Problem 1634: Add Two Polynomials Represented as Linked Lists

Problem Information

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

Problem Description

A polynomial linked list is a special type of linked list where every node represents a term in a polynomial expression.

Each node has three attributes:

coefficient

: an integer representing the number multiplier of the term. The coefficient of the term

9

x

4

is

9

.

power

: an integer representing the exponent. The power of the term

9x

4

is

4

.

next

: a pointer to the next node in the list, or

null

if it is the last node of the list.

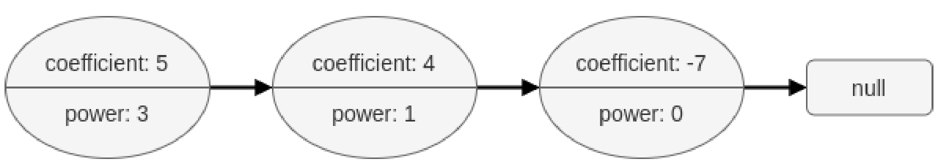
For example, the polynomial

5x

3

+ 4x - 7

is represented by the polynomial linked list illustrated below:



The polynomial linked list must be in its standard form: the polynomial must be in

strictly

descending order by its

power

value. Also, terms with a

coefficient

of

0

are omitted.

Given two polynomial linked list heads,

poly1

and

poly2

, add the polynomials together and return

the head of the sum of the polynomials

.

PolyNode

format:

The input/output format is as a list of

n

nodes, where each node is represented as its

[coefficient, power]

. For example, the polynomial

5x

3

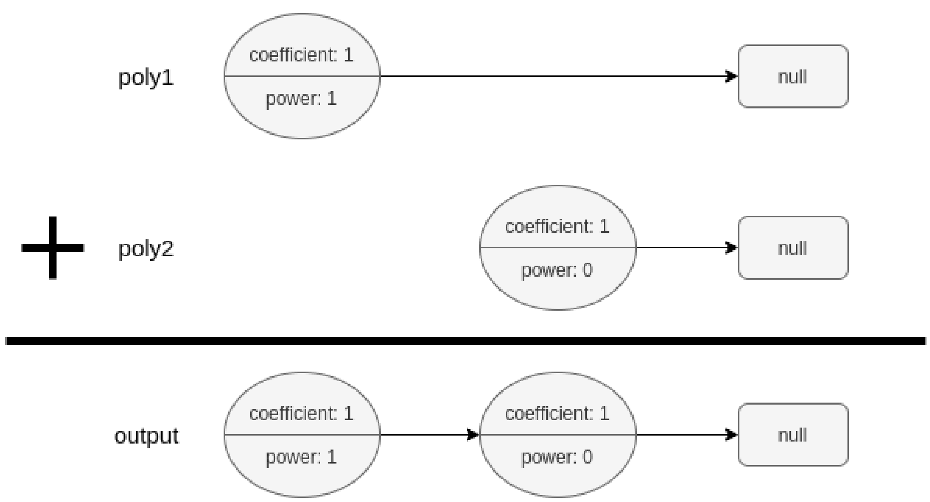
+ 4x - 7

would be represented as:

[[5,3],[4,1],[-7,0]]

.

Example 1:



Input:

poly1 = [[1,1]], poly2 = [[1,0]]

Output:

[[1,1],[1,0]]

Explanation:

poly1 = x. poly2 = 1. The sum is x + 1.

Example 2:

Input:

poly1 = [[2,2],[4,1],[3,0]], poly2 = [[3,2],[-4,1],[-1,0]]

Output:

[[5,2],[2,0]]

Explanation:

poly1 = 2x

2

+ 4x + 3. poly2 = 3x

2

- 4x - 1. The sum is 5x

2

+ 2. Notice that we omit the "0x" term.

Example 3:

Input:

poly1 = [[1,2]], poly2 = [[-1,2]]

Output:

[]

Explanation:

The sum is 0. We return an empty list.

Constraints:

0 <= n <= 10

4

-10

9

<= PolyNode.coefficient <= 10

9

PolyNode.coefficient != 0

0 <= PolyNode.power <= 10

9

PolyNode.power > PolyNode.next.power

Code Snippets

C++:

```
/**
* Definition for polynomial singly-linked list.
* struct PolyNode {
* int coefficient, power;
* PolyNode *next;
* PolyNode(): coefficient(0), power(0), next(nullptr) {};
* PolyNode(int x, int y): coefficient(x), power(y), next(nullptr) {};
* PolyNode(int x, int y, PolyNode* next): coefficient(x), power(y),
next(next) {};
* };
*/

class Solution {
```

```
public:
PolyNode* addPoly(PolyNode* poly1, PolyNode* poly2) {

}
};
```

Java:

```
/**
* Definition for polynomial singly-linked list.
* class PolyNode {
* int coefficient, power;
* PolyNode next = null;

* PolyNode() {}
* PolyNode(int x, int y) { this.coefficient = x; this.power = y; }
* PolyNode(int x, int y, PolyNode next) { this.coefficient = x; this.power =
y; this.next = next; }
* }
*/

class Solution {
public PolyNode addPoly(PolyNode poly1, PolyNode poly2) {

}
}
```

Python3:

```
# Definition for polynomial singly-linked list.
# class PolyNode:
# def __init__(self, x=0, y=0, next=None):
# self.coefficient = x
# self.power = y
# self.next = next

class Solution:
def addPoly(self, poly1: 'PolyNode', poly2: 'PolyNode') -> 'PolyNode':
```

Python:

```
# Definition for polynomial singly-linked list.
# class PolyNode:
# def __init__(self, x=0, y=0, next=None):
# self.coefficient = x
# self.power = y
# self.next = next

class Solution:
def addPoly(self, poly1, poly2):
"""
:type poly1: PolyNode
:type poly2: PolyNode
:rtype: PolyNode
"""
```

JavaScript:

```
/**
* Definition for polynomial singly-linked list.
* function PolyNode(x=0, y=0, next=null) {
* this.coefficient = x;
* this.power = y;
* this.next = next;
* }
*/

/**
* @param {PolyNode} poly1
* @param {PolyNode} poly2
* @return {PolyNode}
*/
var addPoly = function(poly1, poly2) {

};
```

C#:

```
/**
* Definition for polynomial singly-linked list.
* public class PolyNode {
* public int coefficient, power;
* public PolyNode next;
```

```
*
* public PolyNode(int x=0, int y=0, PolyNode next=null) {
* this.coefficient = x;
* this.power = y;
* this.next = next;
* }
* }
*/

public class Solution {
public PolyNode AddPoly(PolyNode poly1, PolyNode poly2) {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Add Two Polynomials Represented as Linked Lists
* Difficulty: Medium
* Tags: array, math, 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 polynomial singly-linked list.
* struct PolyNode {
* int coefficient, power;
* PolyNode *next;
* PolyNode(): coefficient(0), power(0), next(nullptr) {};
* PolyNode(int x, int y): coefficient(x), power(y), next(nullptr) {};
* PolyNode(int x, int y, PolyNode* next): coefficient(x), power(y),
next(next) {};
* };
*/
```

```
class Solution {
public:
PolyNode* addPoly(PolyNode* poly1, PolyNode* poly2) {

}
};
```

Java Solution:

```
/**
* Problem: Add Two Polynomials Represented as Linked Lists
* Difficulty: Medium
* Tags: array, math, 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 polynomial singly-linked list.
* class PolyNode {
* int coefficient, power;
* PolyNode next = null;

* PolyNode() {
// TODO: Implement optimized solution
return 0;
}
* PolyNode(int x, int y) { this.coefficient = x; this.power = y; }
* PolyNode(int x, int y, PolyNode next) { this.coefficient = x; this.power =
y; this.next = next; }
* }
*/

class Solution {
public PolyNode addPoly(PolyNode poly1, PolyNode poly2) {

}
}
```

Python3 Solution:

```
"""
Problem: Add Two Polynomials Represented as Linked Lists
Difficulty: Medium
Tags: array, math, 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 polynomial singly-linked list.
# class PolyNode:
# def __init__(self, x=0, y=0, next=None):
# self.coefficient = x
# self.power = y
# self.next = next

class Solution:
def addPoly(self, poly1: 'PolyNode', poly2: 'PolyNode') -> 'PolyNode':
# TODO: Implement optimized solution
pass
```

Python Solution:

```
# Definition for polynomial singly-linked list.
# class PolyNode:
# def __init__(self, x=0, y=0, next=None):
# self.coefficient = x
# self.power = y
# self.next = next

class Solution:
def addPoly(self, poly1, poly2):
"""
:type poly1: PolyNode
:type poly2: PolyNode
:rtype: PolyNode
"""
```

JavaScript Solution:

```
/**
* Problem: Add Two Polynomials Represented as Linked Lists
* Difficulty: Medium
* Tags: array, math, 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 polynomial singly-linked list.
* function PolyNode(x=0, y=0, next=null) {
* this.coefficient = x;
* this.power = y;
* this.next = next;
* }
*/

/**
* @param {PolyNode} poly1
* @param {PolyNode} poly2
* @return {PolyNode}
*/
var addPoly = function(poly1, poly2) {

};
```

C# Solution:

```
/*
* Problem: Add Two Polynomials Represented as Linked Lists
* Difficulty: Medium
* Tags: array, math, 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 polynomial singly-linked list.
* public class PolyNode {
* public int coefficient, power;
* public PolyNode next;
*
* public PolyNode(int x=0, int y=0, PolyNode next=null) {
* this.coefficient = x;
* this.power = y;
* this.next = next;
* }
* }
*/

public class Solution {
public PolyNode AddPoly(PolyNode poly1, PolyNode poly2) {

}
}
```