```
/*
给定一个整数数组 nums 和一个目标值 target，请你在该数组中找出和为目标值的那 两个 整数，并返回他们的数组下标。

你可以假设每种输入只会对应一个答案。但是，你不能重复利用这个数组中同样的元素。

示例:

给定 nums = [2, 7, 11, 15], target = 9

因为 nums[0] + nums[1] = 2 + 7 = 9
所以返回 [0, 1]

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/two-sum
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:逐一遍历所有的双整数对,根据排列组合公式,最多需要比较C_n^2次,即时间复杂度为$O(n^2)$;
- 方法二:hash表,先把数组数据hash化(key为数组数据,value为数组索引),逐一遍历元素,由于要规避重复元素,就一边hash一边进行比较.
  - 若target-当前元素的hash值在表中已经存在,就直接存储数据进行返回;
  - 若target-当前元素的hash值在表中不存在,就把当前元素及其在数组中的索引保存到hash表中;
  - 循环遍历直到所有元素遍历完成,最差情况下,时间复杂度为$O(n)$,由于要额外申请hash表空间,最差情况下,空间复杂度为$O(n)$.
  - 要用到哈希表这种数据结构，使用C++语言比较合适

方法一:C_逐一遍历双整数对

```
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int* twoSum(    int*        nums        ,
                int         numsSize    ,
                int         target      ,
                int*        returnSize
            )
{
    int     i       =   0                               ;
    int     j       =   0                               ;
    int*    index   =   (int*)malloc(2*sizeof(int))     ;
    int     valid   =   0                               ;


    for(i = 0; i < numsSize ;i++)
    {
        for(j=i+1;j<numsSize;j++)
```

```
        {
            if(nums[i]+nums[j] == target)
            {
                index[0] = i;
                index[1] = j;
                valid = 1;
            }
        }
        if(valid==1)
        {
            break;
        }
    }

    if(valid == 1)
    {
        *returnSize = 2;
    }
    else
    {
        *returnSize = 0;
        free(index);
        index = NULL;
    }
    return index;
}

/*
执行结果:
通过
显示详情
执行用时 :268 ms, 在所有 C 提交中击败了33.00% 的用户
内存消耗 :7.6 MB, 在所有 C 提交中击败了45.87%的用户
*/
```

方法二: C++_hash表查询

```
class Solution
{
    public:
        vector<int> twoSum(vector<int>& nums, int target)
        {
            vector<int>         ret_val         ;
            map<int,int>        mii             ;
            int                 i       =   0   ;

            for(i=0;i<nums.length();i++)
            {
                if(mii.count(target-nums[i]) == 1)  /*已经有了一个*/
                {
                    ret_val.push_back(i);
                    ret_val.push_back(mii[target-nums[i]]);
                    break;
                }
                mii[nums[i]] = i;
            }
```

```
            return ret_val;
        }
};
/*
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98.87% 的用户
内存消耗 :10.2 MB, 在所有 C++ 提交中击败了28.47%的用户
*/
```

AlimyBreak
2019.08.26