

Experiment – 1.1

Student Name: Avinash Jena

UID: 20BCS2690

Branch: CSE

Section/Group: 707/B

Subject Name: Competitive Coding II

Subject Code: 20CSP-351

1. Aim: 3Sum

2. Objective:

Given an integer array `nums`, return all the triplets `[nums[i], nums[j], nums[k]]` such that $i \neq j$, $i \neq k$, and $j \neq k$, and $nums[i] + nums[j] + nums[k] == 0$.

Notice that the solution set must not contain duplicate triplets.

Example 1:

Input: `nums = [-1,0,1,2,-1,-4]`

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

3. Code:

```
import java.util.ArrayList;
class Solution {
    private List<List<Integer>> res;
    public List<List<Integer>> threeSum(int[] nums) {
        int target = 0;
        return new ArrayList<List<Integer>>() {
            public List<Integer> get(int index) {
                init();
                return res.get(index);
            }
            public int size() {
                init();
                return res.size();
            }
        };
    }
}
```

```
    }  
    private void init() {  
        if (res != null) return;  
        Arrays.sort(nums);  
        int l, r;  
        int sum;  
        Set<List<Integer>> tempRes = new HashSet<>();  
        for (int i = 0; i < nums.length - 2; ++i) {  
            l = i + 1;  
            r = nums.length - 1;  
            while (l < r) {  
                sum = nums[i] + nums[l] + nums[r];  
                if (sum == target) {  
                    List<Integer> t = new ArrayList<>();  
                    t.add(nums[i]);  
                    t.add(nums[l]);  
                    t.add(nums[r]);  
                    tempRes.add(t);  
                }  
                if (sum < target) ++l;  
                else --r;  
            }  
        }  
        res = new ArrayList<List<Integer>>(tempRes);  
    }  
};  
}
```

4. Output:


```
1  import java.util.ArrayList;
2  class Solution {
3      private List<List<Integer>> res;
4      public List<List<Integer>> threeSum(int[] nums) {
5          int target = 0;
6          return new ArrayList<List<Integer>>() {
7              public List<Integer> get(int index) {
8                  init();
9                  return res.get(index);
10             }
11             public int size() {
12                 init();
13                 return res.size();
14             }
15             private void init() {
16                 if (res != null) return;
17                 Arrays.sort(nums);
18                 int l, r;
19                 int sum;
20                 Set<List<Integer>> tempRes = new HashSet<>();
21                 for (int i = 0; i < nums.length - 2; ++i) {
22                     l = i + 1;
23                     r = nums.length - 1;
24                     while (l < r) {
25                         sum = nums[i] + nums[l] + nums[r];
26                         if (sum == target) {
27                             List<Integer> t = new ArrayList<>();
28                             t.add(nums[i]);
29                             t.add(nums[l]);
30                             t.add(nums[r]);
31                             tempRes.add(t);
32                         }
33                         if (sum < target) ++l;
34                         else --r;
35                     }
36                 }
37                 res = new ArrayList<List<Integer>>(tempRes);
38             }
39         };
40     }
41 }
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
14 |         }  
15 |         private void init() {  
16 |             if (res != null) return;  
    ...
```

Testcase **Result** 

Accepted Runtime: 1 ms

• Case 1 • Case 2 • Case 3

Input


nums =
[-1,0,1,2,-1,-4]


Output




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

Expected

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

 [Contribute a testcase](#)

Console 

5. Aim: Merge Two Sorted Lists

6. Objective:

You are given the heads of two sorted linked lists list1 and list2. Merge the two lists in a one sorted list. The list should be made by splicing together the nodes of the first two lists. Return the head of the merged linked list.

Example 1:

Input: list1 = [1,2,4], list2 = [1,3,4]

Output: [1,1,2,3,4,4]

Example 2:

Input: list1 = [], list2 = []

Output: []

7. Code:

```
class Solution {
    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {
        ListNode list3=new ListNode(0);
        ListNode head=list3;

        while(list1!=null && list2!=null){
            if(list1.val<list2.val){
                head.next=list1;
                list1=list1.next;
            }else{
                head.next=list2;
                list2=list2.next;
            }
            head=head.next;
        }
        if(list1!=null){
            head.next=list1;
            list1=list1.next;
        }
        if(list2!=null){
```

```
        head.next=list2;
        list2=list2.next;
    }
    return list3.next;
}
}
```

8. Output:

```
class Solution {
    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {
        ListNode list3=new ListNode(0);
        ListNode head=list3;
        while(list1!=null && list2!=null){
            if(list1.val<list2.val){
                head.next=list1;
                list1=list1.next;
            }else{
                head.next=list2;
                list2=list2.next;
            }
            head=head.next;
        }
        if(list1!=null){
            head.next=list1;
            list1=list1.next;
        }
        if(list2!=null){
            head.next=list2;
            list2=list2.next;
        }
        return list3.next;
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Testcase

Result

Accepted Runtime: 0 ms

• Case 1

• Case 2

• Case 3

Input

list1 =
[1, 2, 4]

list2 =
[1, 3, 4]

Output

[1, 1, 2, 3, 4, 4]

Expected

[1, 1, 2, 3, 4, 4]

Console ▾

Run Submit