

# Problem 148: Sort List

## Problem Information

Difficulty: **Medium**

Acceptance Rate: 0.00%

Paid Only: No

## Problem Description

Given the

head

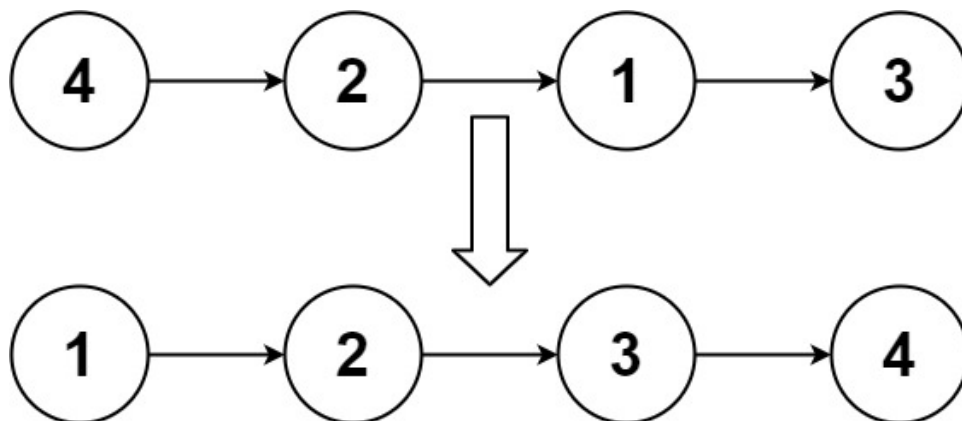
of a linked list, return

the list after sorting it in

ascending order

.

Example 1:



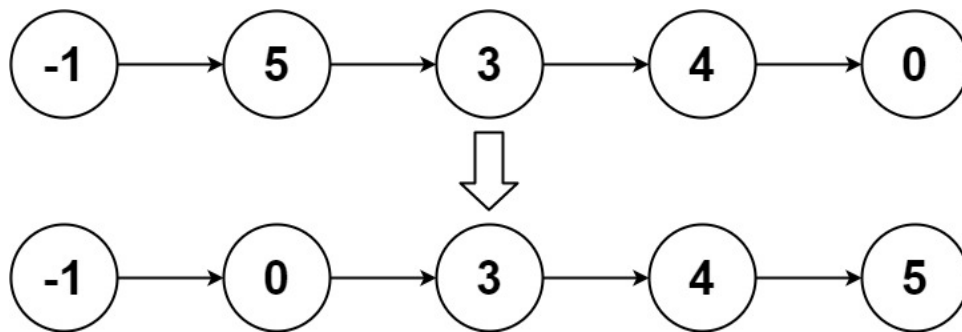
Input:

head = [4,2,1,3]

Output:

[1,2,3,4]

Example 2:



Input:

head = [-1,5,3,4,0]

Output:

[-1,0,3,4,5]

Example 3:

Input:

head = []

Output:

[]

Constraints:

The number of nodes in the list is in the range

[0, 5 \* 10

4

]

.

-10

5

`<= Node.val <= 10`

5

Follow up:

Can you sort the linked list in

$O(n \log n)$

time and

$O(1)$

memory (i.e. constant space)?

## Code Snippets

**C++:**

```
/**
 * 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* sortList(ListNode* head) {

}
};

```

## Java:

```

/**
 * Definition for singly-linked list.
 * public class ListNode {
 *   int val;
 *   ListNode next;
 *   ListNode() {}
 *   ListNode(int val) { this.val = val; }
 *   ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
public ListNode sortList(ListNode head) {

}

}

```

## Python3:

```

# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def sortList(self, head: Optional[ListNode]) -> Optional[ListNode]:

```

## Python:

```

# Definition for singly-linked list.
# class ListNode(object):
#     def __init__(self, val=0, next=None):

```

```

# self.val = val
# self.next = next
class Solution(object):
def sortList(self, head):
    """
    :type head: Optional[ListNode]
    :rtype: Optional[ListNode]
    """

```

### JavaScript:

```

/**
 * Definition for singly-linked list.
 * function ListNode(val, next) {
 *   this.val = (val===undefined ? 0 : val)
 *   this.next = (next===undefined ? null : next)
 * }
 */
/**
 * @param {ListNode} head
 * @return {ListNode}
 */
var sortList = function(head) {

};

```

### TypeScript:

```

/**
 * Definition for singly-linked list.
 * class ListNode {
 *   val: number
 *   next: ListNode | null
 *   constructor(val?: number, next?: ListNode | null) {
 *     this.val = (val===undefined ? 0 : val)
 *     this.next = (next===undefined ? null : next)
 *   }
 * }
 */

function sortList(head: ListNode | null): ListNode | null {

```

```
};
```

### C#:

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 * public int val;
 * public ListNode next;
 * public ListNode(int val=0, ListNode next=null) {
 * this.val = val;
 * this.next = next;
 * }
 * }
 */
public class Solution {
public ListNode SortList(ListNode head) {

}
}
```

### C:

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 * int val;
 * struct ListNode *next;
 * };
 */
struct ListNode* sortList(struct ListNode* head) {

}
```

### Go:

```
/**
 * Definition for singly-linked list.
 * type ListNode struct {
 * Val int
 * Next *ListNode
 * }
 */
```

```

*/
func sortList(head *ListNode) *ListNode {

}

```

## Kotlin:

```

/**
 * Example:
 * var li = ListNode(5)
 * var v = li.`val`
 * Definition for singly-linked list.
 * class ListNode(var `val`: Int) {
 *   var next: ListNode? = null
 * }
 */
class Solution {
fun sortList(head: ListNode?): ListNode? {

}

}

```

## Swift:

```

/**
 * Definition for singly-linked list.
 * public class ListNode {
 *   public var val: Int
 *   public var next: ListNode?
 *   public init() { self.val = 0; self.next = nil; }
 *   public init(_ val: Int) { self.val = val; self.next = nil; }
 *   public init(_ val: Int, _ next: ListNode?) { self.val = val; self.next =
next; }
 * }
 */
class Solution {
func sortList(_ head: ListNode?) -> ListNode? {

}

}

```

## Rust:

```

// Definition for singly-linked list.
// #[derive(PartialEq, Eq, Clone, Debug)]
// pub struct ListNode {
//     pub val: i32,
//     pub next: Option<Box<ListNode>>
// }
//
// impl ListNode {
//     #[inline]
//     fn new(val: i32) -> Self {
//         ListNode {
//             next: None,
//             val
//         }
//     }
// }
impl Solution {
    pub fn sort_list(head: Option<Box<ListNode>>) -> Option<Box<ListNode>> {

    }
}

```

## Ruby:

```

# Definition for singly-linked list.
# class ListNode
#   attr_accessor :val, :next
#   def initialize(val = 0, _next = nil)
#     @val = val
#     @next = _next
#   end
# end
# @param {ListNode} head
# @return {ListNode}
def sort_list(head)

end

```

## PHP:

```

/**
 * Definition for a singly-linked list.
 * class ListNode {

```



```

* public $val = 0;
* public $next = null;
* function __construct($val = 0, $next = null) {
*     $this->val = $val;
*     $this->next = $next;
* }
* }
*/
class Solution {

    /**
     * @param ListNode $head
     * @return ListNode
     */
    function sortList($head) {

    }

}

```

### Dart:

```

/**
 * Definition for singly-linked list.
 * class ListNode {
 *     int val;
 *     ListNode? next;
 *     ListNode([this.val = 0, this.next]);
 * }
 */
class Solution {
    ListNode? sortList(ListNode? head) {

    }

}

```

### Scala:

```

/**
 * Definition for singly-linked list.
 * class ListNode(_x: Int = 0, _next: ListNode = null) {
 *     var next: ListNode = _next
 *     var x: Int = _x
 * }
 */

```

```

* }
*/
object Solution {
def sortList(head: ListNode): ListNode = {

}
}

```

## Elixir:

```

# Definition for singly-linked list.
#
# defmodule ListNode do
# @type t :: %__MODULE__{
#   val: integer,
#   next: ListNode.t() | nil
# }
# defstruct val: 0, next: nil
# end

defmodule Solution do
@spec sort_list(head :: ListNode.t | nil) :: ListNode.t | nil
def sort_list(head) do

end
end

```

## Erlang:

```

%% Definition for singly-linked list.
%%
%% -record(list_node, {val = 0 :: integer(),
%%   next = null :: 'null' | #list_node{}}).

-spec sort_list(Head :: #list_node{} | null) -> #list_node{} | null.
sort_list(Head) ->
.

```

## Racket:

```

; Definition for singly-linked list:
#|

```

```

; val : integer?
; next : (or/c list-node? #f)
(struct list-node
  (val next) #:mutable #:transparent)

; constructor
(define (make-list-node [val 0])
  (list-node val #f))

|#

(define/contract (sort-list head)
  (-> (or/c list-node? #f) (or/c list-node? #f))
  )

```

## Solutions

### C++ Solution:

```

/*
 * Problem: Sort List
 * Difficulty: Medium
 * Tags: array, sort, linked_list
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * 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* sortList(ListNode* head) {

    }

};

```

### Java Solution:

```

/**
 * Problem: Sort List
 * Difficulty: Medium
 * Tags: array, sort, linked_list
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {
 *         // TODO: Implement optimized solution
 *     }
 *     return 0;
 * }
 *
 * ListNode(int val) { this.val = val; }
 * ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */

class Solution {
    public ListNode sortList(ListNode head) {

    }

}

```

### Python3 Solution:

```

"""
Problem: Sort List
Difficulty: Medium
Tags: array, sort, linked_list

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

# Definition for singly-linked list.
# class ListNode:
# def __init__(self, val=0, next=None):
# self.val = val
# self.next = next
class Solution:
def sortList(self, head: Optional[ListNode]) -> Optional[ListNode]:
# TODO: Implement optimized solution
pass

```

### Python Solution:

```

# Definition for singly-linked list.
# class ListNode(object):
# def __init__(self, val=0, next=None):
# self.val = val
# self.next = next
class Solution(object):
def sortList(self, head):
"""
:type head: Optional[ListNode]
:rtype: Optional[ListNode]
"""

```

### JavaScript Solution:

```

/**
 * Problem: Sort List
 * Difficulty: Medium
 * Tags: array, sort, linked_list
 *
 * Approach: Use two pointers or sliding window technique

```

```

* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Definition for singly-linked list.
* function ListNode(val, next) {
*   this.val = (val===undefined ? 0 : val)
*   this.next = (next===undefined ? null : next)
* }
*/
/**
* @param {ListNode} head
* @return {ListNode}
*/
var sortList = function(head) {

};

```

## TypeScript Solution:

```

/**
* Problem: Sort List
* Difficulty: Medium
* Tags: array, sort, linked_list
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Definition for singly-linked list.
* class ListNode {
*   val: number
*   next: ListNode | null
*   constructor(val?: number, next?: ListNode | null) {
*     this.val = (val===undefined ? 0 : val)
*     this.next = (next===undefined ? null : next)
*   }
* }

```

```

*/

function sortList(head: ListNode | null): ListNode | null {

};

```

### C# Solution:

```

/*
 * Problem: Sort List
 * Difficulty: Medium
 * Tags: array, sort, linked_list
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * Definition for singly-linked list.
 * public class ListNode {
 * public int val;
 * public ListNode next;
 * public ListNode(int val=0, ListNode next=null) {
 * this.val = val;
 * this.next = next;
 * }
 * }
 */

public class Solution {
    public ListNode SortList(ListNode head) {

    }
}

```

### C Solution:

```

/*
 * Problem: Sort List
 * Difficulty: Medium
 * Tags: array, sort, linked_list

```

```

*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Definition for singly-linked list.
* struct ListNode {
*   int val;
*   struct ListNode *next;
* };
*/
struct ListNode* sortList(struct ListNode* head) {

}

```

### Go Solution:

```

// Problem: Sort List
// Difficulty: Medium
// Tags: array, sort, linked_list
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

/**
* Definition for singly-linked list.
* type ListNode struct {
*   Val int
*   Next *ListNode
* }
*/
func sortList(head *ListNode) *ListNode {

}

```

### Kotlin Solution:



```

/**
 * Example:
 * var li = ListNode(5)
 * var v = li.`val`
 * Definition for singly-linked list.
 * class ListNode(var `val`: Int) {
 *   var next: ListNode? = null
 * }
 */
class Solution {
fun sortList(head: ListNode?): ListNode? {

}

}

```

### Swift Solution:

```

/**
 * Definition for singly-linked list.
 * public class ListNode {
 *   public var val: Int
 *   public var next: ListNode?
 *   public init() { self.val = 0; self.next = nil; }
 *   public init(_ val: Int) { self.val = val; self.next = nil; }
 *   public init(_ val: Int, _ next: ListNode?) { self.val = val; self.next =
next; }
 * }
 */
class Solution {
func sortList(_ head: ListNode?) -> ListNode? {

}

}

```

### Rust Solution:

```

// Problem: Sort List
// Difficulty: Medium
// Tags: array, sort, linked_list
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)

```

```

// Space Complexity: O(1) to O(n) depending on approach

// Definition for singly-linked list.
// #[derive(PartialEq, Eq, Clone, Debug)]
// pub struct ListNode {
//     pub val: i32,
//     pub next: Option<Box<ListNode>>
// }
//
// impl ListNode {
//     #[inline]
//     fn new(val: i32) -> Self {
//         ListNode {
//             next: None,
//             val
//         }
//     }
// }
impl Solution {
    pub fn sort_list(head: Option<Box<ListNode>>) -> Option<Box<ListNode>> {

    }
}

```

### Ruby Solution:

```

# Definition for singly-linked list.
# class ListNode
#   attr_accessor :val, :next
#   def initialize(val = 0, _next = nil)
#     @val = val
#     @next = _next
#   end
# end
# @param {ListNode} head
# @return {ListNode}
def sort_list(head)

end

```

### PHP Solution:

```

/**
 * Definition for a singly-linked list.
 * class ListNode {
 * public $val = 0;
 * public $next = null;
 * function __construct($val = 0, $next = null) {
 * $this->val = $val;
 * $this->next = $next;
 * }
 * }
 */
class Solution {

/**
 * @param ListNode $head
 * @return ListNode
 */
function sortList($head) {

}

}

```

### Dart Solution:

```

/**
 * Definition for singly-linked list.
 * class ListNode {
 * int val;
 * ListNode? next;
 * ListNode([this.val = 0, this.next]);
 * }
 */
class Solution {
  ListNode? sortList(ListNode? head) {

  }

}

```

### Scala Solution:

```

/**
 * Definition for singly-linked list.

```

```

* class ListNode(_x: Int = 0, _next: ListNode = null) {
*   var next: ListNode = _next
*   var x: Int = _x
* }
*/
object Solution {
def sortList(head: ListNode): ListNode = {

}
}

```

### Elixir Solution:

```

# Definition for singly-linked list.
#
# defmodule ListNode do
#   @type t :: %__MODULE__{
#     val: integer,
#     next: ListNode.t() | nil
#   }
#   defstruct val: 0, next: nil
# end

defmodule Solution do
  @spec sort_list(head :: ListNode.t | nil) :: ListNode.t | nil
  def sort_list(head) do

  end
end

```

### Erlang Solution:

```

%% Definition for singly-linked list.
%%
%% -record(list_node, {val = 0 :: integer(),
%%   next = null :: 'null' | #list_node{}}).

-spec sort_list(Head :: #list_node{} | null) -> #list_node{} | null.
sort_list(Head) ->
.

```

## Racket Solution:

```
; Definition for singly-linked list:
#|

; val : integer?
; next : (or/c list-node? #f)
(struct list-node
  (val next) #:mutable #:transparent)

; constructor
(define (make-list-node [val 0])
  (list-node val #f))

|#

(define/contract (sort-list head)
  (-> (or/c list-node? #f) (or/c list-node? #f))
  )
```