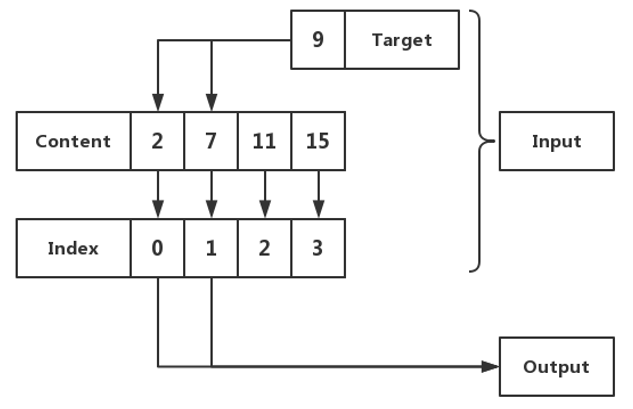
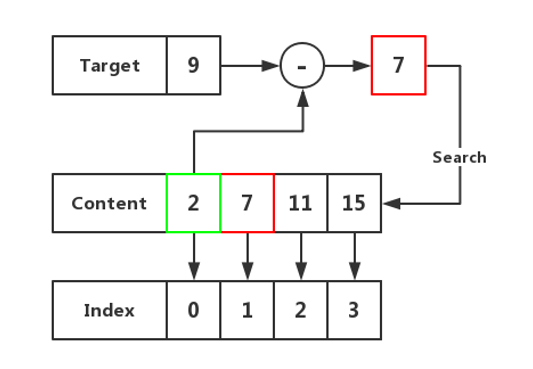
**问题图解重述:**



**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。**

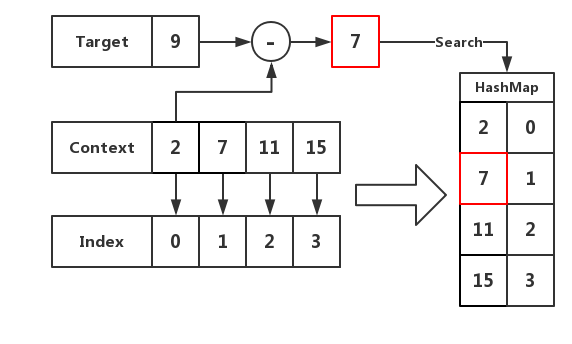


**此方法需要遍历原数组，最坏情况下数组最后两位为目标组成，每次取出一个数后又需要再次遍历数组，所以时间复杂度为O()，空间复杂度为O(1)**

**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。**

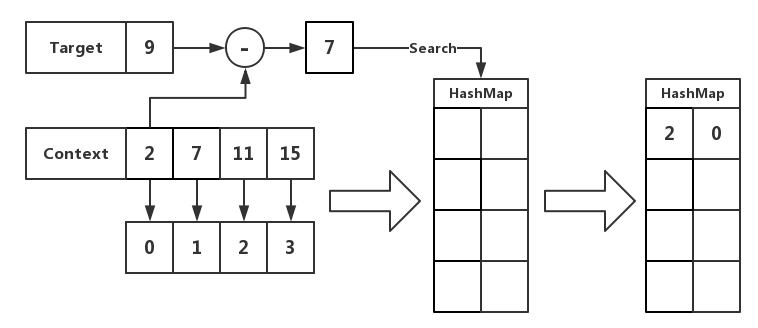
**具体步骤：建立content和index的Hash表，之后计算A=Target-Content[i]，使用Hash表对A进行查询并返回查询结果以及i.**

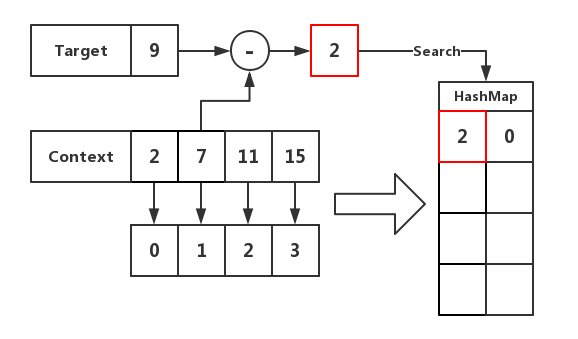


**此方法仅要遍历一遍原数组，由于需要先建立hash表，所以时间复杂度为O(n)+ O(n)= O(n)，空间复杂度为hash表所需要的储存空间O(n).**

**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;

}