

剑指Offer（二十九）：最小的K个数

🕒 2017年12月22日 10:16:11 🗨 3 🌡 4,637 °C 📄 编辑



一、前言

本系列文章为《剑指Offer》刷题笔记。

刷题平台：牛客网

书籍下载：共享资源

二、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。

1、思路

最简单的方法就是先排序，然后在遍历输出最小的K个数，方法简单粗暴。

2、代码

C++:

```
1 class Solution {
2 public:
3     vector<int> GetLeastNumbers_Solution(vector<int> input, int k) {
4         vector<int> result;
5         if(input.empty() || k > input.size()){
6             return result;
7         }
8         sort(input.begin(), input.end());
9         for(int i = 0; i < k; ++i){
10             result.push_back(input[i]);
11         }
12         return result;
13     }
14 };
```

3、优化

上述算法的时间复杂度为 $O(n^2)$ ，速度慢。另一种 $O(n\log k)$ 的算法是基于堆排序的，特别适合处理海量数据。

我们可以先创建一个大小为k的数据容器来存储最小的k个数字，接下来我们每次从输入的n个整数中读入一个数。如果容器中已有的数字少于k个，则直接把这次读入的整数放入容器之中；如果容器已经有k个数字了，也就是容器满了，此时我们不能再插入新的数字而只能替换已有的数字。找出这已有的k个数中的最大值，然后拿这次待插入的整数和最大值进行比较。如果待插入的值比当前已有的最大值小，则用这个数替换当前已有的最大值；如果待插入的值比当前已有的最大值还要大，那么这个数不可能是最小的k个整数之一，于是我们可以抛弃这个整数。

因此当容器满了之后，我们要做3件事情：一是在k个整数中找到最大数；二是有可能在这个容器中删除最大数；三是有可能要插入一个新的数字。如果用一个二叉树来实现这个数据容器，那么我们在 $O(\log k)$ 时间内实现这三步操作。因此对于n个输入数字而言，总的时间效率就是 $O(n\log k)$ 。

C++:

```
1 class Solution {
2 public:
3     vector<int> GetLeastNumbers_Solution(vector<int> input, int k) {
4         vector<int> result;
```

```

5     int length = input.size();
6     if(length <= 0 || k <= 0 || k > length){
7         return result;
8     }
9
10    for(int i = 0; i < input.size(); i++){
11        if(result.size() < k){
12            result.push_back(input[i]);
13        }
14        else{
15            for(int j = k / 2; j >= 0; j--){
16                HeadAdjust(result, j, k);
17            }
18            for(int j = k - 1; j > 0; j--){
19                swap(result[0], result[j]);
20                HeadAdjust(result, 0, j);
21            }
22            if(result[k-1] > input[i]){
23                result[k-1] = input[i];
24            }
25        }
26    }
27    return result;
28 }
29 private:
30 void HeadAdjust(vector<int> &input, int parent, int length){
31     int temp = input[parent];
32     int child = 2 * parent + 1;
33     while(child < length){
34         if(child + 1 < length && input[child] < input[child+1]){
35             child++;
36         }
37         if(temp >= input[child]){
38             break;
39         }
40         input[parent] = input[child];
41
42         parent = child;
43         child = 2 * parent + 1;
44     }
45     input[parent] = temp;
46 }
47 };

```

对于上述代码，我们还可以进一步优化，不是每次循环都需要重新排序的，只有在更新了容器的数据之后，才需要重新排序。

进一步优化：

```

1  class Solution {
2  public:
3      vector<int> GetLeastNumbers_Solution(vector<int> input, int k) {
4          vector<int> result;
5          int length = input.size();
6          bool change = true;
7          if(length <= 0 || k <= 0 || k > length){
8              return result;
9          }
10
11         for(int i = 0; i < input.size(); i++){
12             if(result.size() < k){
13                 result.push_back(input[i]);
14             }
15             else{
16                 if(change == true){
17                     for(int j = k / 2; j >= 0; j--){
18                         HeadAdjust(result, j, k);
19                     }
20                     for(int j = k - 1; j > 0; j--){
21                         swap(result[0], result[j]);
22                         HeadAdjust(result, 0, j);
23                     }
24                     change = false;
25                 }
26                 if(result[k-1] > input[i]){
27                     result[k-1] = input[i];
28                     change = true;
29                 }
30             }
31         }
32         return result;
33     }
34 private:
35     void HeadAdjust(vector<int> &input, int parent, int length){
36         int temp = input[parent];
37         int child = 2 * parent + 1;
38         while(child < length){
39             if(child + 1 < length && input[child] < input[child+1]){
40                 child++;
41             }
42             if(temp >= input[child]){
43                 break;
44             }
45             input[parent] = input[child];
46
47             parent = child;
48             child = 2 * parent + 1;
49         }
50         input[parent] = temp;
51     }
52 };

```

C+

Python：

用数组的方法：


Pytho

```
1 # -*- coding:utf-8 -*-
2 class Solution:
3     def GetLeastNumbers_Solution(self, tinput, k):
4         if len(tinput) < k:
5             return []
6         tmp = sorted(tinput[:k])
7         for each in tinput[k:]:
8             index = k - 1
9             flag = False
10            while index >= 0 and tmp[index] > each:
11                index -= 1
12                flag = True
13            if flag == True:
14                tmp.insert(index+1, each)
15                tmp.pop()
16        return tmp
```

用堆排序的方法：

Pytho

```
1 # -*- coding:utf-8 -*-
2 class Solution:
3     def HeadAdjust(self, input_list, parent, length):
4         temp = input_list[parent]
5         child = 2 * parent + 1
6         while child < length:
7             if child + 1 < length and input_list[child] < input_list[child+1]:
8                 child += 1
9             if temp >= input_list[child]:
10                 break
11             input_list[parent] = input_list[child]
12             parent = child
13             child = 2 * parent + 1
14             input_list[parent] = temp
15
16     def GetLeastNumbers_Solution(self, tinput, k):
17         # write code here
18         res = []
19         length = len(tinput)
20         change = True
21         if length <= 0 or k <= 0 or k > length:
22             return res
23         res = tinput[:k]
24
25         for i in range(k, length+1):
26             if change == True:
27                 for j in range(0, k//2+1)[::-1]:
28                     self.HeadAdjust(res, j, k)
29                 for j in range(1, k)[::-1]:
30                     res[0], res[j] = res[j], res[0]
31                     self.HeadAdjust(res, 0, j)
32                 chage = False
33             if i != length and res[k-1] > tinput[i]:
34                 res[k-1] = tinput[i]
35                 chage = True
36         return res
```



微信公众号

分享技术，乐享生活：微信公众号搜索「JackCui-AI」关注一个在互联网摸爬滚打的潜行者。

人正是因为度过了一段阴暗岁月，才会更渴求黎明的曙光。--- 岸本齐史《火影忍者》