

# Problem 86: Partition List

## Problem Information

**Difficulty:** Medium

**Acceptance Rate:** 0.00%

**Paid Only:** No

## Problem Description

Given the

head

of a linked list and a value

x

, partition it such that all nodes

less than

x

come before nodes

greater than or equal

to

x

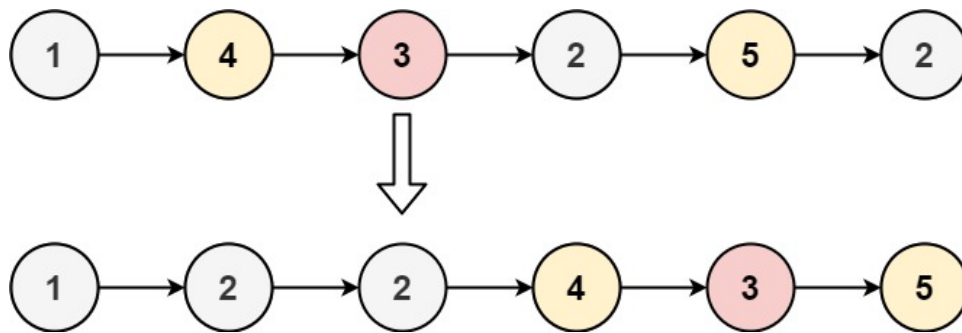
.

You should

preserve

the original relative order of the nodes in each of the two partitions.

Example 1:



Input:

head = [1,4,3,2,5,2], x = 3

Output:

[1,2,2,4,3,5]

Example 2:

Input:

head = [2,1], x = 2

Output:

[1,2]

Constraints:

The number of nodes in the list is in the range

[0, 200]

.

-100 <= Node.val <= 100

-200 <= x <= 200

## 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* partition(ListNode* head, int x) {

    }
};
```

### 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 partition(ListNode head, int x) {

    }
}
```

```
}
```

### Python3:

```
# Definition for singly-linked list.
# class ListNode:
# def __init__(self, val=0, next=None):
# self.val = val
# self.next = next
class Solution:
def partition(self, head: Optional[ListNode], x: int) -> 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 partition(self, head, x):
    """
    :type head: Optional[ListNode]
    :type x: int
    :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
 * @param {number} x
 * @return {ListNode}
 */
var partition = function(head, x) {
```

```
};
```

### 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 partition(head: ListNode | null, x: number): 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 Partition(ListNode head, int x) {

    }
}
```

### C:

```

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

}

```

### Go:

```

/**
 * Definition for singly-linked list.
 * type ListNode struct {
 *   Val int
 *   Next *ListNode
 * }
 */
func partition(head *ListNode, x int) *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 partition(head: ListNode?, x: Int): 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 partition(_ head: ListNode?, _ x: Int) -> 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 partition(head: Option<Box<ListNode>>, x: i32) ->
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
# @param {Integer} x
# @return {ListNode}
def partition(head, x)

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
 * @param Integer $x
 * @return ListNode
 */
function partition($head, $x) {

}

}
```

## Dart:

```

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

  }
}

```

## 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 partition(head: ListNode, x: Int): 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 partition(head :: ListNode.t | nil, x :: integer) :: ListNode.t | nil

```

```

def partition(head, x) do

end

end

```

## Erlang:

```

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

-spec partition(Head :: #list_node{} | null, X :: integer()) -> #list_node{}
| null.
partition(Head, X) ->
.

```

## 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 (partition head x)
  (-> (or/c list-node? #f) exact-integer? (or/c list-node? #f))
  )

```

## Solutions

### C++ Solution:

```

/*
 * Problem: Partition List
 * Difficulty: Medium
 * Tags: array, 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* partition(ListNode* head, int x) {

    }
};

```

## Java Solution:

```

/**
 * Problem: Partition List
 * Difficulty: Medium
 * Tags: array, 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 partition(ListNode head, int x) {

}
}

```

### Python3 Solution:

```

"""
Problem: Partition List
Difficulty: Medium
Tags: array, 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 partition(self, head: Optional[ListNode], x: int) -> 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 partition(self, head, x):
    """
    :type head: Optional[ListNode]
    :type x: int
    :rtype: Optional[ListNode]
    """

```

### JavaScript Solution:

```

/**
 * Problem: Partition List
 * Difficulty: Medium
 * Tags: array, 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
 * @param {number} x
 * @return {ListNode}
 */
var partition = function(head, x) {

};

```

### TypeScript Solution:

```

/**
 * Problem: Partition List
 * Difficulty: Medium
 * Tags: array, 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 partition(head: ListNode | null, x: number): ListNode | null {

};

```

## C# Solution:

```

/*
 * Problem: Partition List
 * Difficulty: Medium
 * Tags: array, 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 Partition(ListNode head, int x) {

}
}

```

### C Solution:

```

/*
* Problem: Partition List
* Difficulty: Medium
* Tags: array, 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* partition(struct ListNode* head, int x) {

}

```

### Go Solution:

```

// Problem: Partition List
// Difficulty: Medium
// Tags: array, 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 partition(head *ListNode, x int) *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 partition(head: ListNode?, x: Int): 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 partition(_ head: ListNode?, _ x: Int) -> ListNode? {

}
}

```

## Rust Solution:

```

// Problem: Partition List
// Difficulty: Medium
// Tags: array, 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 partition(head: Option<Box<ListNode>>, x: i32) ->
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  
# @param {Integer} x  
# @return {ListNode}  
def partition(head, x)  
  
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  
     * @param Integer $x  
     * @return ListNode  
     */  
    function partition($head, $x) {
```

```
}  
}
```

### Dart Solution:

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

### 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 partition(head: ListNode, x: Int): 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 partition(head :: ListNode.t() | nil, x :: integer) :: ListNode.t() | nil
  def partition(head, x) do

  end
end

```

### Erlang Solution:

```

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

-spec partition(Head :: #list_node{} | null, X :: integer()) -> #list_node{}
| null.
partition(Head, X) ->
.

```

### 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 (partition head x)
  (-> (or/c list-node? #f) exact-integer? (or/c list-node? #f))
)
```