```
/*
给定两个有序整数数组 nums1 和 nums2，将 nums2 合并到 nums1 中，使得 num1 成为一个有序数组。

说明:

    初始化 nums1 和 nums2 的元素数量分别为 m 和 n。
    你可以假设 nums1 有足够的空间（空间大小大于或等于 m + n）来保存 nums2 中的元素。

示例:

输入:
nums1 = [1,2,3,0,0,0], m = 3
nums2 = [2,5,6],       n = 3

输出: [1,2,2,3,5,6]

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/merge-sorted-array
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：申请临时辅助空间实现常规的归并
- 方法二：由于$nums1$的长度已经足够长，我们可以考虑从大到小，从尾到头进行归并，这样就不需要额外申请空间了(效果貌似没有多大改善

方法1：C_常规归并

```
void merge( int*    nums1       ,
            int     nums1Size   ,
            int     m           ,
            int*    nums2       ,
            int     nums2Size   ,
            int     n
            )
{
    int* temp       =   (int*)malloc(nums1Size*sizeof(int)) ;   /*申请辅助空间用于归并*/
    int  num_count  =   0                                   ;
    int  i          =   0                                   ;
    int  j          =   0                                   ;

    while(1)
    {
        if((i<m)&&(j<n))
        {
            if(nums1[i]<nums2[j])
            {
                temp[num_count] = nums1[i];
```

```
                i++;
            }
            else
            {
                temp[num_count] = nums2[j];
                j++;
            }
            num_count++;
        }
        else if(i < m)
        {
            temp[num_count] = nums1[i];
            i++;
            num_count++;
        }
        else if(j < n)
        {
            temp[num_count] = nums2[j];
            j++;
            num_count++;
        }
        else
        {
            break;
        }
    }

    memcpy(nums1,temp,(m+n)*sizeof(int));
    free(temp);/*释放空间*/
    return ;
}
```

方法二:C_从尾巴开始归并

```
void merge( int*    nums1       ,
            int     nums1Size   ,
            int     m           ,
            int*    nums2       ,
            int     nums2Size   ,
            int     n
            )
{
    int  num_count  =   m+n-1   ;
    int  i          =   m-1     ;
    int  j          =   n-1     ;

    while(1)
    {
        if(  (i>=0)
           &&(j>=0)
        )
        {
            if(nums1[i]>nums2[j])
            {
```

```
                nums1[num_count] = nums1[i];
                i--;
            }
            else
            {
                nums1[num_count] = nums2[j];
                j--;
            }
            num_count--;
        }
        else if(i >=0)
        {
            if(num_count!=i)
            {
                nums1[num_count] = nums1[i];
            }
            i--;
            num_count--;
        }
        else if(j >= 0)
        {
            nums1[num_count] = nums2[j];
            j--;
            num_count--;
        }
        else
        {
            break;
        }
    }

    return ;
}

/*
执行结果:
通过
显示详情
执行用时 :16 ms, 在所有 C 提交中击败了8.52% 的用户
内存消耗 :7.3 MB, 在所有 C 提交中击败了75.70%的用户
*/
```

AlimyBreak
2019.08.16