# 题目

合并 k 个排序链表，返回合并后的排序链表。请分析和描述算法的复杂度。

**示例:**

输入:

[

  1->4->5,

  1->3->4,

  2->6

]

输出: 1->1->2->3->4->4->5->6

# 分析

## 方法一：顺序合并

## 方法二：分治合并

## 方法三：堆/优先队列

**思路：推荐该方法，具有普适性。**

我们需要维护当前每个链表没有被合并的元素的最前面一个，k个链表就最多有k个满足这样条件的元素，每次在这些元素里面选取val属性最小的元素合并到答案中。在选取最小元素的时候，我们可以用优先队列来优化这个过程。

**代码：**

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode(int x) : val(x), next(NULL) {}

\* };

\*/

class Solution {

public:

// 小根堆的回调函数

struct compare{

bool operator()(ListNode \*a,ListNode \*b)

{

return a->val > b->val;

}

};

ListNode\* mergeKLists(vector<ListNode\*>& lists) {

priority\_queue<ListNode\*, vector<ListNode\*>, compare> pri\_que;

for(auto node : lists)

{

if(node) pri\_que.push(node);//构造堆,操作堆就是最K值问题

}

ListNode dummy(-1);

ListNode \*tail = &dummy;//哑结点

while(!pri\_que.empty())

{

ListNode\* top = pri\_que.top();//此时取出来的就是排序的

pri\_que.pop();

tail->next = top;

tail = top;

if(top->next) pri\_que.push(top->next);

}

return dummy.next;

}

};