001、排序最大的i个数

> O（nlgi）

i次最大堆

- 建堆O（n）
- max_extract：i*lgn

先找第i大

- O（n）线性时间分治算法找第i大
- partition
- 排序i：i*lgi

n次i元最小堆

- 前i个元素建最小堆
- 遍历n，维持i元最小堆
- max_extract：i*lgi

002、最大堆第k大

不变量：下一目标范围

第l+1，一定是前l个的孩子

- O（klgk）

003、sorted A并B 找第k大(ps5-6)

二分排除

004、最大政体寻找（ps3-5）

二分查找变形：二元匹配

原因：最大政体人数过半

005、分治--最大子数组问题（ps3-6）

不变量：4个上传子数组

We only need to maintain four values for an interval $A[l, \cdots, r]$

- sum : sum of elements of $A[l, \cdots, r]$
- maxl : maximum subarray of $A[l, \cdots, r]$ which starts from l
- maxr : maximum subarray of $A[l, \cdots, r]$ which ends at r
- ans : maximum subarray of $A[l, \cdots, r]$

Algorithm 4 FIND-MAXIMUM-SUBARRAY

```
function FIND-MAXIMUM-SUBARRAY(A, low, high)
    if high == low then
        return (a[low], a[low] , a[low], a[low])
    else
        mid = ⌊ (low + high) / 2 ⌋;
        (lAns, lMaxl, lMaxr, lSum) = FIND-MAXIMUM-SUBARRAY(A, low, mid)
        (rAns, rMaxl, rMaxr, rSum) = FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)
        ans = max(lMaxr + rMaxl, lAns, rAns)
        maxl = max(lMaxl, lSum + rMaxl)
        maxr = max(rMaxr, rSum + lMaxr)
        sum = lSum + rSum
        return (ans, maxl, maxr, sum)
```

006、合并k 个 sorted list

天际堆

> 取每个堆顶，维护k元最小堆

- O（nlgk）

```
Merge_k(lst[]):传入k个list
list L//初始化列表
min_heap q//初始化最小堆
for (i=0 to k-1) do
    q.push(make_pair(lst[i].front,i))//组建二元对
    //二元对为了从刚被拿走的那个list里补充
    lst[i].pop_front()//踢出首元素
while(!q.empty())  do
    temp=q.top(),key=temp.first,id=temp.second
    q.pop()
    L.push_back(key)
```

```
12      if(!lst[id].empty())//小列表未被掏空
13          q.push(make_pair(lst[id].front,id))
14          lst[i].pop_front()
15  return L
```