

```

1  /*
2  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。
3
4  初始化 A 和 B 的元素数量分别为 m 和 n。
5
6  示例：
7
8  输入：
9  A = [1,2,3,0,0,0], m = 3
10 B = [2,5,6],          n = 3
11
12 输出：[1,2,2,3,5,6]
13
14 通过次数8,838
15 提交次数15,576
16
17 来源：力扣（LeetCode）
18 链接：https://leetcode-cn.com/problems/sorted-merge-lcci
19 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。
20 */

```

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

```

1  class Solution
2  {
3      public:
4          void merge(vector<int>& A, int m, vector<int>& B, int n)
5          {
6
7              for( int i = 0; i < n; i++ )
8              {
9                  A[m+i] = B[i];
10             }
11
12             sort(A.begin(),A.end());
13         }
14     };
15     /*
16     执行结果：
17     通过
18     显示详情

```

```
19 执行用时 :8 ms, 在所有 C++ 提交中击败了28.76% 的用户
20 内存消耗 :11.5 MB, 在所有 C++ 提交中击败了100.00%的用户
21 */
22
```

方法二: C++_从小到大归并

```
1  class Solution
2  {
3      public:
4          void merge(vector<int>& A, int m, vector<int>& B, int n)
5          {
6              int i = 0;
7              int j = 0;
8              int k = 0;
9
10
11              /* 拷贝*/
12              for( i = m-1 ; i >= 0 ; i-- )
13              {
14                  A[i+n] = A[i];
15              }
16
17
18              /*从小到大归并*/
19              i = 0 ;
20              j = 0 ;
21              while(1)
22              {
23                  if(i < m && j < n )
24                  {
25                      if(A[n+i]<B[j])
26                      {
27                          A[k++] = A[n+i];
28                          i++;
29                      }
30                      else
31                      {
32                          A[k++] = B[j];
33                          j++;
34                      }
35                  }
36                  else if(i < m )
37                  {
38                      A[k++] = A[n+i];
39                      i++;
40                  }
41                  else if(j < n )
42                  {
43                      A[k++] = B[j];
44                      j++;
45                  }
46                  else
47                  {
48                      break;
49                  }
50              }
51          }
52      }
53  }
```

```

50         }
51     }
52 }
53 }
54 };
55
56 class Solution
57 {
58     public:
59         void merge(vector<int>& A, int m, vector<int>& B, int n)
60         {
61
62             for( int i = 0; i < n; i++ )
63             {
64                 A[m+i] = B[i];
65             }
66
67             sort(A.begin(), A.end());
68         }
69 };
70
71
72 /*
73 执行结果:
74 通过
75 显示详情
76 执行用时 :8 ms, 在所有 C++ 提交中击败了28.76% 的用户
77 内存消耗 :11.6 MB, 在所有 C++ 提交中击败了100.00%的用户
78 */
79
80

```

方法三: C++_从大到小归并

```

1  class Solution
2  {
3      public:
4          void merge(vector<int>& A, int m, vector<int>& B, int n)
5          {
6              int i = m-1;
7              int j = n-1;
8              int k = m+n-1;
9
10
11
12              /*从大到小归并*/
13              while(1)
14              {
15                  if(i >= 0 && j >= 0 )
16                  {
17                      if(A[i]<B[j])
18                      {
19                          A[k--] = B[j--];
20                      }
21                      else
22                      {

```

```
23         A[k--] = A[i--];
24     }
25 }
26 else if(i >= 0 )
27 {
28     A[k--] = A[i--];
29 }
30 else if(j >= 0 )
31 {
32     A[k--] = B[j--];
33 }
34 else
35 {
36     break;
37 }
38
39 }
40 }
41 };
42
43
44 /*
45 执行结果:
46 通过
47 显示详情
48 执行用时 :4 ms, 在所有 C++ 提交中击败了78.97% 的用户
49 内存消耗 :11.6 MB, 在所有 C++ 提交中击败了100.00%的用户
50 */
```