```
/*
给定两个排序后的数组 A 和 B，其中 A 的末端有足够的缓冲空间容纳 B。 编写一个方法，将 B 合并入 A 并排序。

初始化 A 和 B 的元素数量分别为 m 和 n。

示例:

输入:
A = [1,2,3,0,0,0], m = 3
B = [2,5,6],       n = 3

输出: [1,2,2,3,5,6]

通过次数8,838
提交次数15,576

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/sorted-merge-lcci
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析：

- 方法一：将B数组内容完全复制到A数组中，然后对A数组调用sort函数，缺点是默认了A的尝试$m+n$,而且不是考察目的；
- 方法二：先将A数组内容后移n个位置，然后从A的n位置和B的0位置开始归并过程，从小到大进行归并；
- 方法三: 直接从A的m-1和B的n-1从大到小归并.

方法一：C++_sort函数法

```
class Solution
{
    public:
        void merge(vector<int>& A, int m, vector<int>& B, int n)
        {

            for( int i = 0;i < n; i++ )
            {
                A[m+i] = B[i];
            }

            sort(A.begin(),A.end());
        }
};
/*
执行结果：
通过
显示详情
```

```
执行用时 :8 ms, 在所有 C++ 提交中击败了28.76% 的用户
内存消耗 :11.5 MB, 在所有 C++ 提交中击败了100.00%的用户
*/

```

方法二：C++_从小到大归并

```
class Solution
{
    public:
        void merge(vector<int>& A, int m, vector<int>& B, int n)
        {
            int i = 0;
            int j = 0;
            int k = 0;


            /* 拷贝*/
            for( i = m-1 ; i >= 0 ; i-- )
            {
                A[i+n] = A[i];
            }


            /*从小到大归并*/
            i = 0 ;
            j = 0 ;
            while(1)
            {
                if(i < m && j < n )
                {
                    if(A[n+i]<B[j])
                    {
                        A[k++] = A[n+i];
                        i++;
                    }
                    else
                    {
                        A[k++] = B[j];
                        j++;
                    }
                }
                else if(i < m )
                {
                    A[k++] = A[n+i];
                    i++;

                }
                else if(j < n )
                {
                        A[k++] = B[j];
                        j++;
                }
                else
                {
                    break;
```

```
                }

            }
        }
};

class Solution
{
    public:
        void merge(vector<int>& A, int m, vector<int>& B, int n)
        {

            for( int i = 0;i < n; i++ )
            {
                A[m+i] = B[i];
            }

            sort(A.begin(),A.end());
        }
};


/*
执行结果：
通过
显示详情
执行用时 :8 ms, 在所有 C++ 提交中击败了28.76% 的用户
内存消耗 :11.6 MB, 在所有 C++ 提交中击败了100.00%的用户
*/

```

方法三：C++_从大到小归并

```
class Solution
{
    public:
        void merge(vector<int>& A, int m, vector<int>& B, int n)
        {
            int i = m-1;
            int j = n-1;
            int k = m+n-1;



            /*从大到小归并*/
            while(1)
            {
                if(i >= 0 && j >= 0 )
                {
                    if(A[i]<B[j])
                    {
                        A[k--] = B[j--];
                    }
                    else
                    {
```

```
                        A[k--] = A[i--];
                    }
                }
                else if(i >= 0 )
                {
                    A[k--] = A[i--];
                }
                else if(j >= 0 )
                {
                        A[k--] = B[j--];
                }
                else
                {
                    break;
                }

            }
        }
};


/*
执行结果：
通过
显示详情
执行用时 :4 ms, 在所有 C++ 提交中击败了78.97% 的用户
内存消耗 :11.6 MB, 在所有 C++ 提交中击败了100.00%的用户
*/
```

AlimyBreak
2020.03.03(2020年第09周)