

Description Accepted Editorial Solutions Submissions

All Submissions

Accepted 114 / 114 testcases passed
amit_8933 submitted at Feb 10, 2026 17:01

Runtime 0 ms | Beats 100.00%
Memory 43.19 MB | Beats 93.71%

Analyze Complexity

100%
50%
0%
1ms 2ms 3ms 4ms

Code | Java

```
1 class Solution {
2     public int[] plusOne(int[] digits) {
3         // Start from last digit
4         for (int i = digits.length - 1; i >= 0; i--) {
```

Code | Java

```
14     }
15
16     // If all digits were 9
17     int[] result = new int[digits.length + 1];
18     result[0] = 1;
19
20     return result;
21 }
22 }
```

Java Auto

Saved

Ln 23, Col 1

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

digits =
[1,2,3]

Output

[1,2,4]

Expected

Description Accepted Editorial Solutions Submissions

All Submissions

Accepted 128 / 128 testcases passed
amit_8933 submitted at Feb 10, 2026 16:51

Runtime 7 ms Beats 76.55% Memory 48.26 MB Beats 99.94%

Analyze Complexity

0% 20% 40%

Code Java

```
1 import java.util.*;
2
3 class Solution {
4     public List<List<String>> groupAnagrams(String[] strs) {
5         Map<String, List<String>> map = new HashMap<String, List<String>>();
6
7         for (String s : strs) {
8             char[] arr = s.toCharArray();
9             Arrays.sort(arr);
10            String key = new String(arr);
11
12            if (!map.containsKey(key)) {
13                map.put(key, new ArrayList<String>());
14            }
15            map.get(key).add(s);
16        }
17
18        // Step 3: Return all grouped values
19        return new ArrayList<List<String>>(map.values());
20    }
21
22 }
23
24
25 }
26
```

Java Auto

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```
strs = ["eat","tea","tan","ate","nat","bat"]
```

Output

```
[["eat","tea","ate"],["bat"],["tan","nat"]]
```

Expected

← All Submissions

Accepted 110 / 110 testcases passed

amit_8933 submitted at Feb 10, 2026 16:48

Editorial Solution

Runtime 1 ms | Beats 99.53% Memory 47.61 MB Beats 6.56%

Analyze Complexity

100% 50% 0%

17ms 33ms 50ms 66ms 83ms 99ms 116ms

Code | Java

```
1 class Solution {  
2     public int jump(int[] nums) {  
3         int jumps = 0;  
4         int currentEnd = 0;  
5         ...  
6     }  
7 }
```

Code

Java v Auto

```
1 class Solution {  
2     public int jump(int[] nums) {  
3         int jumps = 0;  
4         int currentEnd = 0;  
5         int farthest = 0;  
6  
7         for (int i = 0; i < nums.length - 1; i++) {  
8             farthest = Math.max(farthest, i + nums[i]);  
9         }  
10    }  
11 }
```

Saved

Ln 22, Col 1

Testcase Test Result

You must run your code first

← All Submissions

Accepted 160 / 160 testcases passed
amit_8933 submitted at Feb 10, 2026 16:43

Runtime 1 ms | Beats 99.78% | Analyze Complexity

Memory 46.00 MB | Beats 19.64%

Code | Java

```
1 class Solution {  
2     public List<List<Integer>> combinationSum(int[] candidates, int target)  
3     {  
4         List<List<Integer>> ans = new ArrayList<>();  
5         List<Integer> path = new ArrayList<>();  
6         soln(candidates, target, 0, path, ans);  
7     }  
8     void soln(int[] arr, int target, int i, List<Integer> path, List<List<Integer>> ans)  
9     {  
10        if (target < 0) return;  
11        if (target == 0) ans.add(new ArrayList<Integer>(path));  
12        for (int j = i; j < arr.length; j++)  
13        {  
14            path.add(arr[j]);  
15            soln(arr, target - arr[j], j, path, ans);  
16            path.remove(path.size() - 1);  
17        }  
18    }  
19}
```

Code | Java

Editorial Solution

Code | Java

Runtime 1 ms | Beats 99.78% | Analyze Complexity

Memory 46.00 MB | Beats 19.64%

Testcase > Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

candidates = [2,3,6,7]

target = 7

Output

Array < > ⌂

Description | Editorial | Solutions | Accepted X | Submissions

← All Submissions

Accepted 66 / 66 testcases passed
amit_8933 submitted at Feb 10, 2026 16:27

Runtime 0 ms | Beats 100.00% | Analyze Complexity

Memory 45.02 MB | Beats 9.96%

150%

100%

50%

0%

1ms 2ms 3ms 4ms

Code | Java

```
1 class Solution {  
2     public int searchInsert(int[] nums, int target) {  
3         int start = 0;  
4         int end = nums.length - 1;  
5         int mid = start + (end - start) / 2;  
6         if (nums[mid] == target) {  
7             return mid;  
8         } else if (target < nums[mid]) {  
9             searchInsert(nums, target);  
10        } else {  
11            searchInsert(nums, target);  
12        }  
13    }  
14}
```

Submit

Java v Auto

1 class Solution {
2 public int searchInsert(int[] nums, int target) {
3 int start = 0;
4 int end = nums.length - 1;
5 int mid = start + (end - start) / 2;
6 if (nums[mid] == target) {
7 return mid;
8 } else if (target < nums[mid]) {
9 searchInsert(nums, target);
10 } else {
11 searchInsert(nums, target);
12 }
13 }
14}

Saved Ln 24, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums = [1,3,5,6]

target = 5

Output

Description | Editorial | Solutions | Accepted X | Submissions

All Submissions

Accepted 176 / 176 testcases passed
amit_89333 submitted at Feb 10, 2026 16:45

Runtime: 5 ms | Beats 99.22% | Memory: 44.84 MB | Beats 96.75%

Analyze Complexity

Runtime distribution chart showing execution times from 1ms to 7ms.

Code | Java

```

1 class Solution {
2     public List<List<Integer>> combinationSum2(int[] candidates, int target)
3         Arrays.sort(candidates);
4         List<List<Integer>> ans = new ArrayList<>();
5         List<Integer> path = new ArrayList<>();
6         soln(arr, target, arr[0], 0, 0, ans);
7         return ans;
8     }
9     void soln(int[] arr, int target, int sum, int index, List<List<Integer>> ans, List<Integer> path) {
10        if (sum > target) return;
11        if (sum == target) {
12            ans.add(new ArrayList<Integer>(path));
13            return;
14        }
15        for (int i = index; i < arr.length; i++) {
16            if (j > i && arr[j] == arr[j-1]) continue;
17            if (arr[j] > target) break;
18            path.add(arr[j]);
19            soln(arr, target - arr[j], j+1, path, ans);
20            path.remove(path.size() - 1);
21        }
22    }
23 }
24 }
```

Code | Java

Testcase | Test Result

Accepted Runtime: 1 ms

Case 1 Case 2

Input

```
candidates =
[10, 1, 2, 7, 6, 1, 5]
```

target =
8

Output

Description | Accepted | Editorial | Solutions | Submissions

All Submissions

Accepted 114 / 114 testcases passed

amit_8933 submitted at Feb 10, 2026 17:01

Editorial

Solution

Runtime

0 ms | Beats 100.00%

Analyze Complexity

Memory

43.19 MB | Beats 93.71%



Code | Java

```
1 class Solution {
2     public int[] plusOne(int[] digits) {
3
4         // Start from last digit
5         for (int i = digits.length - 1; i >= 0; i--) {
```

</> Code

Java | Auto

```
14 }
15
16 // If all digits were 9
17 int[] result = new int[digits.length + 1];
18 result[0] = 1;
19
20 }
21
22 }
23 }
```

Saved

Ln 23, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

digits =

[1,2,3]

Output

[1,2,4]

Expected

← All Submissions

Accepted 110 / 110 testcases passed
amit_8933 submitted at Feb 10, 2026 16:48

Runtime 1 ms | Beats 99.53%
Memory 47.61 MB | Beats 6.56%

Analyze Complexity

Runtime distribution: 17ms, 33ms, 50ms, 66ms, 83ms, 99ms, 116ms

Code | Java

```
</> Code
Java v Auto
1 class Solution {
2     public int jump(int[] nums) {
3
4         int jumps = 0;
5         int currentEnd = 0;
6         int farthest = 0;
7
8         for (int i = 0; i < nums.length - 1; i++) {
9
10            farthest = Math.max(farthest, i + nums[i]);
11
}
Saved
Ln 22, Col 1
Testcase Test Result
You must run your code first
```

Array < > ⌂

Description | Editorial | Solutions | Accepted X | Submissions

← All Submissions

Accepted 66 / 66 testcases passed
amit_8933 submitted at Feb 10, 2026 16:27

Runtime 0 ms | Beats 100.00% | Analyze Complexity

Memory 45.02 MB | Beats 9.96%

150%

100%

50%

0%

1ms 2ms 3ms 4ms

Code | Java

```
1 class Solution {  
2     public int searchInsert(int[] nums, int target) {  
3         int start = 0;  
4         int end = nums.length - 1;  
5         int mid = start + (end - start) / 2;  
6         if (nums[mid] == target) {  
7             return mid;  
8         } else if (target < nums[mid]) {  
9             searchInsert(nums, target);  
10        } else {  
11            searchInsert(nums, target);  
12        }  
13    }  
14}
```

Submit

Java v Auto

1 class Solution {
2 public int searchInsert(int[] nums, int target) {
3 int start = 0;
4 int end = nums.length - 1;
5 int mid = start + (end - start) / 2;
6 if (nums[mid] == target) {
7 return mid;
8 } else if (target < nums[mid]) {
9 searchInsert(nums, target);
10 } else {
11 searchInsert(nums, target);
12 }
13 }
14}

Saved Ln 24, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums = [1,3,5,6]

target = 5

Output

Description | Editorial | Solutions | Accepted X | Submissions

All Submissions

Accepted 176 / 176 testcases passed
amit_89333 submitted at Feb 10, 2026 16:45

Runtime 5 ms | Beats 99.22%
Memory 44.84 MB | Beats 96.75%

Analyze Complexity

Code | Java

```

1 class Solution {
2     public List<List<Integer>> combinationSum2(int[] candidates, int target)
3         Arrays.sort(candidates);
4         List<List<Integer>> ans = new ArrayList<>();
5         List<Integer> path = new ArrayList<>();
6         soln(arr, target, arr[0], 0, 0, ans);
7         return ans;
8     }
9     void soln(int[] arr, int target, int sum, int index, List<List<Integer>> ans, List<Integer> path) {
10        if (sum > target) return;
11        if (sum == target) {
12            ans.add(new ArrayList<Integer>(path));
13            return;
14        }
15        for (int i = index; i < arr.length; i++) {
16            if (j > i && arr[j] == arr[j-1]) continue;
17            if (arr[j] > target) break;
18            path.add(arr[j]);
19            soln(arr, target - arr[j], j+1, path, ans);
20            path.remove(path.size() - 1);
21        }
22    }
23 }
24 }
```

Java | Auto

Testcase | Test Result

Accepted Runtime: 1 ms

Case 1 Case 2

Input

```
candidates =
[10, 1, 2, 7, 6, 1, 5]
```

target =
8

Output

Description | Accepted X | Editorial | Solutions | Submissions

All Submissions

Accepted 128 / 128 testcases passed

amit_8933 submitted at Feb 10, 2026 16:51

Runtime 7 ms | Beats 76.55% Memory 48.26 MB | Beats 99.94%

Analyze Complexity

Code | Java

```
1 import java.util.*;
2
3 class Solution {
4     public List<List<String>> groupAnagrams(String[] strs) {
5         ...
```

Code

Java v Auto

```
17     map.put(key, new ArrayList<>());
18 }
19 }
20 }
21
22 // Step 3: Return all grouped values
23 return new ArrayList<>(map.values());
24
25 }
```

Saved

Ln 26, C

Testcase > Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```
strs = ["eat","tea","tan","ate","nat","bat"]
```

Output

```
[["eat","tea","ate"],["bat"],["tan","nat"]]
```

Expected

← All Submissions

Accepted 160 / 160 testcases passed
amit_8933 submitted at Feb 10, 2026 16:43

Runtime 1 ms | Beats 99.78% | Analyze Complexity

Memory 46.00 MB | Beats 19.64%

Code | Java

```
1 class Solution {  
2     public List<List<Integer>> combinationSum(int[] candidates, int target)  
3     {  
4         List<List<Integer>> ans = new ArrayList<>();  
5         List<Integer> path = new ArrayList<>();  
6         soln(candidates, target, 0, path, ans);  
7     }  
8     void soln(int[] arr, int target, int i, List<Integer> path, List<List<Integer>> ans)  
9     {  
10        if (target < 0) return;  
11        if (target == 0) ans.add(new ArrayList<Integer>(path));  
12        for (int j = i; j < arr.length; j++)  
13        {  
14            path.add(arr[j]);  
15            soln(arr, target - arr[j], j, path, ans);  
16            path.remove(path.size() - 1);  
17        }  
18    }  
19}
```

Code

Java | Auto

```
17  
18  
19  
20  
21  
22  
23 }
```

Saved

Testcase > Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```
candidates =  
[2,3,6,7]
```

target =
7

Output

Code

Java Auto

```

1 class Solution {
2     public int[] searchRange(int[] nums, int target) {
3         int[] ans = {-1, -1};
4
5         // First occurrence
6         ans[0] = binarySearch(nums, target, true);
7         // Last occurrence
8         ans[1] = binarySearch(nums, target, false);
9
10        return ans;
11    }
12
13    // Helper method to find first or last position

```

Saved

Ln 1, Col 1

Description | **Editorial** | **Solutions** | **Submissions**

Status	Language	Runtime	Memory	Notes
Accepted a few seconds ago	Java	0 ms	48.5 MB	
Accepted Feb 13, 2026	Java	0 ms	48.1 MB	

Testcase | **Test Result** | **Accepted** | **X**

All Submissions

Accepted 88 / 88 testcases passed
amit_8933 submitted at Feb 15, 2026 23:38

Runtime | **Memory**

0 ms | Beats 100.00% | 48.54 MB | Beats 11.72%

Analyze Complexity

Code Java

```

1 class Solution {
2     public int[] searchRange(int[] nums, int target) {
3         int[] ans = {-1, -1};
4
5         // First occurrence
6         ans[0] = binarySearch(nums, target, true);
7         // Last occurrence
8         ans[1] = binarySearch(nums, target, false);
9
10        return ans;
11    }
12
13    // Helper method to find first or last position

```

Java Auto

```
1 class Solution {
2     public int search(int[] nums, int target) {
3         int start = 0, end = nums.length - 1;
4
5         while (start <= end) {
6             int mid = start + (end - start) / 2;
7
8             if (nums[mid] == target) return mid;
9
10            // Check if left part is sorted
11            if (nums[mid] >= nums[start]) {
12                if (target >= nums[start] && target < nums[mid]) {
13                    end = mid - 1; // move left
14                }
15            }
16        }
17    }
18}
```

Ln 1, Col 1

All Submissions

Accepted 196 / 196 testcases passed
amit_89333 submitted at Feb 13, 2026 21:17

Editorial Solution

Runtime Memory

0 ms Beats 100.00% 43.97 MB Beats 44.98%

Analyze Complexity

0.3% of solutions used 2 ms of runtime

Code Java

```
1 class Solution {
2     public int search(int[] nums, int target) {
3         int start = 0, end = nums.length - 1;
4
5         while (start <= end) {
6             int mid = start + (end - start) / 2;
7
8             if (nums[mid] == target) return mid;
9
10            // Check if left part is sorted
11            if (nums[mid] >= nums[start]) {
12                if (target >= nums[start] && target < nums[mid]) {
13                    end = mid - 1; // move left
14                }
15            }
16        }
17    }
18}
```