完全二叉树（完美，满）

> 除了最底下一层都是满的
> 最底层不满也是从左到右排列的

大顶堆：父亲都大于孩子

```
堆调整
建堆
插入
获取最大值
堆排序
拓展：对顶堆
拓展：可删除堆
```

1、堆调整

- 一层一层向下互换，每次换大儿子

> 调整i为根的子堆
> 假设数组从1开始
> 递归形式

- 时间O（lgn）

```
Maxheapify(i):
il=2*i,ir=2*i+1
imax=(il<=heap_size&&A[i]>A[il]?i:il)//记得判断不越界
imax=(ir<=heap_size&&A[imax]>A[ir]?imax:ir)
if(imax!=i)
    swap(A[i],A[imax])
    Maxheapify(imax)
```

2、建堆

- 存储方式：横向编号的数组

> 优点：已知下标后易于寻找父子结点
> 注意：初始为0或为1时的索引父子方式不同！

- 时间O（n）

```
Buildmaxheap(A[1,n]):
heap_size=n
for (i= n/2 downto 1) //所有非叶结点都要操作，从序号最大的非叶结点开始
    maxhepify(i)
//return A[1,n] //要有返回?，方便堆排序复制并改变
```

3、插入

- 先加到末尾，如大小违反堆性质，互换父子

> 利用了路径唯一
> 与maxheapify无关

- 时间O（lgn）

```
Heapinsert(x):
heap_size++//默认为全局变量，不需要传递
A[heap_size]=x
i=heap_size
while(i>=1 && A[i]>A[i/2])//无需if，直接放在while里面
    swap(A[i],A[i/2])
    i=i/2
```

正确性

- 每次交换后，局部子堆合法

4、获取最大

- 先删掉，把末位换上来，再maxheapify
- 功能：返回最大值，使堆-1，保持堆性质

```
Heapextractmax(A):
max=A[1]
A[1]=A[heap_size]
heap_size--//可与上合并为一句：A[1]=A[heap_size--]
Maxheapify(1)
```

```
6 return max
```

- 时间O（lgn）

5、堆排序

- 建堆，取max，重排

> 在原数组上操作，得到的是升序

```
1 Heapsort(A[1,n]):
2 Buildmaxheap(A[1,n])
3 for (i=n downto 2)
4     max=Heapextractmax(A)
5     A[i]=max//
```

- 时间：O（nlgn），空间：O（1）