01 Two Sum

vector<T>,序列式容器

构造函数：

void test\_vector\_constructor()

{

// 0. Create an empty vector v0

std::vector<int> v0;

assert(v0.empty());

// 1. Create a vector v1 with 3 elements of default value 0

std::vector<int> v1(3);

// 2. Create a vector v2 with 5 elements of value 2

std::vector<int> v2(5, 2);

// 3. Create a vector v3 with 3 elements of value 1 and with the allocator of vector v2

std::vector<int> v3(3, 1, v2.get\_allocator());

// 4. Create a copy, vector v4, of vector v2

std::vector<int> v4(v2);

// 5. Create a vector v5 by copying the range v4[\_First, \_Last)

std::vector<int> v5(v4.begin() + 1, v4.begin() + 3);

}

压入元素：使用空间末尾插入一个元素

vector<T>::push\_back(const T &x)

弹出元素：将使用空间最后一个元素删除

vector<T>::pop\_back()

当前使用空间:

vector<T>::size()

当前可用可用空间：

vector<T>::capacity()

迭代器:vector的指针

vector<T>::iterator

find():在区间内查找值的位置，并返回位置

find(区间开始指针，区间结束指针，值)

end():

return finish; //not end \_of\_storage

begin():

return start;

erase():

erase(iterator first, interator last);

erase(iterator position)

insert():

insert(iterator position, size\_type n, const T&x)

---------------------

Sort(); //快速排序

难度：Easy

问题描述：

给定一个整型数组作为输入，如果数组中的两个数相加能够得到指定目标结果，则返回两个数的索引。我们假定每个输入所得的结果唯一，即数组中只存在一对元素相加得到目标结果。同时假定输入数组中的元素不能使用两次。

例如：

Nums = [2, 7, 11, 15], target = 9,

因为nums[0] + nums[1] = 2 + 7 =9,

所以返回[0, 1].

解决思路1：

时间复杂度：O(nlgn) 空间复杂度：O(n)

采用双指针法，先将数组排序形成一个有序的区间。指针i,j分别指向头尾：

1. 当nums[i]+nums[j]>target时，j--;
2. 当nums[i] + nums[j] <target时，i++；
3. 直到nums[i]+nums[j] == target;

C++代码：

class Solution

{

public:

vector<int> twoSum(vector<int>& nums, int target)

{

vector<pair<int,int> > nums1;

for(int i = 0;i < nums.size();++i)

nums1.push\_back(make\_pair(nums[i],i));

sort(nums1.begin(),nums1.end());

int i = 0,j = nums1.size() - 1;

vector<int> ret;

while(i < j)

{

if(nums1[i].first + nums1[j].first == target)

{

ret.push\_back(nums1[i].second);

ret.push\_back(nums1[j].second);

return ret;

}

nums1[i].first +nums1[j].first < target ? ++i : --j;

}

}

};

解决思路2：

时间复杂度：O(n) 空间复杂度：O(n)

C++中提供了unordered\_map容器，unordered\_map中的元素没有按照它们的键值或映射值的任何顺序排序，而是根据它们的散列值组织成桶以允许通过它们的键值直接快速访问单个元素（具有常数平均时间复杂度），将先出现的元素存储在unordered\_map中，遍历数组，每次查找target - nums[i]是否存在即可。

C++代码：

class Solution

{

public:

vector<int> twoSum(vector<int>& nums, int target)

{

unordered\_map<int, int> m;

vector<int> res;

for (int i = 0; i < nums.size(); ++i) {

m[nums[i]] = i;

}

for (int i = 0; i < nums.size(); ++i) {

int t = target - nums[i];

if (m.count(t) && m[t] != i) {

res.push\_back(i);

res.push\_back(m[t]);

break;

}

}

return res;

}

};