Problem 3294: Convert Doubly Linked List to Array II

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

You are given an

arbitrary

node

from a

doubly linked list

, which contains nodes that have a next pointer and a previous pointer.

Return an integer array which contains the elements of the linked list

in order

.

Example 1:

Input:

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

Output:

[1,2,3,4,5]

Example 2:

Input:

head = [4,5,6,7,8], node = 8

Output:

[4,5,6,7,8]

Constraints:

The number of nodes in the given list is in the range

[1, 500]

.

1 <= Node.val <= 1000

All nodes have unique

Node.val

.

Code Snippets

C++:

```
/**
* Definition for doubly-linked list.
* class Node {
* int val;
* Node* prev;
* Node* next;
* Node() : val(0), next(nullptr), prev(nullptr) {}
* Node(int x) : val(x), next(nullptr), prev(nullptr) {}
* Node(int x, Node *prev, Node *next) : val(x), next(next), prev(prev) {}
* };
*/
class Solution {
public:
vector<int> toArray(Node *node){

}
};
```

Java:

```
/*
// Definition for a Node.
class Node {
public int val;
public Node prev;
public Node next;
};
*/

class Solution {
public int[] toArray(Node node) {

}
}
```

Python3:

```
"""
# Definition for a Node.
class Node:
def __init__(self, val, prev=None, next=None):
self.val = val
self.prev = prev
self.next = next
"""
class Solution:
def toArray(self, node: 'Optional[Node]') -> List[int]:
```

Python:

```
"""
# Definition for a Node.
class Node:
def __init__(self, val, prev=None, next=None):
self.val = val
self.prev = prev
self.next = next
"""

class Solution:
def toArray(self, node):
"""
:type head: Node
:rtype: List[int]
"""
```

JavaScript:

```
/**
* // Definition for a _Node.
* function _Node(val,prev,next) {
* this.val = val;
* this.prev = prev;
* this.next = next;
* };
*/

/**
* @param {_Node} head
* @return {number[]}
*/
var toArray = function(node) {

};
```

TypeScript:

```
/**
* Definition for _Node.
* class _Node {
* val: number
* prev: _Node | null
* next: _Node | null
*
* constructor(val?: number, prev? : _Node, next? : _Node) {
* this.val = (val===undefined ? 0 : val);
* this.prev = (prev===undefined ? null : prev);
* this.next = (next===undefined ? null : next);
* }
* }
*/


function toArray(node: _Node | null): number[] {

};
```

C#:

```
/*
// Definition for a Node.
public class Node {
public int val;
public Node prev;
public Node next;
}
*/

public class Solution {
public int[] ToArray(Node node) {

}
}
```

C:

```
/*
// Definition for a Node.
struct Node {
int val;
struct Node* next;
struct Node* prev;
};
```

```
*/

int* toArray(struct Node *node, int *returnSize) {

}
```

Go:

```
/**
* Definition for a Node.
* type Node struct {
* Val int
* Next *Node
* Prev *Node
* }
*/

func toArray(head *node) []int {

}
```

Kotlin:

```
/**
* Definition for a Node.
* class Node(var `val`: Int) {
* var prev: Node? = null
* var next: Node? = null
* }
*/

class Solution {
fun toArray(node: Node?): IntArray {

}
}
```

PHP:

```
/**
* Definition for a Node.
* class Node {
```

```
* public $val = null;
* public $prev = null;
* public $next = null;
* function __construct($val = 0) {
* $this->val = $val;
* $this->prev = null;
* $this->next = null;
* }
* }
*/

class Solution {
/**
* @param Node $head
* @return Node
*/
function toArray($node) {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Convert Doubly Linked List to Array II
* 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 doubly-linked list.
* class Node {
* int val;
* Node* prev;
* Node* next;
* Node() : val(0), next(nullptr), prev(nullptr) {}
* Node(int x) : val(x), next(nullptr), prev(nullptr) {}
* Node(int x, Node *prev, Node *next) : val(x), next(next), prev(prev) {}
* };
*/
class Solution {
public:
vector<int> toArray(Node *node){

}
};
```

Java Solution:

```
/**
* Problem: Convert Doubly Linked List to Array II
* 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 a Node.
class Node {
public int val;
public Node prev;
public Node next;
};
*/

class Solution {
public int[] toArray(Node node) {

}
}
```

Python3 Solution:

```
"""
Problem: Convert Doubly Linked List to Array II
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 a Node.
class Node:
def __init__(self, val, prev=None, next=None):
self.val = val
self.prev = prev
self.next = next
"""
class Solution:
def toArray(self, node: 'Optional[Node]') -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
"""
# Definition for a Node.
class Node:
def __init__(self, val, prev=None, next=None):
self.val = val
self.prev = prev
self.next = next
"""

class Solution:
def toArray(self, node):
"""
:type head: Node
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Convert Doubly Linked List to Array II
* 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 a _Node.
* function _Node(val,prev,next) {
* this.val = val;
* this.prev = prev;
* this.next = next;
* };
*/

/**
* @param {_Node} head
* @return {number[]}
*/
var toArray = function(node) {

};
```

TypeScript Solution:

```
/**
* Problem: Convert Doubly Linked List to Array II
* 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 _Node.
* class _Node {
```

```
* val: number
* prev: _Node | null
* next: _Node | null
*
* constructor(val?: number, prev? : _Node, next? : _Node) {
* this.val = (val===undefined ? 0 : val);
* this.prev = (prev===undefined ? null : prev);
* this.next = (next===undefined ? null : next);
* }
* }
*/


function toArray(node: _Node | null): number[] {

};
```

C# Solution:

```
/*
* Problem: Convert Doubly Linked List to Array II
* 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 a Node.
public class Node {
public int val;
public Node prev;
public Node next;
}
*/

public class Solution {
public int[] ToArray(Node node) {
}
}
```

C Solution:

```
/*
* Problem: Convert Doubly Linked List to Array II
* 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 a Node.
struct Node {
int val;
struct Node* next;
struct Node* prev;
};
*/

int* toArray(struct Node *node, int *returnSize) {

}
```

Go Solution:

```
// Problem: Convert Doubly Linked List to Array II
// 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 a Node.
* type Node struct {
```

```
* Val int
* Next *Node
* Prev *Node
* }
*/

func toArray(head *node) []int {

}
```

Kotlin Solution:

```
/**
* Definition for a Node.
* class Node(var `val`: Int) {
* var prev: Node? = null
* var next: Node? = null
* }
*/

class Solution {
fun toArray(node: Node?): IntArray {

}
}
```

PHP Solution:

```
/**
* Definition for a Node.
* class Node {
* public $val = null;
* public $prev = null;
* public $next = null;
* function __construct($val = 0) {
* $this->val = $val;
* $this->prev = null;
* $this->next = null;
* }
* }
*/
```

```
class Solution {
/**
* @param Node $head
* @return Node
*/
function toArray($node) {

}
}
```