4Sum II

Given four lists A, B, C, D of integer values, compute how many tuples `(i, j, k, l)` there are such that `A[i] + B[j] + C[k] + D[l]` is zero.

To make problem a bit easier, all A, B, C, D have same length of N where 0 ≤ N ≤ 500. All integers are in the range of -2^{28} to $2^{28} - 1$ and the result is guaranteed to be at most $2^{31} - 1$.

Example:

```
Input:
A = [ 1, 2]
B = [-2,-1]
C = [-1, 2]
D = [ 0, 2]

Output:
2

Explanation:
The two tuples are:
1. (0, 0, 0, 1) -> A[0] + B[0] + C[0] + D[1] = 1 + (-2) + (-1) + 2 = 0
2. (1, 1, 0, 0) -> A[1] + B[1] + C[0] + D[0] = 2 + (-1) + (-1) + 0 = 0
```

Solution 1

```
public int fourSumCount(int[] A, int[] B, int[] C, int[] D) {
    Map<Integer, Integer> map = new HashMap<>();

    for(int i=0; i<C.length; i++) {
        for(int j=0; j<D.length; j++) {
            int sum = C[i] + D[j];
            map.put(sum, map.getOrDefault(sum, 0) + 1);
        }
    }

    int res=0;
    for(int i=0; i<A.length; i++) {
        for(int j=0; j<B.length; j++) {
            res += map.getOrDefault(-1 * (A[i]+B[j]), 0);
        }
    }

    return res;
}

Time complexity:  O(n^2)
Space complexity: O(n^2)
```

written by asurana28 original link here

Solution 2

```python
def fourSumCount(self, A, B, C, D):
    AB = collections.Counter(a+b for a in A for b in B)
    return sum(AB[-c-d] for c in C for d in D)
```

written by StefanPochmann original link here

Solution 3

```
public int fourSumCount(int[] A, int[] B, int[] C, int[] D) {
 Map<Integer,Integer> sums = new HashMap<>();
 int count = 0;
 for(int i=0; i<A.length;i++) {
  for(int j=0;j<B.length;j++){
   int sum = A[i]+B[j];
   if(sums.containsKey(sum)) {
    sums.put(sum, sums.get(sum)+1);
   } else {
    sums.put(sum, 1);
   }
  }
 }
 for(int k=0; k<A.length;k++) {
  for(int z=0;z<C.length;z++){
   int sum = -(C[k]+D[z]);
   if(sums.containsKey(sum)) {
    count+=sums.get(sum);
   }
  }
 }
 return count;
}
```

Take the arrays A and B, and compute all the possible sums of two elements. Put the sum in the Hash map, and increase the hash map value if more than 1 pair sums to the same value.

Compute all the possible sums of the arrays C and D. If the hash map contains the opposite value of the current sum, increase the count of four elements sum to 0 by the counter in the map.

written by fabrizio3 original link here