Two Sum III - Data structure design

Design and implement a TwoSum class. It should support the following operations: `add` and `find`.

`add` - Add the number to an internal data structure.
`find` - Find if there exists any pair of numbers which sum is equal to the value.

For example,

```
add(1); add(3); add(5);
find(4) -> true
find(7) -> false
```

Solution 1

I use HashMap to store times of number be added.

When find be called, we iterate the keys of HashMap, then find another number minus by value. Then combine the detections together.

Java:

```java
public class TwoSum {
    private HashMap<Integer, Integer> map = new HashMap<Integer, Integer>();

    public void add(int number) {
        map.put(number, map.containsKey(number) ? map.get(number) + 1 : 1);
    }

    public boolean find(int value) {
        for (Map.Entry<Integer, Integer> entry : map.entrySet()) {
            int i = entry.getKey();
            int j = value - i;
            if ((i == j && entry.getValue() > 1) || (i != j && map.containsKey(j)
)) {
                return true;
            }
        }
        return false;
    }
}
```

C++:

```cpp
class TwoSum {
    unordered_map<int,int> map;
public:
    void add(int number) {
        map[number]++;
    }

    bool find(int value) {
        for (unordered_map<int,int>::iterator it = map.begin(); it != map.end();
it++) {
            int i = it->first;
            int j = value - i;
            if ((i == j && it->second > 1) || (i != j && map.find(j) != map.end()
)) {
                return true;
            }
        }
        return false;
    }
};
```

Python:

```python
class TwoSum:

    # initialize your data structure here
    def __init__(self):
        self.table = dict()

    # @return nothing
    def add(self, number):
        self.table[number] = self.table.get(number, 0) + 1;

    # @param value, an integer
    # @return a Boolean
    def find(self, value):
        for i in self.table.keys():
            j = value - i
            if i == j and self.table.get(i) > 1 or i != j and self.table.get(j, 0
) > 0:
                return True
        return False
```

written by xcv58 original link here

Solution 2

```cpp
class TwoSum {
    unordered_multiset<int> nums;
public:
    void add(int number) {
        nums.insert(number);
    }
    bool find(int value) {
        for (int i : nums) {
            int count = i == value - i ? 1 : 0;
            if (nums.count(value - i) > count) {
                return true;
            }
        }
        return false;
    }
};
```

written by RayZ_O original link here

Solution 3

```
public class TwoSum {
List<Integer> list = new ArrayList<Integer>();
Map<Integer, Integer> map = new HashMap<Integer, Integer>();

// Add the number to an internal data structure.
public void add(int number) {
    list.add(number);
    if (map.containsKey(number))
        map.put(number, map.get(number)+1);
    else
        map.put(number, 1);
}

// Find if there exists any pair of numbers which sum is equal to the value.
public boolean find(int value) {
    for (int i = 0; i < list.size(); i++) {
        int num1 = list.get(i);
        int num2 = value - num1;
        if ((num1 == num2 && map.get(num1) > 1) || (num1 != num2 && map.containsK
ey(num2)))
            return true;
    }
    return false;
}
```

}

written by whdawn original link here