堆(heap)又可以称作优先队列(priority queue)，但是它和一般的队列是有区别的。队列操作数据是遵循数据进入队列的先后顺序来读(dequeue)或者写(enqueue)的。而优先队列是按照数据的优先级来操作的，跟数据进入的顺序没有关系。比如最大堆，就是按照数值大的优先级高来定义的。所以，看似堆的有些操作比如dequeue和enqueue和队列相同，但是它们是两个不同的概念。

堆的一个经典实现模式是完全二叉树(complete binary tree)（什么是完全二叉树？），这样实现的堆又称为二叉堆(binary heap)。所谓的完全二叉树是在二叉树上又加上一个条件，就是除了最后一层外，所有其他层必须都填满，并且最后一层所有的节点都从左往右填。（为什么从左往右？为了遍历的方便?）同时，作为堆要满足所有的父节点的优先级要高于子节点。如果是最大堆，那么父节点要大于子节点。

一般我们都会用一个数组来表示堆，那么很容易可以知道父节点和子节点在数组中的相对位置，假设父节点在数组中是k，那么两个子节点就分别是2k+1, 2k+2。同样，如果我们知道了子节点也可以推出父节点的位置，假设子节点在数组中是m，那么它的父节点是(m – 1) / 2 （也可以写成(m-1)>>1）。

构造堆：

在构造堆的时候，当然可以在每次插入时排序，但是那样的效率不高。每次插入排序为O(log(n)), 插入n次, 那么就是O(nlong(n))。比较快的方式是直接在数组上进行排序。

这个思路是：从最后一个父节点开始，调整本子树的节点，然后再调整以子节点为父节点的子树，直到叶子节点。假设一个树的节点数为n，那么最后一个父节点为n/2。

for (int i = size / 2; i >= 0; i--) // 注意i>=0，这样第一个值是数组的0

删除(dequeue)：取数组的第一个，并把最后一个叶子节点放到第一个，并重新向下排序。向下是对于子树。

添加(enqueue): 把新值放到数组最末尾，并重新向上排序(如果比父节点大，调换，再往上比较)。向上是对于整个树。

另：在STL中有priority\_queue 的用法，<algorithm>头文件

priority\_queue<int> pq;

pq.push(…); //会自动排序

pq.pop(); //得到排序后的值

默认是最大堆，想要最小堆需要输入三个参数

priority\_queue<int, vector<int>, greater<int> >pq;

如果是自定义对象要重载 > 操作符。