LOBACHEVSKY
UNIVERSITY

# Advanced C++

4. Task: 3Sum

14.11.2017

Sidnev A.A.

# 3Sum

**Task**: 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.

**Example**: given array S = [-1, 0, 1, 2, -1, -4],

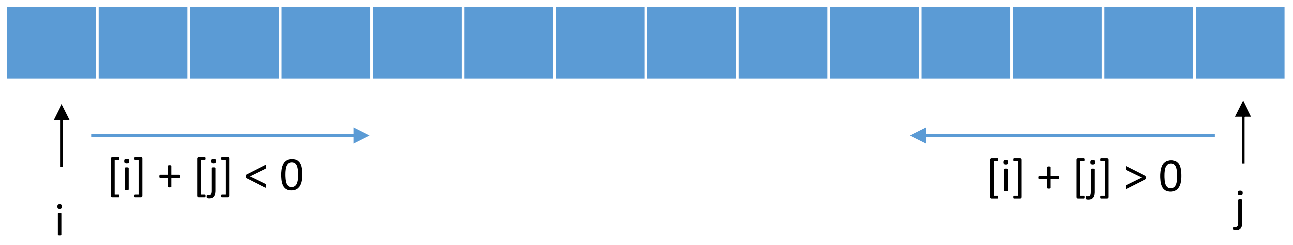
A solution set is:

```
[
  [-1, 0, 1],
  [-1, -1, 2]
]
```

# 3Sum: Reduce to 2Sum

- Time complexity: O(n^2).
- Sort the array and reduce task to 2Sum problem. Move first pointer k from left to right and solve 2Sum problem for other elements: [i] + [j] = -[k].

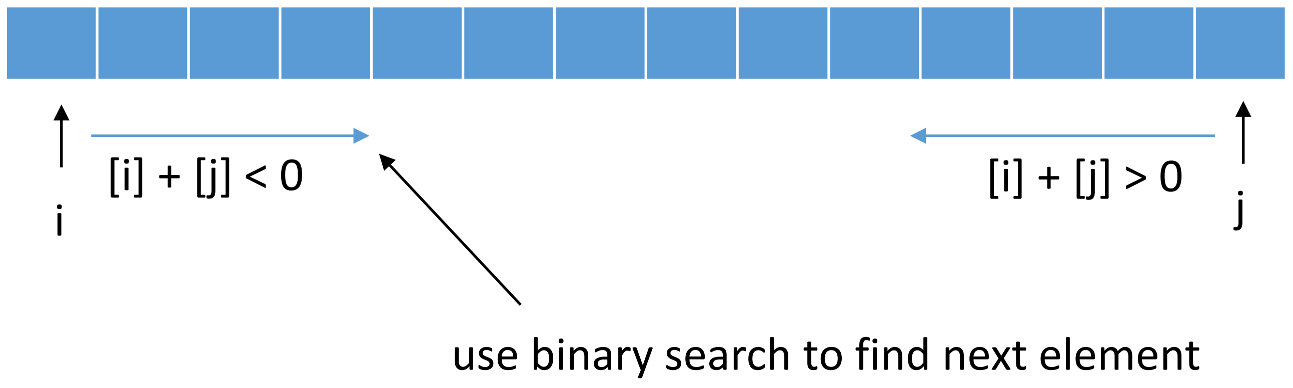
- 2Sum problem with 2 pointers (array already sorted).



# 3Sum: Reduce to 2Sum + binary search

- Time complexity: O(n^2 logn).
- Sort the array and reduce task to 2Sum problem. Move first pointer k from left to right and solve 2Sum problem for other elements: [i] + [j] = -[k].

- 2Sum problem with 2 pointers and binary search (array already sorted).



## 3Sum: hash tables

- Time complexity: O(n^3).
- Use unorderd_map to store elements and their quantity.
- Two loops iterate over elements i and j. The third element -[i] -[j] is found in the unordered_map.

# 3Sum: permutations

- Time complexity: O(n!).

```
vector<vector<int>> threeSum(vector<int>& nums) {
  auto nsize = nums.size();
  if (nsize < 3)
    return vector<vector<int>>();
  vector<vector<int>> output;
  sort(nums.begin(), nums.end());
  do {
    if (nums[nsize - 1] + nums[nsize - 2] + nums[nsize - 3] == 0) {
      vector<int> v = { nums[nsize - 1], nums[nsize - 2], nums[nsize - 3] };
      sort(v.begin(), v.end());
      output.emplace_back(v);
    }
  } while (next_permutation(nums.begin(), nums.end()));
  sort(output.begin(), output.end());
  output.erase(unique(output.begin(), output.end()), output.end());
  return output;
}
```