面试题 40：最小的 k 个数

题目：输入 n 个整数，找出其中最小的 k 个数。例如，输入 4、5、1、6、2、7、3、8 这 8 个数字，则最小的 4 个数字是 1、2、3、4。

解法一 排序 O(nlogn)

解法二 随机快排 O(n)

比第k个数字小的放左边，大的放右边

要取得[a,b)的随机整数，使用(rand() % (b-a))+ a;

要取得[a,b]的随机整数，使用(rand() % (b-a+1))+ a;

要取得(a,b]的随机整数，使用(rand() % (b-a))+ a + 1;

通用公式:a + rand() % n；其中的a是起始值，n是整数的范围。

要取得a到b之间的随机整数，另一种表示：a + (int)b * rand() / (RAND_MAX + 1)。

要取得0～1之间的浮点数，可以使用rand() / double(RAND_MAX)。

```
vector<int> getLeastk(vector<int> nums,int k)
{
    vector<int> res;
    if(nums.empty() || k<0)
        return res;
    int start=0,end=nums.size()-1;
    int index=partition(nums,start,end);
    while(index!=k-1)
    {
        if(index>k-1)
        {
            end=index-1;
            index=partition(nums,start,end);
        }
        else
        {
            start=index+1;
            index=partition(nums,start,end);
        }
    }
    for(int i=0;i<k;i++)
        res[i]=nums[i];
    return res;
}
```

```
int partition(vector<int> nums,int start,int end)
{
  int small=start-1;
  int index=(rand()%(end-start+1))+start;
  swap(&nums[index],&nums[end]);
  for(index=start;index<end;index++)
  {
     if(nums[index]<nums[end])
     {
        small++;
        swap(nums[index],nums[small]);
     }
  }
  small++;
  swap(&nums[small],&nums[end]);
  return small;
}
```

解法三 最大堆 O(nlogk)

适合处理海量数据，最大堆可以在O(1)时间内得到已有的k个数的最大值，但需要O(logk)时间完成删除及插入操作

1. 如果容器未满，直接插入
2. 如果容器已满，当前值小于堆顶，则把堆顶替换为当前值
3. 如果容器已满，当前值大于堆顶，不处理

```
class Solution {
public:
   vector<int> GetLeastNumbers_Solution(vector<int> input, int k) {
      vector<int> res;
      if(input.empty() || k<=0 || k>input.size())
         return res;
      priority_queue<int> heap;
      for(int i=0;i<input.size();i++)
      {
         if(heap.size()<k)
            heap.push(input[i]);
         else
         {
            if(input[i]<heap.top())
            {
               heap.pop();
               heap.push(input[i]);
            }
         }
```

```
        }
        while(!heap.empty())
        {
            res.push_back(heap.top());
            heap.pop();
        }

        reverse(res.begin(),res.end());
        return res;
    }
};

//牛客

class Solution {
public:
    vector<int> GetLeastNumbers_Solution(vector<int> input, int k) {
        vector<int> res;
        if(input.empty() || k>input.size())
            return res;
        multiset<int,greater<int>> m;
        for(int i=0;i<input.size() && i<k;i++)
            m.insert(input[i]);
        for(int i=k;i<input.size();i++)
        {
            int num=input[i];
            if(num<*m.begin())
            {
                m.erase(m.begin());
                m.insert(num);
            }
        }

        while(!m.empty())
        {
            res.push_back(*m.begin());
            m.erase(m.begin());
        }
        return res;
    }
};

vector<int> getLeastk(vector<int> nums,int k)
{
    vector<int> res;
    int n=nums.size();
```

```
    if(nums.empty() || k<0 || k>n)
        return res;
    multiset<int,greater<int>> leastk;
    multiset<int,greater<int>>::iterator iterGreater;
    vector<int>::iterator iter=nums.begin();
    for(;iter!=nums.end();iter++)
    {
        if(leastk.size()<k)
            leastk.insert(*iter);
        else
        {
            iterGreater=leastk.begin();
            if(*iter<*(leastk.begin()))
            {
                leastk.erase(iterGreater);
                leastk.insert(*iter);
            }

        }
    }
    for(iterGreater=leastk.begin();iterGreater!=leastk.end();iterGreater++)
    {
        res.push_back(*iterGreater);
    }
    return res;
}
```