**Lab3 订正**

1. How would you represent a *d*-ary heap in an array?

用 array[0]存头结点，array[i]结点的孩子为 array[i\*d+1]到 array[i\*d+d],array[i]结点的双亲为array[(i-1)/d]

1. What is the height of a *d*-ary heap of*n*elements in terms of*n*and *d*?

当为满二叉树时，设高度为 h， 1+d+d2+...dh=n

(dh-1-1)/(h-1)=n

则 h=logd(n\*d-n+1)+1

因此 h=O(lgdn)。

1. Given an efficient implementation of EXTRACT-MAX in a *d*-ary max-heap. Analyze its

running time in terms of *d*and *n*.

EXTRACT-MAX()：swap(array,0,n-1)之后，因为是排好序的 heap，所以其他元素不用改变，最差情况是要依次交换把 arry[0]放到最底层（高度为 0 的地方），高度差为 O(lgdn)，其他操作均为 O(1)，所以时间复杂度为 O(lgdn)。（错误！！！！）

每次向下移都要与其d个子元素比较，是所以时间复杂度为 O(d\*lgdn)

1. Given an efficient implementation of INSERT in a *d*-ary max-heap. Analyze its running time

in terms of *d*and *n*.

在 heap 中加入一个值为 x 的元素,因为是排好序的 heap，所以其他元素不用改变，最差情况是要把这个元素依次交换放到放到最顶层，高度差为 O(lgdn)，其他操作均为 O(1)，所以时间复杂度为 O(lgdn)。

1. Give an efficient implementation of INCREASE-KEY(*A*, *i*, *k*), which first sets *A*[*i*] ՚

max(*A*[*i*], *k*) and then updates the *d*-ary max-heap structure appropriately. Analyze its running time in terms of *d*and *n*.

把 array[i]增大为 k,因为是排好序的 heap，所以其他元素不用改变，最差情况是这个元素是最底层并且要把这个元素依次交换放到放到最顶层，高度差为 O(lgdn)，其他操作均为 O(1)，所以时间复杂度为 O(lgdn)。（但是更精确的时间复杂度可能是与 i 有关，通过 i,d 确定 array[i]与最顶层的高度差 h=lgdi，时间复杂度可能是 O(lgdi)。）

每次都只和父元素比较（所以是对的）