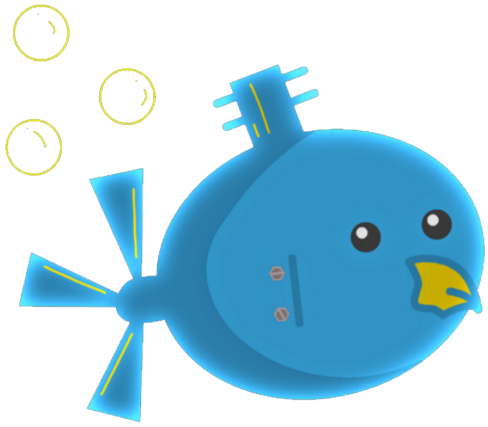


RoboSub MLO1
Interview Questions

Thomas Benson, David Camacho, Bailey Canham,
Brandon Cao, Roberto Hernandez, Andrew Heusser,
Hector Mora-Silva, Bart Rando, Victor Solis

Met as a group for 45 minutes

Subsets

Pseudocode and Code for the LeetCode problem: https://leetcode.com/problems/subsets/

Pseudocode

BIT-MAPPING ALGORITHM

```
Subset(Array A)
        -Create an Array B to hold all subsets found.
        -Create an Array C to hold the current subset.

        n ← Size of Array A
        i ← 0
        While i < 2^n
                j ← 0
                While j < n
                        If the j-th bit of i is set
                                Insert A[j] to C
                Insert C to B
                Clear C
        Return B
```

```
* This algorithm generates all possible subsets of a given array using Bit Mapping.
* Example:
*   An array containing 3 elements -->     [1,2,3]
*
*   All possible representations of a 3 bit number:
*                                   0 0 0
*                                   0 0 1
*                                   0 1 0
*                                   0 1 1
*                                   1 0 0
*                                   1 0 1
*                                   1 1 0
*                                   1 1 1
*
*   Each of those is a bit representation of all possible subsets of the array [1,2,3]
*   On meaning the element is present, off meaning it is not.
*   For any n-length array, all the representations of that n-bit number give all possible subsets.
*   The solution involves looping through the bit representations of an n-bit number, where n is
*   the length of the array, and checking for the set bits. If the 'j^th' bit is set, it's corresponding
*   array value 'A[j]' is added to the subset.
```

Actual Code

[C++] (Thomas)

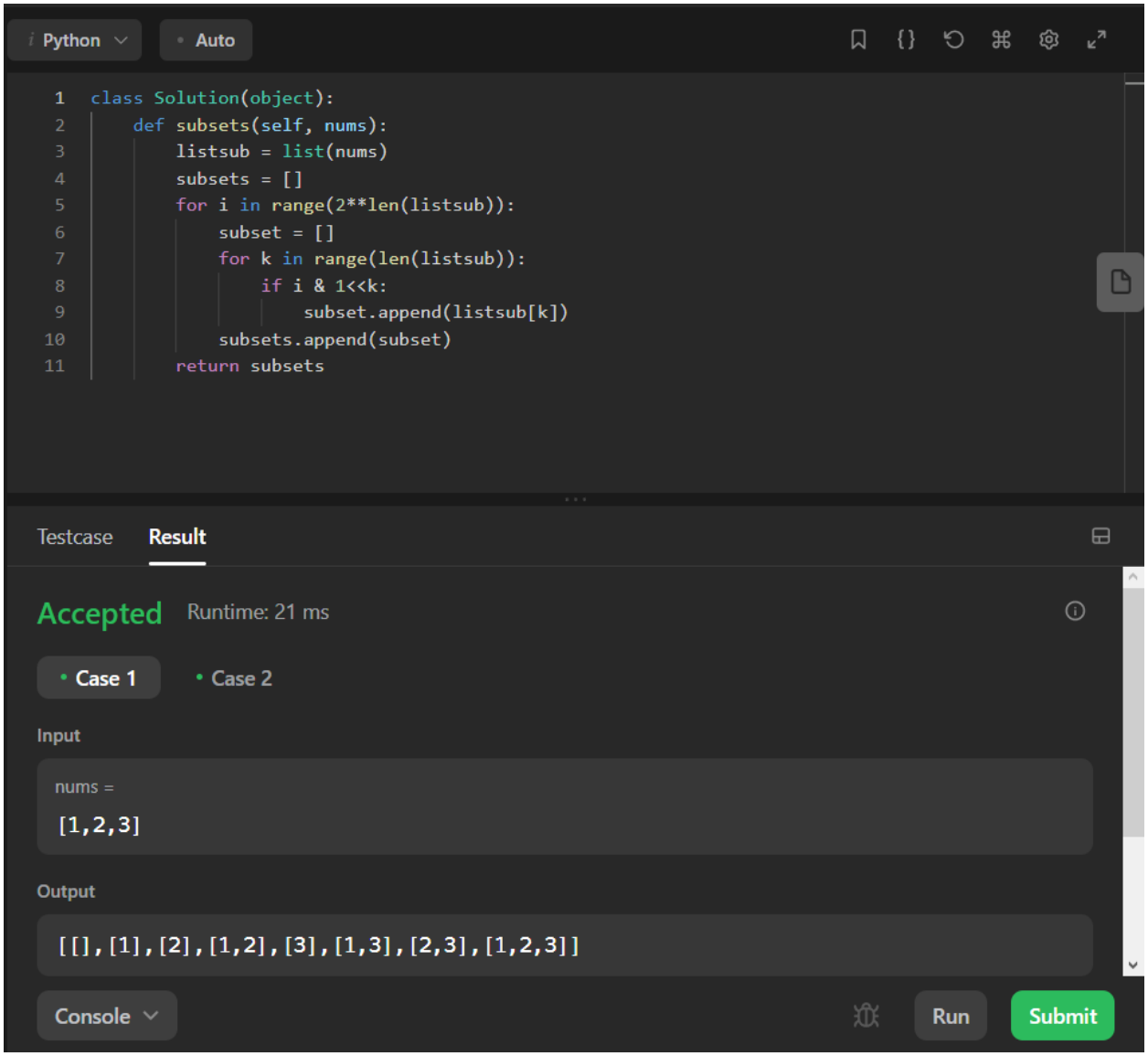
```cpp
class Solution {
public:
    vector<vector<int>> subsets(vector<int>& nums) {
        size_t n = nums.size();
        vector<vector<int>> ans;
        vector<int> temp;
        for (int i=0; i < pow(2, n); ++i) {
            for (int j=0; j < n; ++j) {
                if ((i & (1 << j)) != 0) {
                    temp.push_back(nums[j]);
                }
            }
            ans.push_back(temp);
            temp.clear();
        }
        return ans;
    }
};
```

[Java] (Bailey)

*Solution.java

```java
import java.util.ArrayList;
import java.util.List;

public class Solution {

    public static List<List<Integer>> subsets(int[] nums) {
        List<List<Integer>> finalAns = new ArrayList<> ();
        List<Integer> empty = new ArrayList<> ();
        finalAns.add(empty);

        for (int i = 0; i < nums.length; i++) {
            List<List<Integer>> holdAns = new ArrayList<> ();

            for (int j = 0; j < finalAns.size(); j++) {
                List<Integer> x = new ArrayList<> ();
                x.addAll(finalAns.get(j));
                x.add(nums[i]);
                holdAns.add(x);
            }

            finalAns.addAll(holdAns);
        }

        System.out.println(finalAns);
        return finalAns;
      }
```

[Python] (Brandon)



Leetcode Submission

Submission Detail

10 / 10 test cases passed.

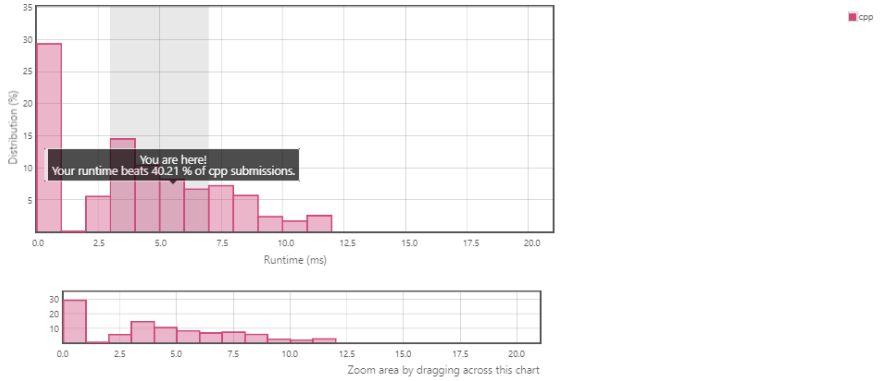
Status: Accepted

Runtime: **5 ms**

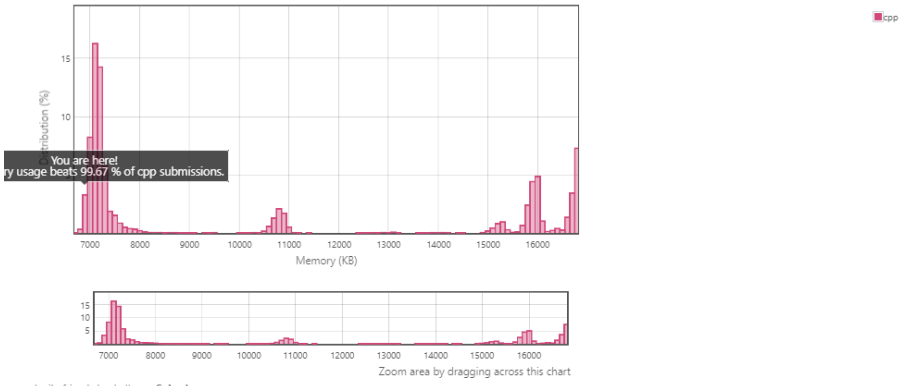
Memory Usage: **6.9 MB**

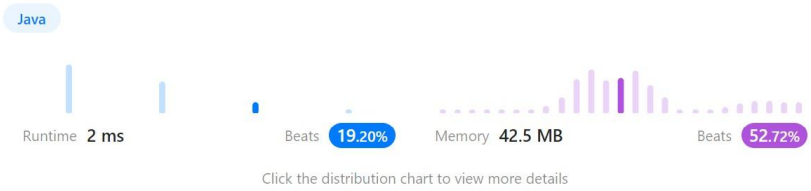
Submitted: **1 hour, 13 minutes ago**

Accepted Solutions Runtime Distribution

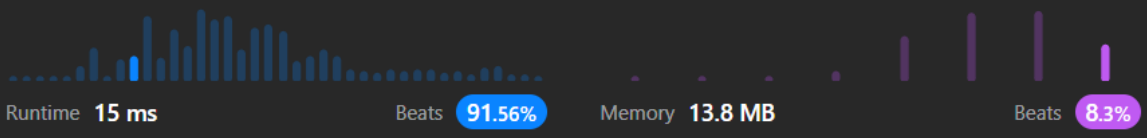


Accepted Solutions Memory Distribution





Python



Runtime **15 ms** Beats **91.56%** Memory **13.8 MB** Beats **8.3%**

Click the distribution chart to view more details

Notes

Write your notes here

Related Tags

Select tags 0/5

```python
class Solution(object):
    def subsets(self, nums):
        listsub = list(nums)
        subsets = []
        for i in range(2**len(listsub)):
            subset = []
            for k in range(len(listsub)):
                if i & 1<<k:
                    subset.append(listsub[k])
```

Remove Duplicates from Sorted List

Pseudocode and Code for the LeetCode problem:
https://leetcode.com/problems/remove-duplicates-from-sorted-list/

Pseudocode

- have a current pointer
- check current and next node exist and if values equal
  - if they do, remove duplicate by changing next node to point to new node
- iterate

(Thomas)
DeleteDuplicate(Head)
Current ← Head
While Current and Current.Next are not Null:
    If Current.Value == Current.Next.Value:
        Temp ← Current.Next.Next
        Delete Current.Next
        Current.Next ← Temp
    Otherwise
        Current ← Current.Next
Return Head

Actual Code

[Java] (David Camacho)

```
public ListNode deleteDuplicates(ListNode head) {
  ListNode current = head;
  while (current != null) {
    while ((current.next != null) && (current.next.val == current.val)) {
      current.next = current.next.next;
    }
    current = current.next;
  }
  return head;
}
```

[C] (Victor)

```c
struct ListNode* deleteDuplicates(struct ListNode* head){
    if (!head) return head;

       struct ListNode* current = head;
       struct ListNode* temp;

       while (current)
       {
            if (current->next && current->next->val == current->val)
           {
                temp = current->next;
                current->next = current->next->next;
                free(temp);
           }
           else
           {
                current = current->next;
            }
        }
        return head;
}
```

[C++] (Thomas)

```cpp
//  Definition for singly-linked list.
struct ListNode {
    int val;
    ListNode *next;
    ListNode() : val(0), next(nullptr) {}
    ListNode(int x) : val(x), next(nullptr) {}
    ListNode(int x, ListNode *next) : val(x), next(next) {}
 };

class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        ListNode* current = head;
        while (current && current->next) {
            if (current->val == current->next->val) {
                ListNode* tmp = current->next->next;
                delete current->next;
                current->next = tmp;
            } else {
                current = current->next;
            }
        }
        return head;
    }
};
```

Leetcode Submission

Remove Duplicates from Sorted List

Submission Detail

166 / 166 test cases passed.

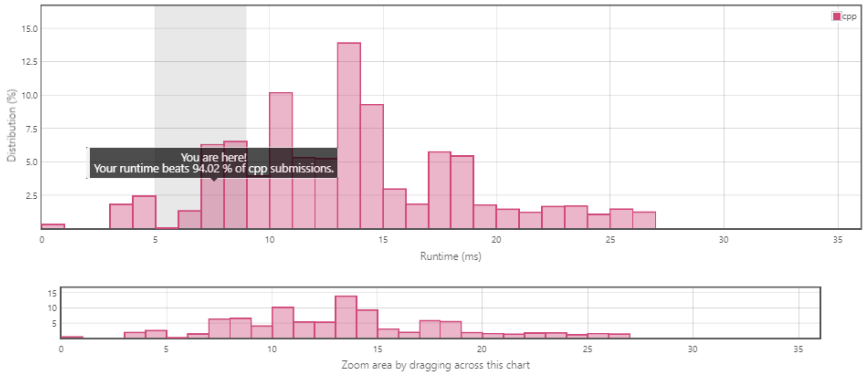
Status: **Accepted**

Runtime: **7 ms**

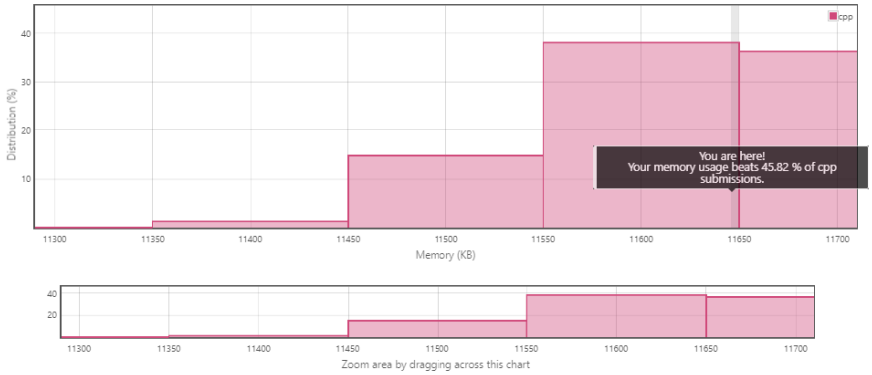
Memory Usage: **11.6 MB**

Submitted: **0 minutes ago**

Accepted Solutions Runtime Distribution



Accepted Solutions Memory Distribution



Invite friends to challenge **Remove Duplicates from Sorted List**

Submitted Code: 0 minutes ago

Language: cpp

Edit Code

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        ListNode* current = head;
        while (current && current->next) {
            if (current->val == current->next->val) {
                ListNode* tmp = current->next->next;
                delete current->next;
                current->next = tmp;
            } else {
                current = current->next;
            }
        }
        return head;
    }
};
```

Back to problem