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：



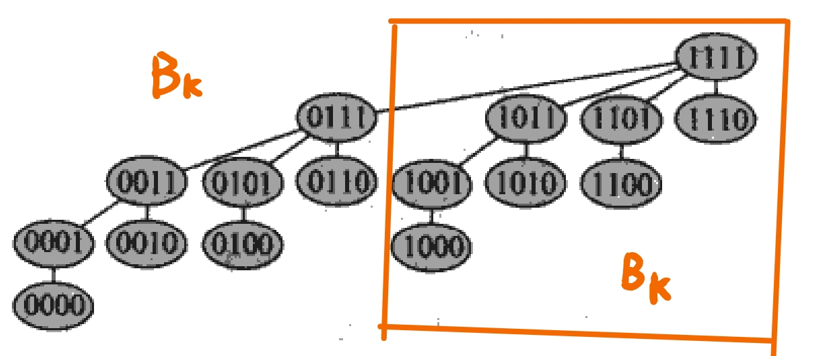
然而Φ(D0)=S0，Φ(Dn)=Sn成立，故有总实际代价上界为：



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：

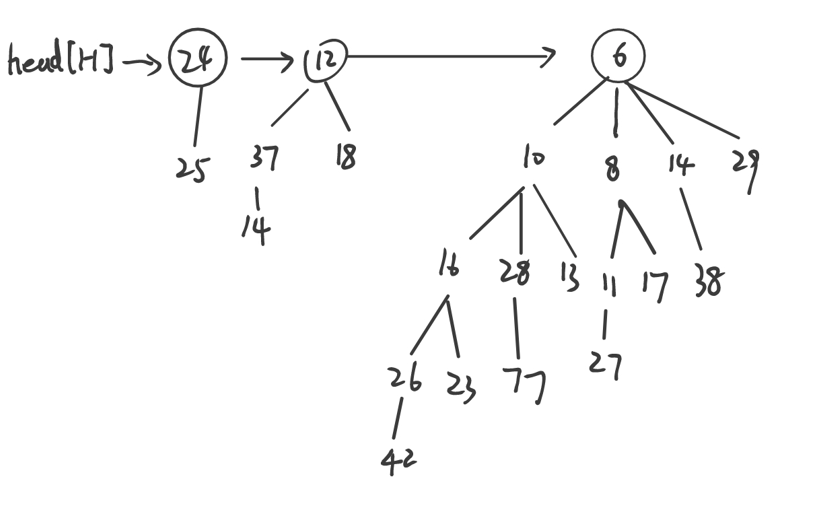


对于左边的Bk由后序遍历是排在最前面的，所以现在的序号相当于在之前二项树Bk的二进制序号前填上了一个0，而右边的二项树排序位于左二项树之后，所以相当于在之前二项树Bk的二进制序号前填上了一个1，由二项树的性质可知，右边的二项树恰好比左边的二项树是高1层的，再利用归纳假设,所以恰好使得每一层结点的二进制表达的1的数目相同。又有左二项树j= k-i= (k+1)-(i+1)，所以归纳对于k+1情况成立，所以归纳成立。

恰好包含j个1的二进制k串共有：

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)