堆

Basic Conclusions

假设下标从0开始，层数从0开始，当前下标为i，树的高度为h，含有n个元素

- 父亲节点下标$(i-1)/2$
- 左孩子节点下标$2*i+1$
- 右孩子节点下标$2*i+2$
- 最后一个非叶子节点为$parent(n-1)$
- 元素个数最多为$2^{h+1}-1$个，元素个数最少为2^h个
- 含n个元素堆的高度为$\lfloor lgn \rfloor$

Sift Up

从堆末尾添加元素，需要对元素进行sift up操作

本例中为大根堆

```
private void siftUp(int k){
        while(k > 0 && data.get(parent(k)).compareTo(data.get(k)) < 0){
            data.swap(k, parent(k));
            k = parent(k);
        }
}
```

Sift Down

```
private void siftDown(int k){
        while(leftChild(k) < data.getSize()){
            int j = leftChild(k);
            if(j+1 < data.getSize() && data.get(j+1).compareTo(data.get(j))>0)
                j = rightChild(k);
            if(data.get(k).compareTo(data.get(j))>=0)
                break;
            data.swap(k,j);
            k = j;
        }
}
```

ExtraceMax

```
public E extractMax(){
        E ret = findMax();
        data.swap(0,data.getSize()-1);
        data.removeLast();
        siftDown(0);
        return ret;
}
```

Heapify

将数组转化为堆，复杂度可以达到O(n)

```
public MaxHeap(E []arr){
        data = new Array<>(arr);
        for(int i = parent(arr.length-1); i>=0; i--)
            siftDown(i);
}
```

Heap Sort

最大堆：从大到小排序

```
Integer [] arr = {4,1,8,2,0,7,5,6,3,9};
MaxHeap<Integer> heap = new MaxHeap<>(arr);
for (int i = 0; i < arr.length; i++)
     arr[i] = heap.extractMax();
```

Priority Queue

默认最小堆

可以传自定义Comparator

常用操作

- add/offer
- remove/poll
- peek
- isEmpty
- size

Applications

Top K 问题

找出前K大的数并按照从大到小的顺序打印。复杂度要求：O(nlog(k))

分析：

- 维护一个大小为k的最小堆，处理元素满后的逻辑。
- 逆序打印

```
#include <iostream>
#include <cstdio>
#include <vector>
#include <queue>
using namespace std;
void printResult(priority_queue<int,vector<int>,greater<int> >&q, bool first){
    if(q.size() == 1){
        cout<< q.top() << " ";
        return;
    }
    int res = q.top();
    q.pop();
    printResult(q, false);
    cout<< res;
    first? cout<<endl : cout<<" ";
}
int main(){
    int n, m;
    while(scanf("%d%d",&n,&m)!=EOF){
        priority_queue<int,vector<int>,greater<int> >q;
        for (int i=0; i<n; i++){
            int num;
            scanf("%d",&num);
            if(q.size() < m)
                q.push(num);
            else{
                if(q.top() < num){
                    q.pop();
                    q.push(num);
                }
            }
        }
        printResult(q, true);
    }
    return 0;
}
```