Solution 1

```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <vector>
#include <unordered_map>
#include <algorithm>
using namespace std;

// Average: O(n^2) time | O(n^2) space
// Worst: O(n^3) time | O(n^2) space
vector<vector<int>> fourNumberSum(vector<int> array, int targetSum) {
  unordered_map<int, vector<vector<int>>> allPairSums;
  vector<vector<int>> quadruplets{};
  for (int i = 1; i < array.size() - 1; i++) {
    for (int j = i + 1; j < array.size(); j++) {
      int currentSum = array[i] + array[j];
      int difference = targetSum - currentSum;
      if (allPairSums.find(difference) != allPairSums.end()) {
        for (vector<int> pair : allPairSums[difference]) {
          pair.push_back(array[i]);
          pair.push_back(array[j]);
          quadruplets.push_back(pair);
        }
      }
    }
    for (int k = 0; k < i; k++) {
      int currentSum = array[i] + array[k];
      if (allPairSums.find(currentSum) == allPairSums.end()) {
        allPairSums[currentSum] = vector<vector<int>>{{array[k], array[i]}};
      } else {
        allPairSums[currentSum].push_back(vector<int>{array[k], array[i]});
      }
    }
  }
  return quadruplets;
}

```