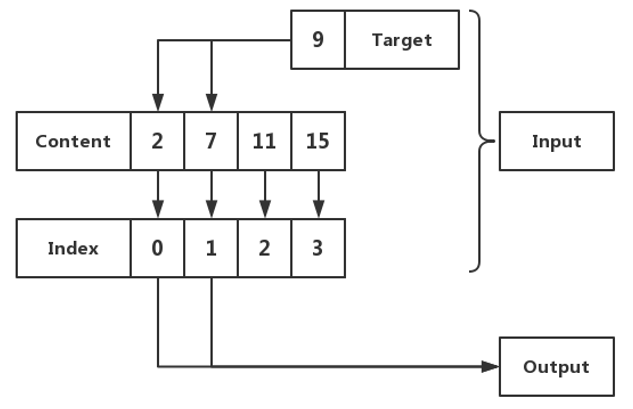
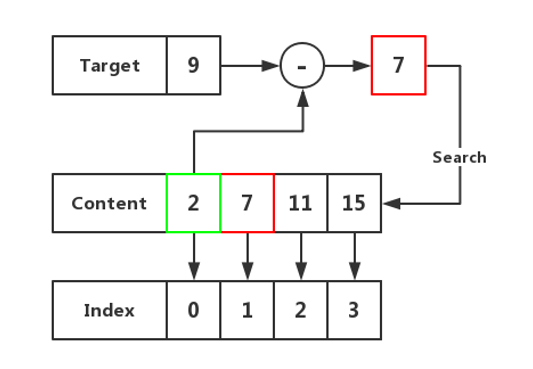
**问题图解重述:**



**解决方法1：**

**遍历输入数组，计算A=Target-Content[i]，搜索数组中是否有等于A的，返回i和搜索到A的下标。**

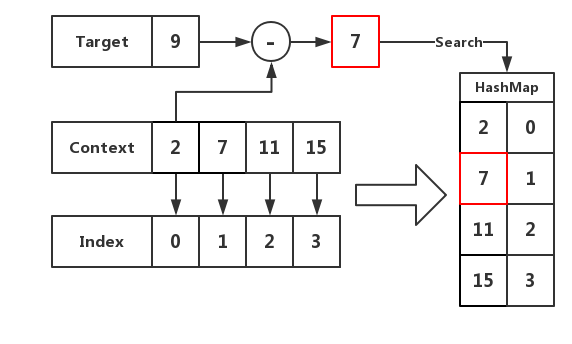


**此方法需要遍历原数组，最坏情况下数组最后两位为目标组成，每次取出一个数后又需要再次遍历数组，所以时间复杂度为O()，空间复杂度为O(1)**

**解决方法2：**

**使用Hashmap对查询步骤进行加速，Hashmap的查询速度为O(1)，相比方法1的查询步骤需要花费O(n)，快了很多。**

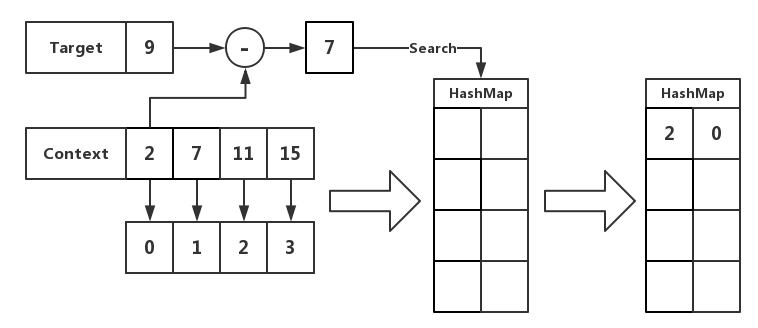
**具体步骤：建立content和index的Hash表，之后计算A=Target-Content[i]，使用Hash表对A进行查询并返回查询结果以及i.**

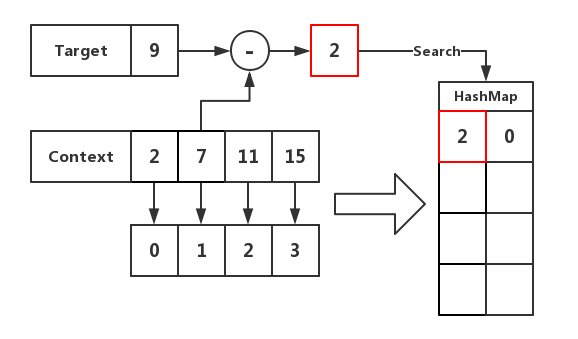


**此方法仅要遍历一遍原数组，由于需要先建立hash表，所以时间复杂度为O(n)+ O(n)= O(n)，空间复杂度为hash表所需要的储存空间O(n).**

**解决方法3：**

**对方法2的再优化，在建立hash表的同时进行搜索。**





方法3的C++代码：

#include <iostream>

#include <vector>

#include <unordered\_map>

using namespace std;

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

unordered\_map<int, int> m;

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

if(m.find(target-nums[i]) != m.end()) {

return vector<int>{ m[target-nums[i]], i};

}

m[nums[i]] = i;

}

}

int main()

{

vector<int> a{1,2,3,7,11,15};

vector<int> aa = twoSum(a,9);

for(auto it = aa.begin(); it != aa.end();++it)

cout<<\*it<<endl;

return 0;

}

Unordered\_map 操作举例：

#include <iostream>

#include <string>

#include <unordered\_map>

using namespace std;

void display(unordered\_map<string,double> myhash\_map,string str)

{

cout << str << endl;

for (auto& x: myhash\_map)

cout << x.first << ": " << x.second << endl;

cout << endl;

}

int main ()

{

unordered\_map<string,double> myhash\_map = {{"提亚娜",2.0},{"卡莲",1.5}};

unordered\_map<string,double> myhash\_map2 = {{"提亚娜",2.0},{"卡莲",1.5}};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*插入\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

pair<string,double> new\_Valkyrja ("符华",2.3);

myhash\_map.insert (new\_Valkyrja); // 复制插入

myhash\_map.insert (make\_pair<string,double>("姬子",6.0)); // 移动插入

myhash\_map.insert ({{"八重樱",0.8},{"德莉莎",0.1}}); // 初始化数组插入(可以用二维一次插入多个元素，也可以用一维插入一个元素)

myhash\_map["舰长"] = 10.0; //数组形式插入

myhash\_map

display(myhash\_map,"myhash\_map contains:");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*查找\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unordered\_map<string,double>::const\_iterator got = myhash\_map.find ("八重樱");

if ( got == myhash\_map.end() )

cout << "not found";

else

cout << "found "<<got->first << " is " << got->second<<"\n\n";

got = myhash\_map.find ("女王");

if ( got == myhash\_map.end() )

cout << "not found";

else

cout << "found "<<got->first << " is " << got->second<<"\n\n";

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*修改\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

myhash\_map.at("舰长") = 9.0;

myhash\_map["德莉莎"] = 3.0;

display(myhash\_map,"After modify myhash\_map contains:");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*擦除/干死舰长\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

myhash\_map.erase(myhash\_map.begin()); //通过位置

myhash\_map.erase("舰长"); //通过key

display(myhash\_map,"After erase myhash\_map contains:");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*交换(交换两个hashmap的值)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

myhash\_map.swap(myhash\_map2);

display(myhash\_map,"After swap with myhash\_map2, myhash\_map contains:");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*清空\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

myhash\_map.clear();

display(myhash\_map,"After clear, myhash\_map contains:");

return 0;

}

