数据结构之树形结构2

堆

1.什么是树?

1.什么是树?

2.什么是二叉树?

1.什么是树?

2.什么是二叉树?

3.二叉树第i层最多有多少个节点?

1.什么是树?

2.什么是二叉树?

3.二叉树第i层最多有多少个节点?

4.高度为h的二叉树最多有多少个节点?

1.什么是树?

2.什么是二叉树?

3.二叉树第i层最多有多少个节点?

4.高度为h的二叉树最多有多少个节点?

5.二叉树叶节点和度为2的节点的关系是?

1.什么是树?

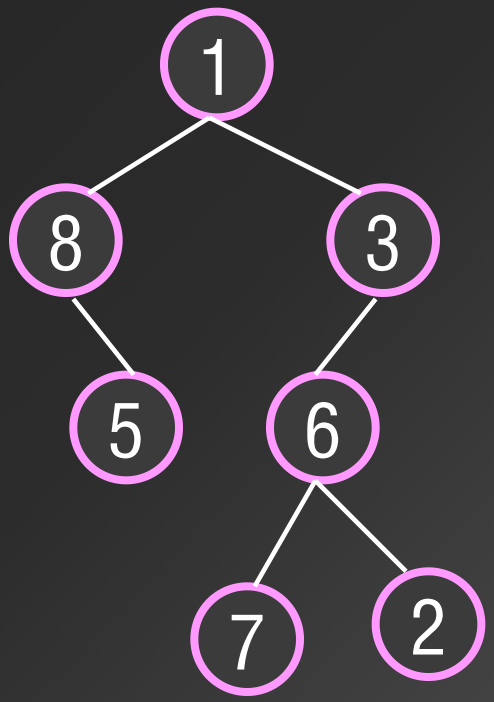
2.什么是二叉树?

3.二叉树第i层最多有多少个节点?

4.高度为h的二叉树最多有多少个节点?

5.二叉树叶节点和度为2的节点的关系是?

6.什么是完全二叉树? 什么是满二叉树?



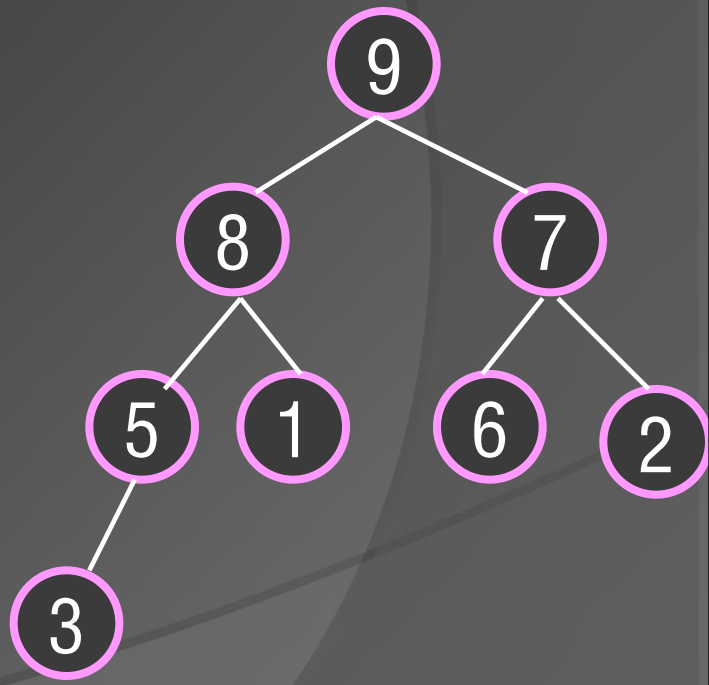
写出前序，中序，后续遍历序列

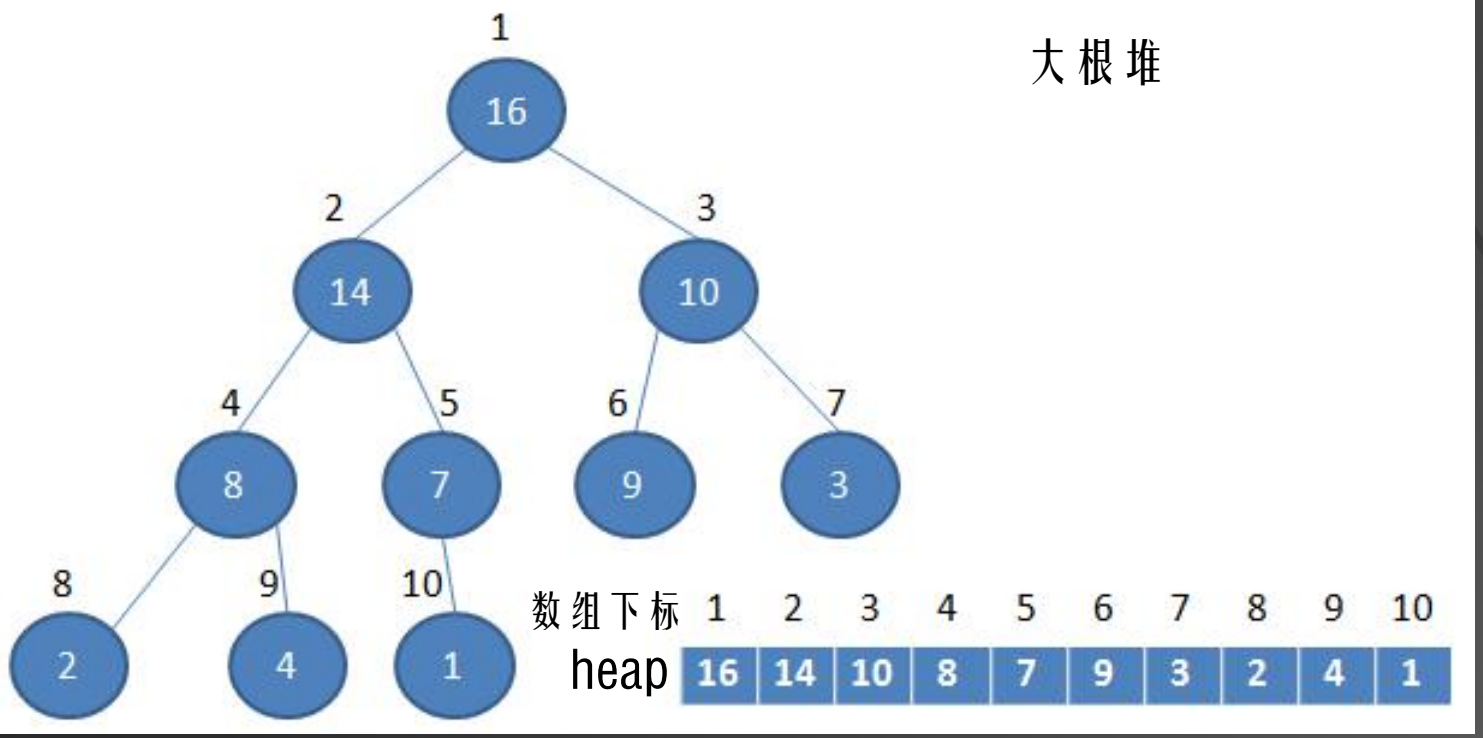
二叉堆(heap)

- 二叉堆是完全二叉树
- 任何一个节点都大于他的儿子节点的值(大根堆)

或者

- 任何一个节点都小于他的儿子节点的值(小根堆)
  - 父>子，则堆顶元素值一定最大
  - 父<子，则堆顶元素值一定最小





堆用一维数组来存储，父子关系可通过数组下标快速计算出来

```
#define Maxn 堆的最大元素个数 //#define maxn 11
int heap[Maxn] ,n;  //整型数组heap来存储堆，n表示堆的结点总数
```

规定这种存储结构中：

heap[1]是堆的根；

结点heap[i]的父亲是heap[i/2]（i>1）//比如5号节点的父亲是2号节点

结点heap[i]左孩子是heap[2*i]，右孩子是heap[2*i+1] i<=n/2

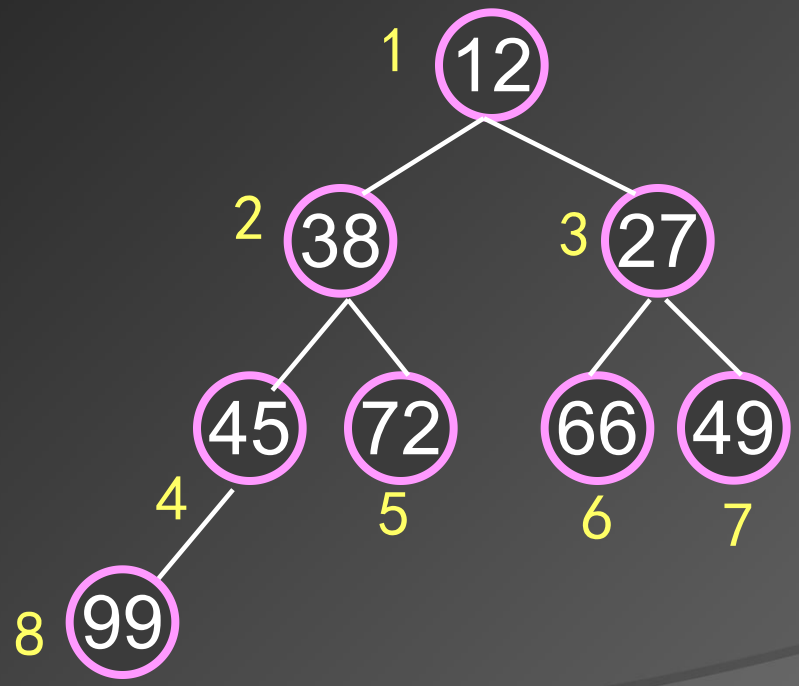
根据堆的定义可以知道：

对于大根堆：heap[1]最大

heap[i]≥h[2*i]且heap[i]≥heap[2*i + 1]（1≤i≤n/2）

已知下列数组表示一个堆，请画出这个堆！

12	38	27	45	72	66	49	99
下标 1	2	3	4	5	6	7	8



定理：堆的高度最大为 $\log_2^{(n+1)}$

因为高度为h的二叉树最多有2^h-1个节点。

$2^h-1=n$

那么$h=\log_2^{(n+1)}$

初始时　数组heap

3	6	5	7	5	2	3	1	3	8	7

n=11　数组下标

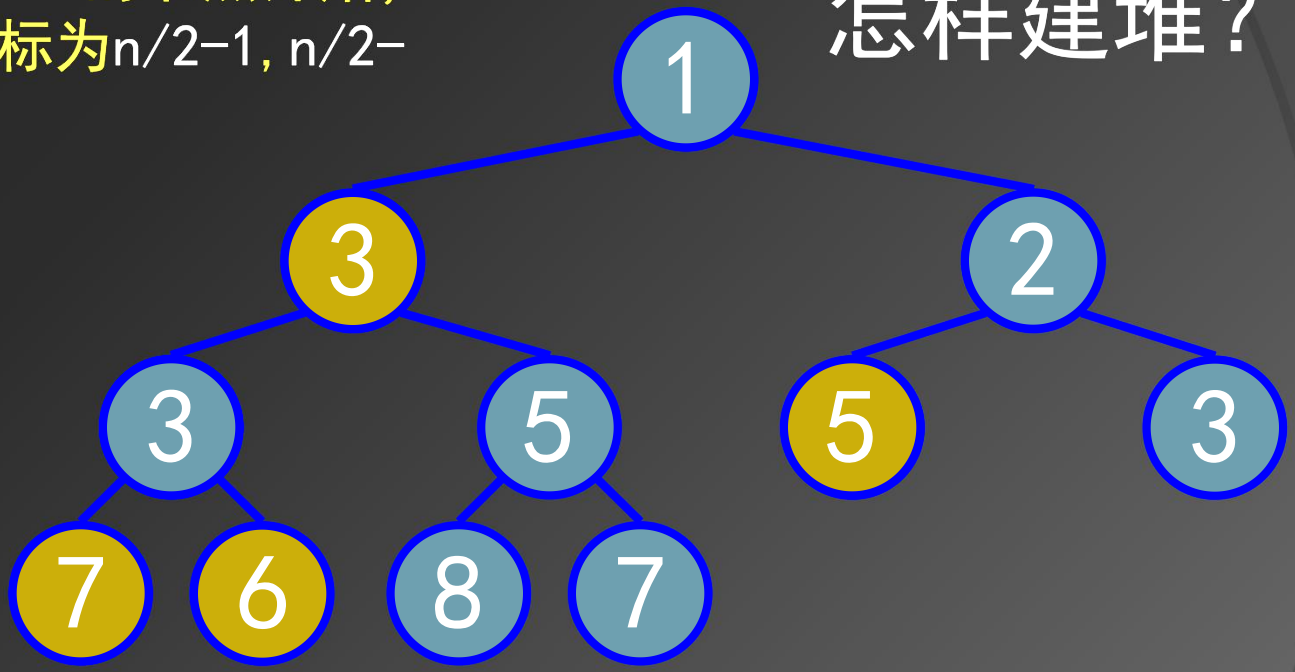
1	2	3	4	5	6	7	8	9	10	11

建堆完毕后 数组heap

1	3	2	3	5	5	3	7	6	8	7

从第下标为n/2的节点开始，
依次讨论下标为n/2−1, n/2−2, . . . , 2, 1

怎样建堆？（小根）



建堆完毕……

建堆的时间复杂度为O(n)

详细证明见《算法导论》第77页，或者《算法分析与设计》第74页

调整堆(小根)

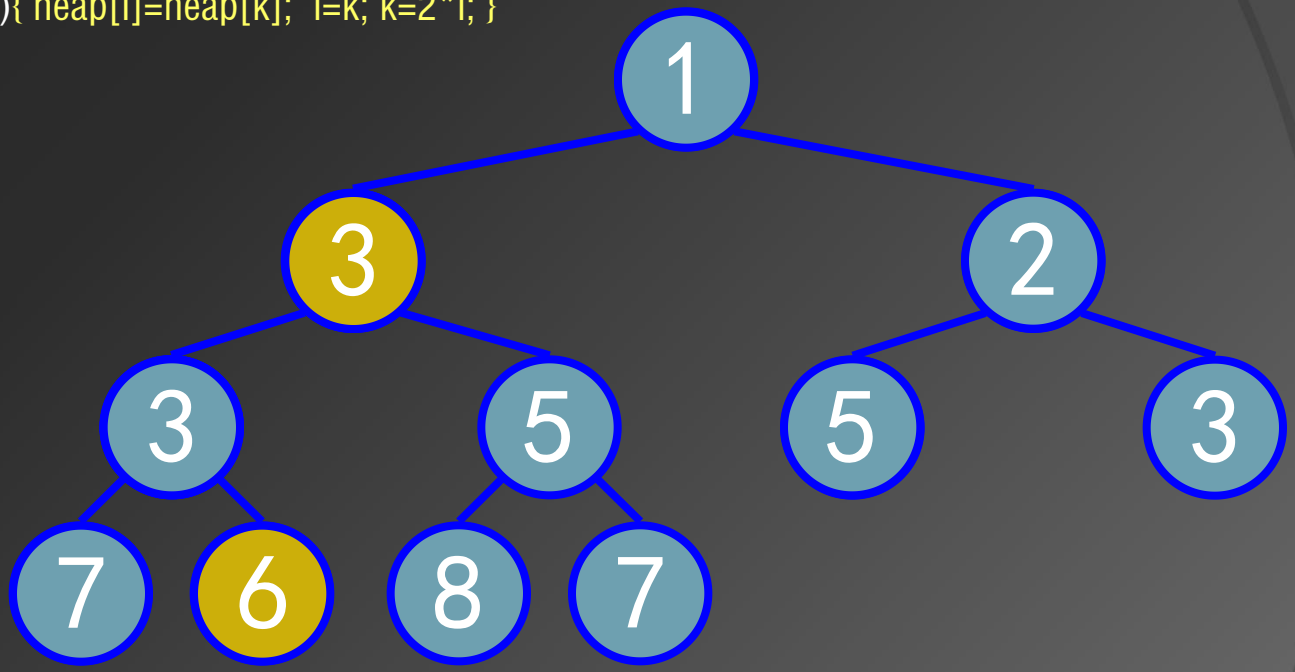
```
#define maxn 1000
int heap[maxn];

void shift(int i,int m)          //把以i号节点(下标为i)为根的子树调整为堆，m为堆最后一个节点的下标(也就是数组的长度)
{
   int k,t;                      //k表示当前节点孩子的下标
   t=heap[i];  k=2*i;
   while(k<=m)
   {
       if((k<m)&&(heap[k]>heap[k+1]))k++;       //找出值最小的孩子的下标
       if(t>heap[k]){ heap[i]=heap[k];  i=k; k=2*i; }
       else break;     //结束循环                  //把值最小的孩子的值赋值给根
   }
  heap[i]=t;       //把根的值赋值交换给孩子
}
```

调整一次堆的时间复杂度在最坏情况下是$O(\log_2 n)$

建堆： `for(i=n/2;i>=1;i--)shift(i,n);`

```
void shift(int i,int m)
{
    int k,t;
    t=heap[i];  k=2*i;
    while(k<=m)
    {
        if((k<m)&&(heap[k]>heap[k+1]))k++;
        if(t>heap[k]){ heap[i]=heap[k];  i=k; k=2*i; }
        else break;
    }
  heap[i]=t;
}
```



例：NK宇宙班

题目描述：

CQNK中学高一年级总共有n(n<=500000)个学生。现在你有他们的“星际语”成绩单，要从中找出“星际语”成绩最好的m(m<=1000并且m<n)个学生组成宇宙班，请按由高到低的顺序打印出加入于周班学生的“星际语”成绩。

输入格式：

第一行，两个整数n和m

第二行，n个用空格间隔的整数，分别表示n个学生的“星际语”成绩

输出格式：

只有一行，m个空格间隔的整数，表示加入宇宙班学生的“星际语”成绩。

样例输入：

12 5

89 87 77 95 68 56 100 80 65 95 99 71

样例输出：

100 99 95 95 89

直接对n个数由大到小排序，然后取最大的m个数取出来，
时间复杂度：
普通排序(冒泡、选择、插入)：$O(n^2+m)$ //一定超时
快速排序:$O(n\log_2 n+m)$

有没有更快的方法?

1. 把n个数字建成堆 $O(n)$
2. 把堆顶节点的打印出来，删除堆顶元素，调整堆。$O(\log_2 n)$
   第2步重复m次。$O(m\log_2 n)$

总时间复杂度：$O(n+m\log_2 n)$

n最大为500000，$2^{19}=524288$,也就是$\log_2 n$的值不超过19

```
#define maxn 500001
using namespace std;
int heap[maxn],n,m;
void shift(int i,int len)     //调整成大根堆
{     int t=heap[i],k=2*i;
      while(k<=len)
      {     if((k<len)&&(heap[k]<heap[k+1]))k++;       //k记录值最大的孩子的编号（大根堆）
            if(t<heap[k]){ heap[i]=heap[k];  i=k;  k=2*i; }
            else break;
      }
      heap[i]=t;
}
int main()
{     int i;
      scanf("%d%d",&n,&m);
      for(i=1;i<=n;i++)scanf("%d",&heap[i]);
      //建堆
      for(i=n/2;i>=1;i--)shift(i,n);
      //选出最大的m个数字
      for(i=1;i<=m;i++)
      {     printf("%d ",heap[1]);    //打印堆顶元素的值，即取出最大元素
            heap[1]=heap[n];          //把最后一个节点移到堆顶，相当于把原来堆顶节点删了
            n--;                      //删除堆顶节点后，总节点数减1
            shift(1,n);               //重新调整堆，把剩余节点中最大的值调到堆顶
      }
      return 0;
}
```

堆排序

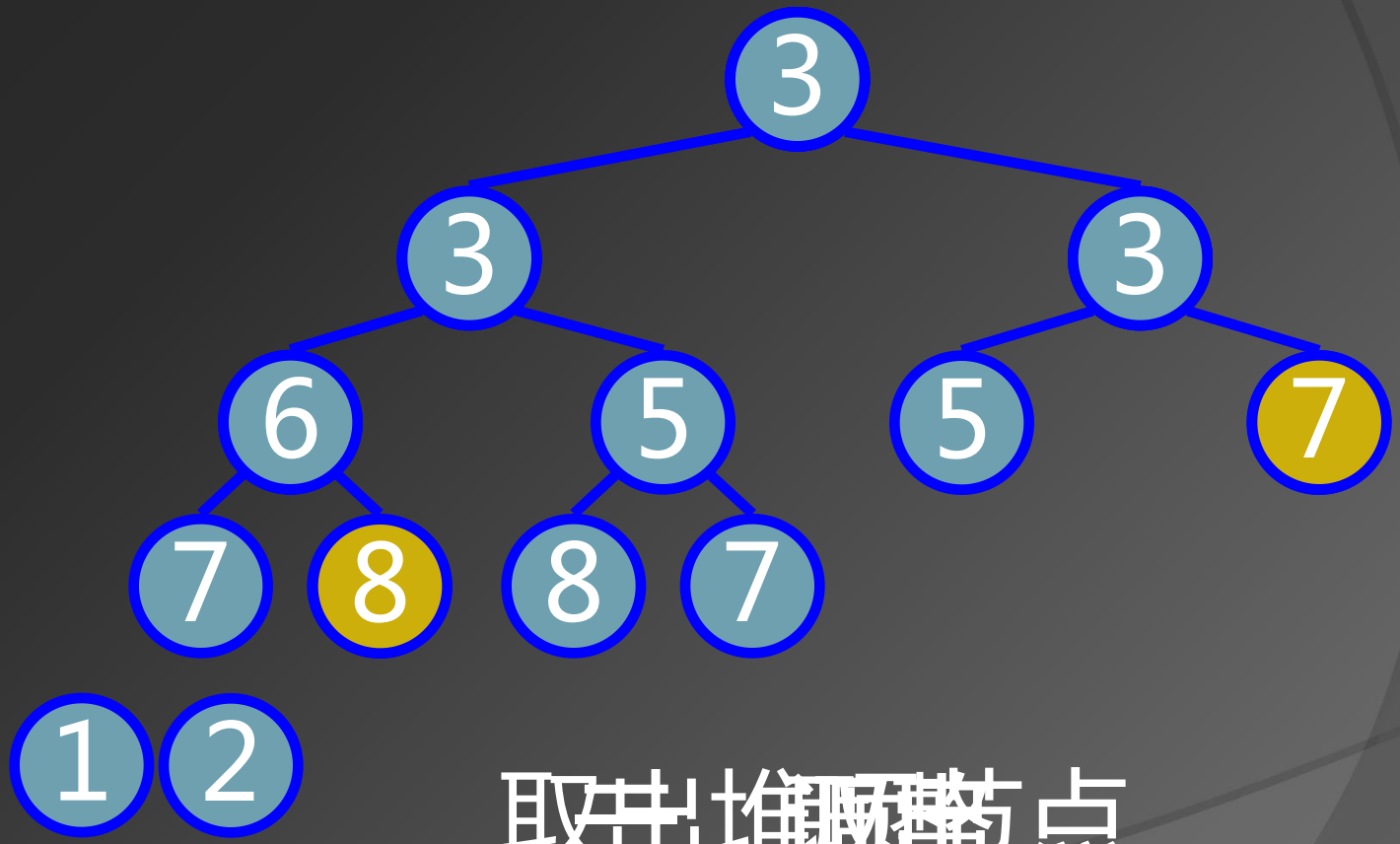
堆排序的基础——选择排序

堆排序的实质是利用二叉堆优化选择排序

堆排序步骤：

- 建堆。将所有节点布置成堆结构
- 取出堆顶节点
- 把堆尾的节点移动至顶部，调整此堆
- 跳转至2步骤，直至堆为空

堆排序——选择与维护



取出堆顶节点

排序

```
for(j=n;j>=2;j--)
{
    t=heap[1];
    heap[1]=heap[ j];
    a[ j]=t;        //heap[j]=t;
    shift(1,j-1);
}
```

```
int heap[100],t,j,n;
void shift(int i,int m)
{
    int k,t;
    t=heap[i];  k=2*i;
    while(k<=m)
    {
        if((k<m)&&(heap[k]>heap[k+1]))k++;
        if(t>heap[k]){ heap[i]=heap[k];i=k; k=2*i; } else break;
    }
    heap[i]=t;
}
```

```
int main()
{
      cin>>n; for(j=1;j<=n;j++) cin>>heap[ j];
      for(j=n/2; j>=1;j--)shift(j,n);
      for(j=n;j>=2;j--)
      {
            t=heap[1];
            heap[1]=heap[ j];
            heap[ j]=t;
            shift(1,j-1);
       }
       for(j=1;j<=n;j++)cout<<heap[j]<<" ";
       return 0;
}
```

堆的操作

```
//堆的向下调整
void ShiftDown(int i,int m)
{
    int k,t;
    t=heap[i];  k=2*i;
    while(k<=m)
    {
        if((k<m)&&(heap[k]>heap[k+1]))k++;
        if(t>heap[k]){ heap[i]=heap[k];i=k; k=2*i; } else break;
    }
    heap[i]=t;
}
```

```
//堆的向上调整
void ShiftUp(int x)//从编号为x的元素往傻上
{
    temp=Heap[x];
    for(i=x/2;i>0&&temp<Heap[i];i=i/2)
    {  Heap[x]=Heap[i];  x=i; }
    Heap[x]=temp;
}
```

```
//删除堆顶元素
void Del( )
{
     Heap[1]=Heap[n]; n--;
     if(n>0)ShiftDown(1,n);

}
```

```
//添加值为x的元素
void Insert( int x )
{
     n++; Heap[n]=x;
     ShiftUp(n);

}
```

特点：　不稳定

时间复杂度最坏情况下不超过

$O(n\log_2^n)$

课后练手：何老板捡钻石

题目描述：

“欢迎来安哥拉观光，运气好能捡到钻石，运气不好就踩中地雷"，看了这则旅游广告，何老板决定去安哥拉碰碰运气。到了安哥拉才发现有个坑爹的规定：游客最多只能带走n颗钻石，否则就视为走私。

何老板运气很好，他很快就搜集齐了n颗钻石，他把它们编号1到n放进了箱子。在回机场的路上，何老板发现路边还可以零星的捡到一些钻石。沿路何老板总共发现了m颗钻石，他把它们编号为n+1到n+m。何老板是个聪明人，他只会带走较重的钻石，如果捡起的钻石比箱子里的都要轻，何老板就直接把它扔掉，否则他就把箱子中最轻的钻石扔了，把新捡的放进箱子。请问何老板沿途把哪些钻石从箱子里扔了出去，请按先后顺序打印出被扔出的钻石的编号。（假定每颗钻石的重量都不同，不超过int范围）。

输入格式：

第一行，两个空格间隔的整数n和m，（n<=20000, m<=100000）

第二行，n个空格间隔的整数，表示已装到箱子中的n颗钻石的重量

第三行，m个空格间隔的整数，表示沿途捡到的m颗钻石的重量

输出格式：

只有一行：若干个空格间隔的整数，表示从箱子里扔出的钻石的编号

样例输入：

5 5

8 5 9 3 7

4 2 1 15 8

样例输出：

4 6 2