**题目描述：**Given an array *S* of *n* integers, are there elements *a*, *b*, *c* in *S* such that *a* + *b* + *c* = 0? Find all unique triplets in the array which gives the sum of zero.

Note: Elements in a triplet (*a*,*b*,*c*) must be in non-descending order. (ie, *a* ≤ *b* ≤ *c*).

The solution set must not contain duplicate triplets.

For example, given array S = {-1 0 1 2 -1 -4},

A solution set is:

(-1, 0, 1)

(-1, -1, 2)

**my solution:**

class Solution {

public:

vector<vector<int>> threeSum(vector<int>& nums) {

int size = nums.size();

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

int low = 0, high = size - 1, sum = 0, target, mid;

if (size< 3 || nums[0]>0 || nums[high]<0)return triplets;

int data\_pre = nums[0] - 1;

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

if (data\_pre!=nums[i]&&nums[i]**<=**0){ //**等号包括{0,0,0}的情况**

low = i+1;

high = size - 1;

int low\_pre=nums[0]-1, high\_pre=nums[0]-1;

while (high>low){

sum = nums[low] + nums[high];

if (abs(sum)>abs(nums[i]))high--;

else if(abs(sum)<abs(nums[i])) low++;

else{

if (nums[low]!=low\_pre&&nums[high]!=high\_pre&&sum+nums[i]==0){

**triplets.push\_back({ nums[i], nums[low], nums[high] });**

**low\_pre = nums[low];**

**high\_pre = nums[high];**

}

low++;

high--;

}

}

}

**data\_pre = nums[i]; //当前值作为历史值**

}

return triplets;

}

vector<vector<int> >triplets;

};

**总结：**

1. 先将数组排序，{-5,-4,-4,-3,-2-1,0,0,0,1,2,2,3,4,5,5,9};
2. 因为三个数的和等于0,，必然包括一个数为负数(或一个数为正数)，在搜索的时候只需要搜索非负数或者非正数就ok.
3. 在nums[i]=-5的时候则可以找到包含{-5}的所有三元组，那么在以后的**三元组中不会出现-5**，所以-5在以后的nums[i]的查找中不会再被纳入搜索范围.