所以 x 二进制表达最右边 0 的右边的 1 个数与 x 的度数相同。

19.2-2



19.2-6

```
BINOMIAL-HEAP-DELETE(H,x)
    P=head[H]
    Min=P.key
    While(P) :
        If P.key<Min :
            Min=P.key
        P=P.sibling
    Min_key=Min-1
    BINOMIAL-HEAP-DECREASE(H,x,Min_key) //调整需要删除的结点的 key 成为最小的 key
    BINOMIAL-HEAP-EXTRACT-MIN(H) //抽取最小 key 的结点完成删除
```

由于改变寻找最小 key 时间复杂度为 O(lgn)，且后续操作亦为 O(lgn)，所以总时间复杂度为 O(lgn)