一、题目说明

这个题目是23. Merge k Sorted Lists，归并k个有序列表生成一个列表。难度为Hard，实际上并不难，我一次提交就对了。

二、我的解答

就是k路归并，思路很简单，实现也不难。

```
#include<iostream>
#include<vector>
using namespace std;
struct ListNode {
    int val;
    ListNode *next;
    ListNode(int x) : val(x), next(NULL) {}
};

class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*>& lists){
        ListNode dummy(-1);
        ListNode *p,*pResult,*cur;
        if(lists.size()<=0) return NULL;
        for(int t=0;t<lists.size();t++){
            p = lists[t];
            pResult = &dummy;
            while(p !=NULL){
                while(pResult->next!=NULL && pResult->next->val < p->val){
                    pResult = pResult->next;
                }
                cur = new ListNode(p->val);
                cur->next = pResult->next;
                pResult->next = cur;
                p = p->next;
            }
        }
        return dummy.next;
    }
};
int main(){
    Solution s;
    ListNode* l1, *l2, *l3, *p;

    // init l1
    l1 = new ListNode(5);
    p = new ListNode(4);
    p->next = l1;
    l1 = p;
    p = new ListNode(1);
    p->next = l1;
    l1 = p;

    // init l2
    l2 = new ListNode(4);
    p = new ListNode(3);
    p->next = l2;
    l2 = p;
    p = new ListNode(1);
    p->next = l2;
    l2 = p;

    // init l1
    l3 = new ListNode(6);
    p = new ListNode(2);
    p->next = l3;
    l3 = p;

    vector<ListNode*> lists;
    lists.push_back(l1);
    lists.push_back(l2);
    lists.push_back(l3);

    ListNode* r = s.mergeKLists(lists);
    while(r != NULL){
        cout<<r->val<<" ";
        r = r->next;
    }
    return 0;
}
```

不过，性能一般：

```
Runtime: 172 ms, faster than 21.46% of C++ online submissions for Merge k Sorted
Lists.
Memory Usage: 12.8 MB, less than 5.95% of C++ online submissions for Merge k
Sorted Lists.
```

三、优化措施

上面的实现，之所以性能不足，在于一次归并一个队列，用的是插入排序。其实n路归并，可以用优先级队列priority_queue一次实现的。

```
class Solution {
    struct CompareNode {
        bool operator()(ListNode* const & p1, ListNode* const & p2) {
            // return "true" if "p1" is ordered before "p2", for example:
            return p1->val > p2->val;
           //Why not p1->val <p2->val; ??
        }
    };
public:
    ListNode *mergeKLists(vector<ListNode *> &lists) {

        ListNode dummy(0);
        ListNode* tail=&dummy;

        priority_queue<ListNode*,vector<ListNode*>,CompareNode> queue;

        for (vector<ListNode *>::iterator it = lists.begin(); it != lists.end();
++it){
```

```
            if (*it)
            queue.push(*it);
        }
        while (!queue.empty()){
            tail->next=queue.top();
            queue.pop();
            tail=tail->next;

            if (tail->next){
                queue.push(tail->next);
            }
        }

        return dummy.next;
    }
};
```