# 题目

给你一个字符串数组nums和一个整数k。nums中的每个字符串都表示一个不含前导零的整数。

返回nums中表示第k大整数的字符串。

注意：重复的数字在统计时会视为不同元素考虑。例如，如果nums是["1","2","2"]，那么"2"是最大的整数，"2"是第二大的整数，"1"是第三大的整数。

示例 1：

输入：nums = ["3","6","7","10"], k = 4

输出："3"

解释：

nums 中的数字按非递减顺序排列为 ["3","6","7","10"]

其中第 4 大整数是 "3"

示例 2：

输入：nums = ["2","21","12","1"], k = 3

输出："2"

解释：

nums 中的数字按非递减顺序排列为 ["1","2","12","21"]

其中第 3 大整数是 "2"

示例 3：

输入：nums = ["0","0"], k = 2

输出："0"

解释：

nums 中的数字按非递减顺序排列为 ["0","0"]

其中第 2 大整数是 "0"

提示：

1 <= k <= nums.length <= 104

1 <= nums[i].length <= 100

nums[i] 仅由数字组成

nums[i] 不含任何前导零

# 分析

## 方法一：最大堆

思路：

对于K值问题，都是采用优先队列。这个与LeetCode 215的区别在，215是直接比较int类型，比较函数直接采用greater即可。这里是比较string类型，需要自己编写仿函数或函数表达式。

代码：

class Solution {

public:

struct compare

{

bool operator()(string s1,string s2)

{

if(s1.size()!=s2.size()) return s1.size()>s2.size();

return s1 > s2;

}

};

public:

string kthLargestNumber(vector<string>& nums, int k) {

priority\_queue<string,vector<string>,compare> pri\_que;

for(int i=0;i<nums.size();i++)

{

pri\_que.push(nums.at(i));

if(i>=k)

pri\_que.pop();

}

return pri\_que.top();

}

};

或：

class Solution {

public:

string kthLargestNumber(vector<string>& nums, int k) {

auto compare = [](string s1, string s2) {

if(s1.size()!=s2.size()) return s1.size()>s2.size();

return s1 > s2;

};

// 不使用仿函数，直接使用函数表达式（函数式编程）

priority\_queue<string,vector<string>,decltype(compare)> pri\_que(compare);

for(int i=0;i<nums.size();i++)

{

pri\_que.push(nums.at(i));

// 在调用push的时候直接使用compare插入，同理pop

if(i>=k)

pri\_que.pop();

}

return pri\_que.top();

}

};