beta Firecode.io™

Paul's Solutions Book

1 2 3 4 5

Delete the Node at a Particular Position in a Linked List

Linked Lists

Given a singly-linked list, implement a method to delete the node at a given position (starting from 1 as the head position) and return the head of the list. Do nothing if the input position is out of range.

```
Examples:

LinkedList: 1->2->3->4 , Head = 1

deleteAtMiddle(Head,3) ==> 1->2->4
```

Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public ListNode deleteAtMiddle(ListNode head, int position) {

    //if (head == null)
    //    return head;

    //ListNode ptr = head;
    //int count = 1;

    if(position == 1){
        return head == null ? null : (head = head.next);
    }

    //whileLoop:
    //while(ptr != null && ptr.next != null){
    //    if(count == (position - 1)){
    //        ptr.next = ptr.next.next;
    //        return head;
    //    }
    //    else if (count > (position)){
    //        break;
    //    }
    //    ptr = ptr.next;
    //    count ++;
```

```
    //}
    //return head


    ListNode curr = head;
    ListNode prev = curr;
    int count = 0;
    while(curr != null && count < position){
        count++;
        if(count == position){
            prev.next = curr.next;
            return head;
        }
        else {
            prev = curr;
            curr = curr.next;
        }
    }
    return head;
}
```

Top voted solution

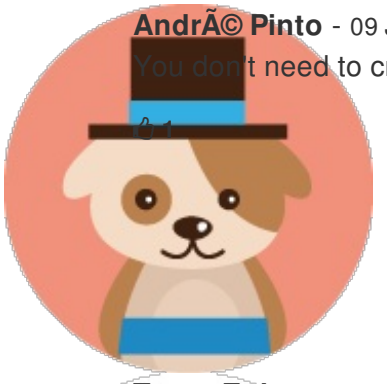
```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public ListNode deleteAtMiddle(ListNode head, int position) {
    if (head == null) { return null; }
    if (position == 1) { return head.next; }
    ListNode result = new ListNode(head.data);
    result.next = deleteAtMiddle(head.next, position - 1);
    return result;

}
```

Comments

Darren - 21 Jun, 2016

Really well done

0



AndrÃ© Pinto - 09 Jul, 2016

You don't need to create a new ListNode every time. Changing .next is enough.

TangoZulu - 15 Jun, 2017

Neat approach, but a no-hire. Your recursion is O(n) memory as well as O(n) time due to the stack. And definitely don't need to allocate a new node every time.

Horizontal Flip

Multi Dimensional Arrays

You are given an m x n 2D image matrix where each integer represents a pixel. Flip it **in-place** along its horizontal axis.

Example:

Input image :

1 1
0 0

Modified to :

0 0
1 1

Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static void flipHorizontalAxis(int[][] matrix) {

    //for(int y = 0; y < matrix.length / 2; y++){
    //    int[] temp = matrix[y];
    //    matrix[y] = matrix[(matrix.length - 1) - y];
    //    matrix[(matrix.length - 1) - y] = temp;
    //}

    for (int y = 0; y < matrix.length / 2; y++){
        for (int x = 0; x < matrix[y].length; x++){
            int temp = matrix[y][x];
            matrix[y][x] = matrix[matrix.length - 1 - y][x];
            matrix[matrix.length - 1 - y][x] = temp;
        }
    }

}
```

Top voted solution

```
// java.util.* has been imported for this problem.
```

```
// You don't need any other imports.

public static void flipHorizontalAxis(int[][] matrix) {
    if(matrix.length > 1) {
        for(int i = 0; i < matrix.length / 2; i++) {
            int[] temp = matrix[i];
            matrix[i] = matrix[matrix.length - 1 - i];
            matrix[matrix.length - 1 - i] = temp;
        }
    }
}
```

Comments

Preetam Bhosle - 26 Apr, 2017

nice solution.

0

Victor Cui - 13 May, 2017

Doesn't the problem say to swap in place?

0

1 2 3 4 5

Binary Search on Array of Integers

Strings Arrays

Write a method that searches an Array of integers for a given integer using the Binary Search Algorithm. If the input integer is found in the array, return `true`. Otherwise, return `false`. You can assume that the given array of integers is already sorted in ascending order.
Examples:

```
binarySearch({2,5,7,8,9},9) -> true
```

```
binarySearch({2,8,9,12},6) -> false
```

```
binarySearch({2},4) -> false
```

```
binarySearch({},9) -> false
```

```
   {} -> [Empty] Array
```

Your Notes

```
Don't do
mid = (end + start) / 2
Resutls in overflow

So either unsigned bit shift right >>> 1
OR
start + ((end - start)/2)
```

Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static Boolean binarySearch(int[] arr, int n){

    if(arr.length == 0)
        return false;
    int start = 0;
    int end = arr.length - 1;
```

```
    while(start <= end){
        int mid = ((end + start) >>> 1);

        if (arr[mid] == n){
            return true;
        }
        if(arr[mid] < n){
            start = mid + 1;
        }
        if (arr[mid] > n){
            end = mid - 1;
        }
    }
    return false;

}
```

Top voted solution

```
public static Boolean binarySearch(int[] arr, int n){
        // Think about maintaining the boundaries of your array with a lo
        // index and a hi index. lo=0; hi=length - 1;
        int lo = 0;
        int hi = arr.length - 1;
        // Use a while loop to iterate until lo <= hi. The moment that lo exceeds
        // hi, we can be sure that the entire array has been searched.
        while(lo <= hi){
            // Create a mid index. int mid = lo + (hi-lo)/2;
            int mid = lo + (hi-lo)/2;
            // If arr[mid] < n, set lo = mid+1 to divide the array
            if(arr[mid] < n) lo = mid+1;
            // Else If arr[mid] > n, set hi = mid - 1 to divide the array
            else if(arr[mid] > n) hi = mid-1;
            // Otherwise, return true!
            else return true;
        }
        return false;
    }
```

1 2 3 4 5

Find the Missing Number in a Set of Numbers from 1 to 10

Strings

Given an Array containing 9 numbers ranging from 1 to 10, write a method to find the missing number. Assume you have 9 numbers between 1 to 10 and only one number is missing.

```
findMissingNumber({1,2,4,5,6,7,8,9,10}) --> 3
```

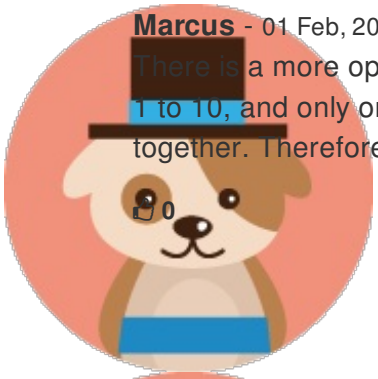
Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static int findMissingNumber(int[] arr) {
    //SortedSet<Integer> set = new TreeSet<Integer>();
    //for(int i = 0; i < arr.length; i++){
    //    set.add(arr[i]);
    //}

    //for(int i = 1; i <= 10; i++){
    //    if(!set.contains(i)){
    //        return i;
    //    }
    //}

    int sum = 55;
    for(int i = 0; i < arr.length; i ++){
        sum -= arr[i];
    }

    return sum;



}
```

Top voted solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static int findMissingNumber(int[] arr) {
    // Add your code below this line. Do not modify any other code.
 int sumInt = 0;
 int total = 55;
for(int i= 0; i <= arr.length-1; i++){
    sumInt += arr[i];
}
 return total-sumInt;



    // Add your code above this line. Do not modify any other code.
}
```

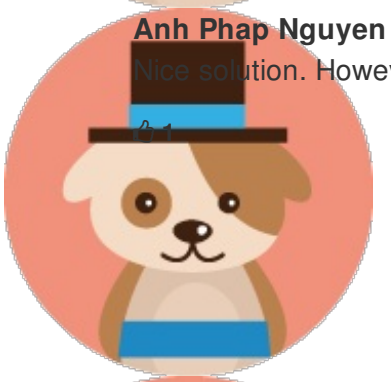
Comments



Marcus - 01 Feb, 2017

There is a more optimized solution: There are two known facts: There are nine numbers, the range is from 1 to 10, and only one is missing. You could have just initialized the sum by adding all of the 9 numbers together. Therefore, the run-time of the program would just be in constant time.

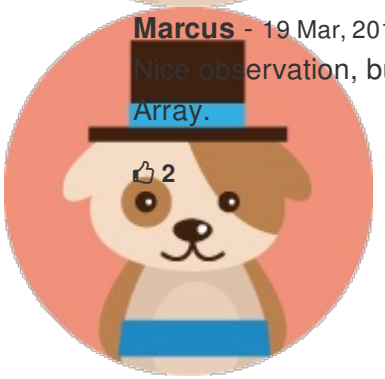
0



Anh Phap Nguyen - 19 Mar, 2017

Nice solution. However it only works when numbers are unique.

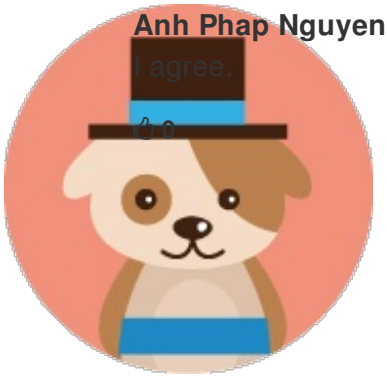
1



Marcus - 19 Mar, 2017

Nice observation, but, considering the scope of the question, there will always be unique numbers in the Array.

2



Anh Phap Nguyen - 19 Mar, 2017

I agree.

0

Fibonacci Number

Recursion Numbers

The Fibonacci Sequence is the series of numbers: `0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...` The next number is found by adding up the two numbers before it.

Write a **recursive** method `fib(n)` that returns the `nth` Fibonacci number. `n` is 0 indexed, which means that in the sequence `0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...`, n == 0 should return 0 and n == 3 should return 2. Assume n is less than 15.

Even though this problem asks you to use recursion, more efficient ways to solve it include using an Array, or better still using 3 volatile variables to keep a track of all required values. Check out this blog post to examine better solutions for this problem.

Examples:

`fib(0) ==> 0`

`fib(1) ==> 1`

`fib(3) ==> 2`

Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static int fib(int n) {
    //if(n == 0)
    //    return 0;
    //else if(n == 1)
    //    return 1;
    //return fib(n-1) + fib(n-2);
    //List<Integer> fibList = new ArrayList<Integer>();
    //fibList.add(1);
    //fibList.add(1);

    //int count = fibList.size() - 1;
    //while(fibList.size() < n){
    //    fibList.add(fibList.get(count) + fibList.get(count - 1));
    //    count++;
    //}
    //return fibList.get(n - 1);
    //if(n == 0)
    //    return 0;
```

```
    //else if (n <= 2)
    //    return 1;

    //int[] fibArray = new int[n];
    //fibArray[0] = 1;
    //fibArray[1] = 1;
    //for(int i = 2; i < n; ++i){
    //    fibArray[i] = fibArray[i-1] + fibArray[i-2];
    //}
    //return fibArray[n-1];

    // O(1) memory O(n) speed
    if (n == 0)
        return 0;
    else if (n <= 2)
        return 1;
    int temp = 0, x = 1, y = 1;

    for(int i = 2; i < n; i++){
        temp = x + y;
        x = y;
        y = temp;
    }
    return temp;

}
```

Top voted solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static int fib(int n) {
    return (n<2) ? n : fib(n-1)+fib(n-2);
}
```

Comments

Sandesh - 30 Jul, 2016

This is a time exponential time complexity. A better approach would be to use dynamic programming.

3

1 2 3 4 5

Repeated Elements in an Array

Arrays

Write a method `duplicate` to find the repeated or duplicate elements in an array.
This method should return a list of repeated integers in a **string** with the elements sorted in **ascending order** (as illustrated below).

```
duplicate({1,3,4,2,1}) --> "[1]"
```

```
duplicate({1,3,4,2,1,2,4}) --> "[1, 2, 4]"
```

Note: You may use `toString()` method to return the **standard string representation** of most data structures, and `Arrays.sort()` to sort your result.

Your solution

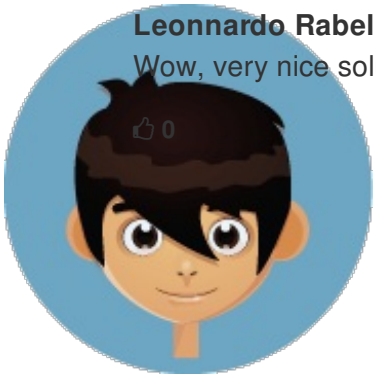
```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static String duplicate(int[] numbers){
    // Map<Integer, Boolean> dupList = new HashMap<Integer, Boolean>();

    // for(int n: numbers){
    //     if(dupList.containsKey(n)){
    //         dupList.put(n, true);
    //     } else {
    //         dupList.put(n, false);
    //     }
    //}

    //SortedSet<Integer> list = new TreeSet<Integer>();
    //for(Map.Entry<Integer, Boolean> m: dupList.entrySet()){
    //     if(m.getValue() == true){
    //         list.add(m.getKey());
    //     }
    //}

    //return list.toString();

    SortedSet<Integer> set = new TreeSet<Integer>();
    Set<Integer> tempSet = new HashSet<Integer>();
    for(int i = 0; i < numbers.length; i ++){
```

```
        if(tempSet.contains(numbers[i]))
            set.add(numbers[i]);
        else
            tempSet.add(numbers[i]);
    }

    return set.toString();

}
```

Top voted solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static String duplicate(int[] numbers){
    // Add your code below this line. Do not modify any other code.
    Set<Integer> map = new HashSet<>();
    Set<Integer> list = new TreeSet<>();
    for (int num : numbers) {
        if (map.contains(num)) list.add(num);
        else map.add(num);
    }

    return list.toString();




    // Add your code above this line. Do not modify any other code.
}
```

Comments



Leonnardo Rabello - 15 Aug, 2016

Wow, very nice solution :)

1 2 3 4 5

Find the Number that Appears Once

Arrays Hash-Tables Numbers Miscellaneous

Write a method that returns a number that appears only once in an array.
Assume the array will surely have a unique value. The array will never be empty.
Examples:

```
{1,2,3,4,1,2,4,3,5} ==> 5
```

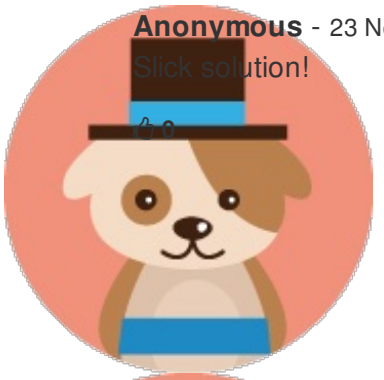
Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static int singleNumber(int[] A) {


    Map<Integer, Boolean> dict = new HashMap<Integer, Boolean>();

    for(int i = 0; i < A.length; i++){
        int c = A[i];
        if(dict.containsKey(c)){
            dict.put(c, true);
        } else {
            dict.put(c, false);
        }
    }

    for(Map.Entry<Integer, Boolean> d: dict.entrySet()){
        if(d.getValue() == false){
            return d.getKey();
        }
    }
    return 0;

}
```

Top voted solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static int singleNumber(int[] A) {
    // Add your code below this line. Do not modify any other code.
    Hashtable<Integer, Integer> nums = new Hashtable<Integer, Integer>();
    int result = -1;

    for (int i = 0; i < A.length; i++) {
        Integer val = nums.get(A[i]);
        if (val == null)
            nums.put(A[i], 1);
        else
            nums.put(A[i], val + 1);
    }

    Enumeration<Integer> keys = nums.keys();
    while (keys.hasMoreElements()) {
        int k = keys.nextElement();
        if (nums.get(k) == 1) result = k;
    }

    return result;
    // Add your code above this line. Do not modify any other code.
}
```

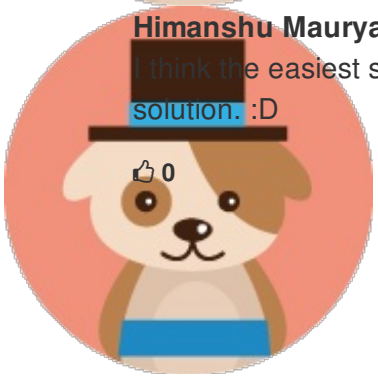
Comments



Anonymous - 23 Nov, 2015

Slick solution!

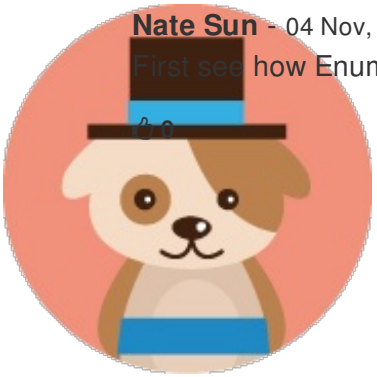
0



Himanshu Maurya - 20 Sep, 2016

I think the easiest solution is to xor all the elements, the result will be the anwser. Look Below for my solution. :D

0



Nate Sun - 04 Nov, 2016

First see how Enumeration used to get the duplicate. HashSet or HashMap is better?

0

Palindrome Tester

Strings Arrays

A palindrome is a string or sequence of characters that reads the same backward as forward. For example, "madam" is a palindrome. Write a method that takes in a `String` and returns a `boolean` -> `true` if the input `String` is a palindrome and `false` if it is not. An empty string and a null input are considered palindromes. You also need to account for the space character. For example, "race car" should return `false` as read backward it is "rac ecar".

Examples:

```
isStringPalindrome("madam") -> true
isStringPalindrome("aabb") -> false
isStringPalindrome("race car") -> false
isStringPalindrome("") -> true
isStringPalindrome(null) -> true
```

Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static boolean isStringPalindrome(String str){

    if(str == null)
        return true;

    for(int i = 0; i < str.length() / 2; i++){
        if(str.charAt(i) != str.charAt(str.length() - 1 - i)){
            return false;
        }
    }
    return true;


}
```

Top voted solution

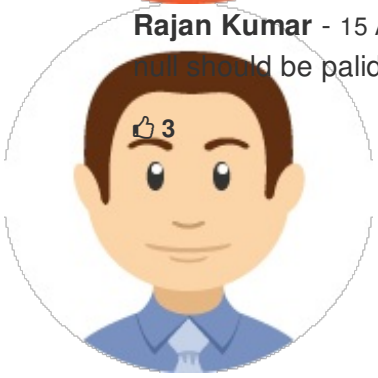
```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static boolean isStringPalindrome(String str){
    // Add your code below this line. Do not modify any other code.
    if(str==null) return false;
    int l = str.length()-1;
    for(int i = 0 ; i<l/2;i++){
        if(str.charAt(i)!=str.charAt(l-i)){
            return false;
        }
    }

    return true;
    // Add your code above this line. Do not modify any other code.
}
```

Comments

Zuzanna - 03 Jul, 2016

The null string is considered a palindrome and should return true not false?

3



Rajan Kumar - 15 Aug, 2016

null should be palidrome according to question. why this code worked?

3

Insert a Node at the Front of a Linked List

Linked Lists

Write a method to insert a node at the front of a singly-linked list and return the head of the modified list.

```
Examples:

LinkedList: 1->2 , Head = 1

InsertAtHead(Head,1) ==> 1->1->2
InsertAtHead(Head,2) ==> 2->1->2
InsertAtHead(Head,3) ==> 3->1->2
```

Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public ListNode insertAtHead(ListNode head, int data) {

    ListNode nHead = new ListNode(data);
    nHead.next = head;
    return nHead;



}
```

Top voted solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public ListNode insertAtHead(ListNode head, int data) {
    // Add your code below this line. Do not modify any other code.

    ListNode n = new ListNode(data);
```

```
    n.next = head;

    return n;

    // Add your code above this line. Do not modify any other code.
}
```

Comments

Anonymous - 23 Nov, 2015

This is right. Everyone seems to be checking for when head is null. However, if it is null your solution would account for it by default. Good one!

Replace all Spaces

Strings

Write a method to replace all spaces in a string with a given replacement string.

`replace("This is a test","/") --> "This/is/a/test"`

Note: Avoid using the in-built `String.replaceAll()` method.

Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static String replace(String a, String b) {
    StringBuilder aNoSpace = new StringBuilder();
    for(int i =  0; i < a.length(); i++){
        if(a.charAt(i) == ' ') {
            aNoSpace.append(b);
        } else {
            aNoSpace.append(a.charAt(i));
        }

    }
    return aNoSpace.toString();




}
```

Top voted solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static String replace(String a, String b) {
// Add your code below this line. Do not modify any other code.
```

```
StringBuilder builder = new StringBuilder();
for(int i = 0; i < a.length(); i++){
    char c = a.charAt(i);
    if(c == ' ') builder.append(b);
    else builder.append(c);
}
return builder.toString();

// Add your code above this line. Do not modify any other code.
}
```

1 2 3 4 5

Flip it!

Multi Dimensional Arrays

You are given an m x n 2D image matrix where each integer represents a pixel. Flip it **in-place** along its vertical axis.

Example:

Input image :

1 0
1 0

Modified to :

0 1
0 1

Your solution

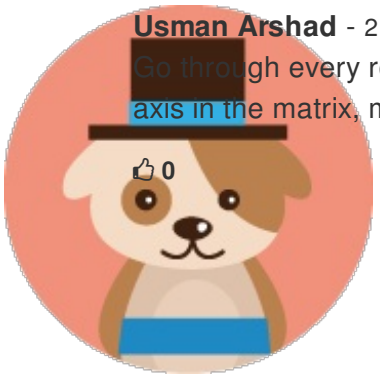
```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static void flipItVerticalAxis(int[][] matrix) {
    for (int y =0; y< matrix.length; y++){
        for (int x = 0; x < matrix[y].length / 2; x++){
            int temp = matrix[y][x];
            matrix[y][x] = matrix[y][matrix[y].length - 1 - x];
            matrix[y][matrix[y].length - 1 - x] = temp;
        }
    }

}
```

Top voted solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static void flipItVerticalAxis(int[][] matrix) {

    int rows = matrix.length;
    int columns = matrix[0].length;

    for (int y = 0; y < rows; y++) {
```

```
        for (int x = 0; x < columns / 2; x++) {
            int temp = matrix[y][x];

            matrix[y][x] = matrix[y][(columns - x) - 1];
            matrix[y][(columns - x) - 1] = temp;
        }
    }

}
```

Comments



Usman Arshad - 21 Mar, 2017

Go through every row in the matrix and just swap :). I've named the y and x iterators to reflect the x and y axis in the matrix, makes things easier to understand.

Delete a List's Head Node

Linked Lists

Given a singly-linked list, write a method to delete the first node of the list and return the new head.

```
Example:

LinkedList: 1->2->3 , Head = 1

deleteAtHead(Head) ==> 2->3
```

Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public ListNode deleteAtHead(ListNode head) {
    if(head == null)
        return null;
    ListNode temp = head.next;
    head.next = null;
    head = temp;
    return head;

}
```

Top voted solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public ListNode deleteAtHead(ListNode head) {

    return (head != null) ? head.next : null;

}
```

Comments

Tim Harris - 28 Mar, 2016

Wouldn't you still want to set the next reference of the current head node to null .. just in case it was being referenced from elsewhere in the application.

3

Jaydeep Ranipa - 30 Aug, 2016

I tried the same way.. But What I think is, this doesn't detach a head physically from the list, instead it just returns the next node to the head. Will this solution work in the case we want to modify our current list?

0

Nate Sun - 08 Nov, 2016

what is ? in the line for? it's a little hard for me to read honestly, could you please explain?

1

Tim Harris - 31 Dec, 2016

@Nate that's a classic ternary operator. What it's saying is return (if head != null) then head.next, else null.

1

Kranthi Kumar - 25 Feb, 2017

@Tim @Jaydeep: References are passed by value in Java. If there was a head pointing to the first node outside of the scope of the current method then that reference will keep pointing to the first node regardless of what you do. The method is probably being called like this: head = deleteAtHead(head); . Please do correct me if I am wrong.

2

Tim Harris - 26 Feb, 2017

@Kranthi so if I understand correctly, you're saying that if something points to head outside of the method, it will continue to do so which is correct. However, if you set head.next = null inside the method the

reference. From the head to the next node will be cleared. Why's that necessary? To allow for better garbage collection.

Delete a List's Tail Node

Linked Lists

Given a singly-linked list, write a method to delete its last node and return the head.

Example:

```
1->2->3->4 ==> 1->2->3
```

Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public ListNode deleteAtTail(ListNode head) {

    if (head == null || head.next == null)
        return null;
    ListNode ptr = head;

        while(ptr != null && ptr.next != null && ptr.next.next != null){
        ptr = ptr.next;
    }
    ptr.next = null;
    return head;

}
```

Top voted solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public ListNode deleteAtTail(ListNode head) {
    // Add your code below this line. Do not modify any other code.
    if(head==null || head.next==null){
        return null;
    }
    ListNode tmp=head;
    while(tmp.next.next!=null){
        tmp=tmp.next;
    }
    tmp.next=null;
```

```
    return head;
    // Add your code above this line. Do not modify any other code.
}
```

1 2 3 4 5

Bubble Sort

Sorting Algorithms | Arrays | Numbers

Write a method that takes in an array of `int`s and uses the Bubble Sort algorithm to sort the array 'in place' in ascending order. The method should return the same, in-place sorted array.

Note: Bubble sort is one of the most inefficient ways to sort a large array of integers. Nevertheless, it is an interview favorite. Bubble sort has a time complexity of $O(n^2)$. However, if the sample size is small, bubble sort provides a simple implementation of a classic sorting algorithm.

Examples:

`bubbleSortArray({5,4,3}) -> {3,4,5}`

`bubbleSortArray({3}) -> {3}`

`bubbleSortArray({}) -> {}`

`{} -> [Empty] Array`

Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static int[] bubbleSortArray(int[] arr){

    for(int i = 0; i < arr.length; i++){
        for(int j = 1; j < arr.length - i; j++){
            if (arr[j - 1] > arr[j]){
                int temp = arr[j];
                arr[j] = arr[j-1];
                arr[j-1] = temp;
            }
        }
    }


    return arr;
}
```

Top voted solution

```java
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static int[] bubbleSortArray(int[] arr){
    if (arr.length <= 1) return arr;

    for (int i = arr.length; i >= 0; i--) {
        for (int j = 0; j + 1 < i; j++) {
            if (arr[j] > arr[j + 1]) {
                int tmp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = tmp;
            }
        }
    }

    return arr;
}
```

1 2 3 4 5

Insert a Node at the End of a Linked List

Linked Lists

Write a method to insert a node at the end of a singly-linked list. Return the head of the modified list.

```
Examples:

LinkedList: 1->2 , Head = 1

InsertAtTail(Head,1) ==> 1->2->1
InsertAtTail(Head,2) ==> 1->2->2
InsertAtTail(Head,3) ==> 1->2->3
```

Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public ListNode insertAtTail(ListNode head, int data) {

    if(head == null){
        head = new ListNode(data);
        return head;
    }

    ListNode ptr = head;
    while(ptr != null && ptr.next != null){
        ptr = ptr.next;
    }
    ptr.next = new ListNode(data);
    return head;


}
```

Top voted solution

```
public ListNode insertAtTail(ListNode head, int data) {
```

```
    // Add your code below this line. Do not modify any other code.
    ListNode newNode = new ListNode(data);
    if (head == null) {
        head = newNode;
    } else {
        ListNode curr = head;
        // Find the last node in the list
        while (curr.next != null) {
            curr = curr.next;
        }
        // insert the new node at the end
        curr.next = newNode;
    }
    return head;
    // Add your code above this line. Do not modify any other code.
}
```

Find the Middle of a List in a Single Pass

Linked Lists

Given a Singly-Linked List, write a method - `findMiddleNode` that finds and returns the middle node of the list in a **single pass**.

Examples:

`1 ==> 1`

`1->2 ==> 1`

`1->2->3->4 ==> 2`

`1->2->3->4->5 ==> 3`

Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public ListNode findMiddleNode(ListNode head) {
    if (head == null)
        return null;

    ListNode slowP = head;
    ListNode fastP = head;

    while(fastP != null && fastP.next != null && fastP.next.next != null){
        slowP = slowP.next;
        fastP = fastP.next.next;
    }

    return slowP;
}
```

Top voted solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public ListNode findMiddleNode(ListNode head) {

 ListNode slow_ptr = head;
        ListNode fast_ptr = head;
```

```
        if (head != null)
        {
            while (fast_ptr.next!= null && fast_ptr.next.next != null)
            {
                fast_ptr = fast_ptr.next.next;
                slow_ptr = slow_ptr.next;
            }


        }


    return slow_ptr;
}
```

Comments

Carson Bradshaw - 16 Jun, 2017

I considered this solution but isn't this technically 1.5 passes not a single pass because the slow_ptr has to pass through half the list and fast_ptr must pass through the whole list?

0

Mohit Banerjee - 01 Jul, 2017

@Carson, Its technically a singe pass only. Look at the while loop. How many iterations does it take to reach the end of the list? Iterations equivalent to single pass over the list if not less.

0

Unique Chars in a String

Strings

Write a method that takes in an input `String` and returns `true` if all the characters in the `String` are unique and `false` if there is even a single repeated character.
The method should return `true` if the input is `null` or empty `String`.

Examples:
`areAllCharactersUnique("abcde") -> true`

`areAllCharactersUnique("aa") -> false`

`areAllCharactersUnique("") -> true`

`areAllCharactersUnique(null) -> true`

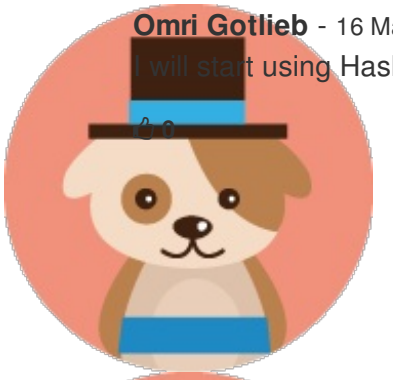
Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static boolean areAllCharactersUnique(String str){

    if (str == null || str.equals(""))
        return true;

    Set<Character> dict = new HashSet<Character>();
    for (int i = 0; i < str.length(); i++){
        char c = Character.toLowerCase(str.charAt(i));
        if(dict.contains(c))
            return false;
        else
            dict.add(c);
    }

    return true;

}
```

Top voted solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static boolean areAllCharactersUnique(String str){
    if(str == null) return true;
    HashSet<Character> hs = new HashSet<Character>();
    for(int i = 0; i < str.length(); i++) {
        if(!hs.add(str.charAt(i))) {
            return false;
        }
    }
    return true;
}
```

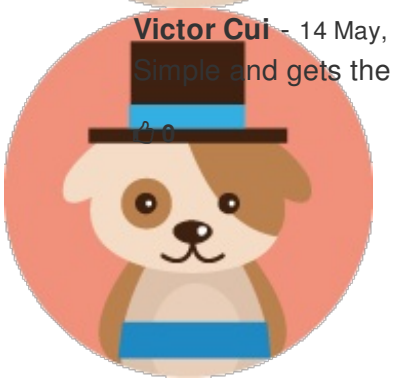
Comments



Omri Gotlieb - 16 Mar, 2017

I will start using HashSet :)

0



Victor Cui - 14 May, 2017

Simple and gets the job done- awesome! It's useful that the add() method of HashSet returns a boolean

0

1 2 3 4 5

Find the First Non Duplicate Character in a String

Strings Hash-Tables

Find the **first** non-duplicate character in a string. Return null if no unique character is found.

`firstNonRepeatedCharacter( "abcdcd" ) --> 'a'`
`firstNonRepeatedCharacter( "cbcd" ) --> 'b'`
`firstNonRepeatedCharacter( "cdcd" ) --> null`

Your solution

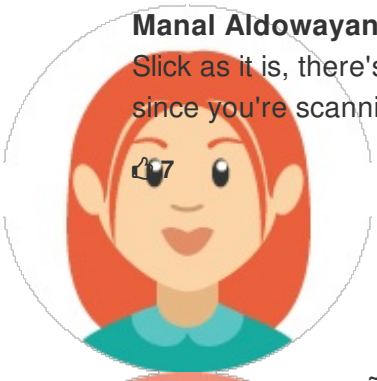
```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static Character firstNonRepeatedCharacter(String str) {

    if(str == null)
        return null;

    Set<Character> dict = new LinkedHashSet<Character>();
    for (int i = 0; i < str.length(); i++){
        char c = Character.toLowerCase(str.charAt(i));
        if (dict.contains(c)){
            dict.remove(c);
        } else {
            dict.add(c);
        }
    }

    if(dict.isEmpty())
        return null;
    return dict.iterator().next();

}
```

Top voted solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.
```

```
public static Character firstNonRepeatedCharacter(String str) {
    // Add your code below this line. Do not modify any other code.

    for (char c : str.toCharArray()) {
        if (str.indexOf(c) == str.lastIndexOf(c)) {
            return c;
        }
    }

    return null;


    // Add your code above this line. Do not modify any other code.
}
```

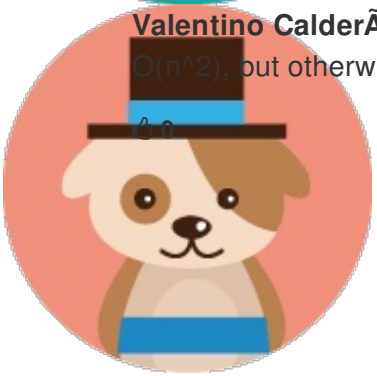
Comments



Manal Aldowayan - 05 May, 2016

Slick as it is, there's a small chance the interviewer may flag this solution for using too much 'magic'. Plus since you're scanning the String with each character in .lastIndexOf(), won't this exceed linear runtime?

7



Valentino CalderÃ³n - 01 Dec, 2016

O(n^2), but otherwise very clean

0

Reverse a string

Strings

Write a method that takes in a `String` and returns the reversed version of the `String`.

```
Examples:
reverseString("abcde") -> "edcba"
reverseString("1") -> "1"
reverseString("") -> ""
reverse("madam") -> "madam"
reverse(null) -> null
```

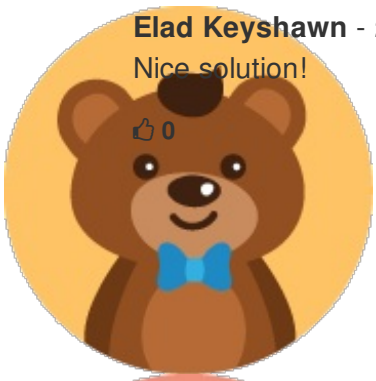
Your solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static String reverseString(String str){
    String inputString = str;
    String outputString = null;
    // StringBuilder rString = new StringBuilder();

    // if(str == null){
    //    return outputString;
    // }

    // for(int i = inputString.length() - 1; i >= 0; --i){
    //    rString.append(inputString.charAt(i));
    // }

    // outputString = rString.toString();

    if(str == null)
        return outputString;
    // if(str.equals(""))
    //    return (outputString = "");

    outputString = new StringBuilder(inputString).reverse().toString();

    return outputString;
}
```

Top voted solution

```
// java.util.* has been imported for this problem.
// You don't need any other imports.

public static String reverseString(String str){
    if(str == null) return null;
    if(str.length() == 0) return "";

    return reverseString(str.substring(1)) + str.charAt(0);
}
```

Comments



Elad Keyshawn - 23 Jul, 2016

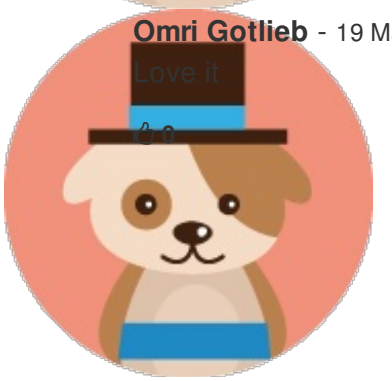
Nice solution!

0

Steve Mason - 07 Oct, 2016

I like the recursive solutions as well.

2



Omri Gotlieb - 19 Mar, 2017

Love it

0

Jin Wang - 24 Apr, 2017

neat!

0