Data Structure

ENCS205

School of Engineering & Technology (SOET)
K.R. Mangalam University

UNIT-2
Session 21: Linked List Application: Polynomial Addition

Sessions 21 Outlook

- Linked List Applications
- Polynomial Manipulation
- Polynomial Manipulation Example
- Code to add two polynomials
- Practice Questions
- Conclusion

Objective

- Understand the representation of a polynomial using a singly linked list.
- Demonstrate the algorithm for adding two polynomials represented as linked lists.
- Analyze the time and space complexity of the polynomial addition algorithm.

Learning Outcomes

- **Implement** the polynomial addition algorithm to generate a new linked list representing the sum.
- **Evaluate** the efficiency of using linked lists for polynomial operations compared to static arrays.

Introduction

Linked list is used in a wide variety of applications such as

- Polynomial Manipulation representation
- Addition of long positive integers
- Representation of sparse matrices
- Addition of long positive integers
- Symbol table creation
- Mailing list
- Memory management
- Linked allocation of files
- Multiple precision arithmetic

https://www.tpointtech.com/application-of-linked-list

Polynomial Manipulation

- A polynomial is an expression of the form

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0.$$

- To represent this with a linked list, each term of the polynomial (e.g., $a_n x^n$) is stored in a single node.
- Each node in our linked list will have two main components:
  - **Coefficient:** The numerical value of the term (a_n).
  - **Exponent:** The power of the variable (n).

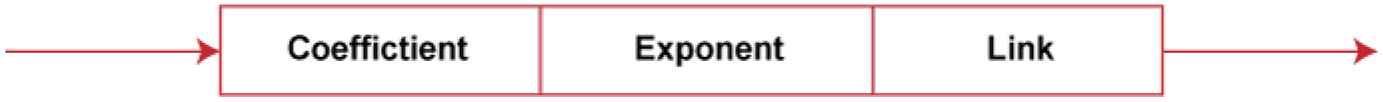
https://www.tpointtech.com/application-of-linked-list

Polynomial Manipulation

Each node of a linked list representing polynomial constitute three parts:

1. The first part contains the value of the coefficient of the term.
2. The second part contains the value of the exponent.
3. The third part, LINK points to the next term (next node).

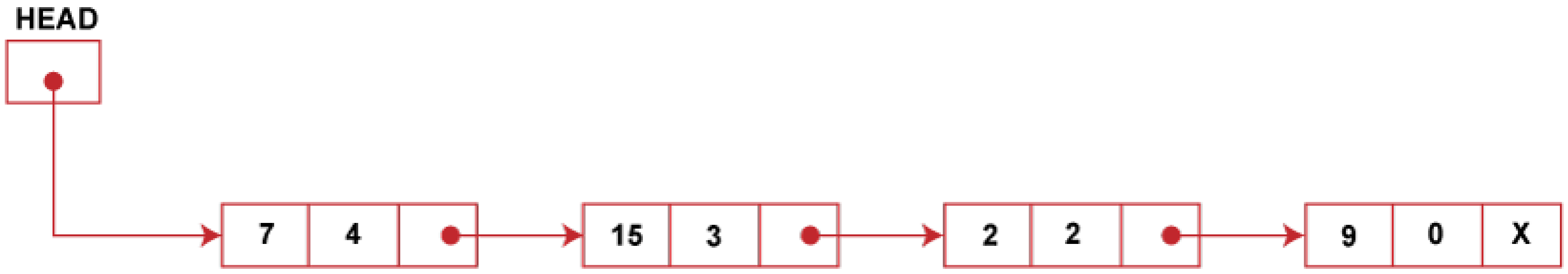
- The structure of a single node would look like this:



Node representing a term of a polynomial

Polynomial Manipulation Example

Consider a polynomial P(x) = $7x^2 + 15x^3 - 2\,x^2 + 9$. Here 7, 15, -2, and 9 are the coefficients, and 4,3,2,0 are the exponents of the terms in the polynomial. On representing this polynomial using a linked list, we have



Linked representation of polynomial P(x)

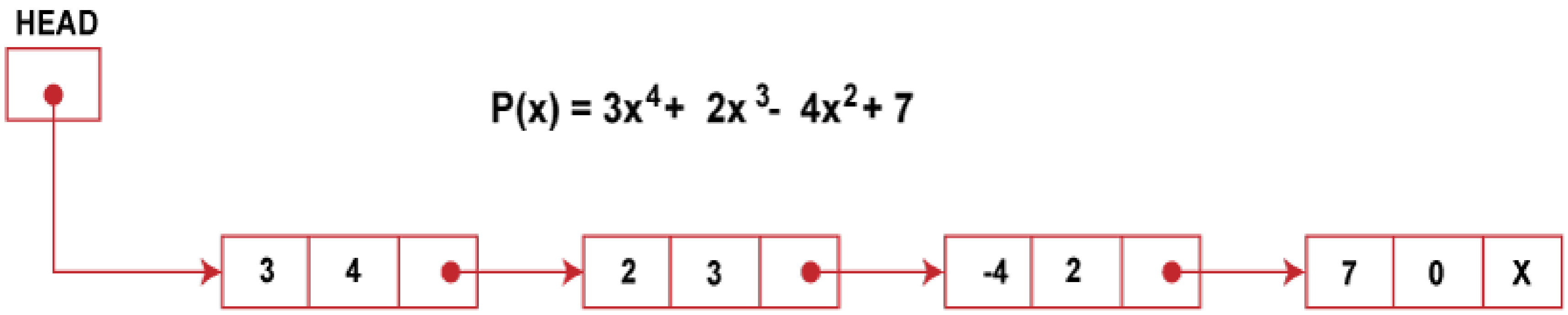
Steps for Polynomial Addition (Using Linked Lists)

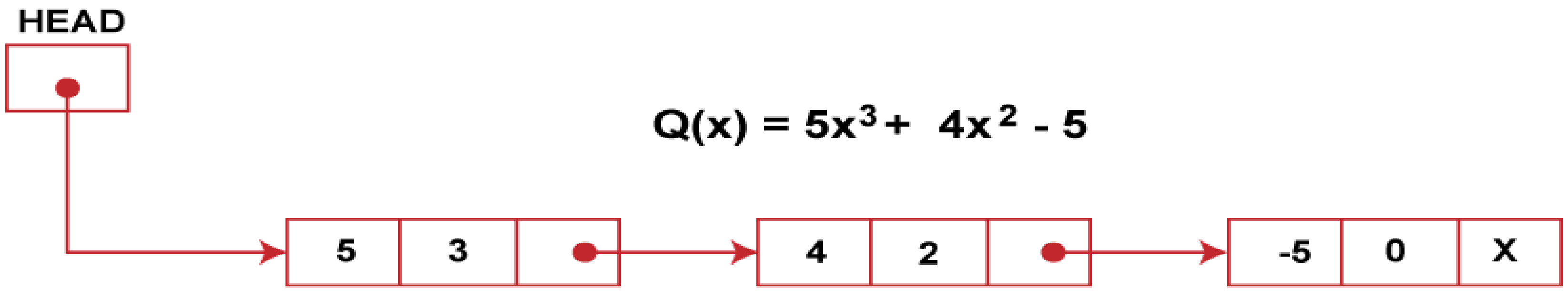
1. **Initialize** a new linked list for the result.
2. **Traverse** both polynomials P and Q simultaneously.
3. **Compare exponents** of current terms from both lists:
4. If **equal**: add coefficients, insert the term in the result if the sum ≠ 0.
5. If one exponent is **greater**, insert that term into the result.
6. **Advance pointers** based on which term was added or processed.
7. **Append remaining terms** from the longer list to the result.
8. **Result list** now contains the sum of polynomials in sorted order by exponent.

Example for Polynomial Addition

$P(x) = 3x^4 + 2x^3 - 4\ x^2 + 7$

$Q\ (x) = 5x^3 + 4\ x^2 - 5$



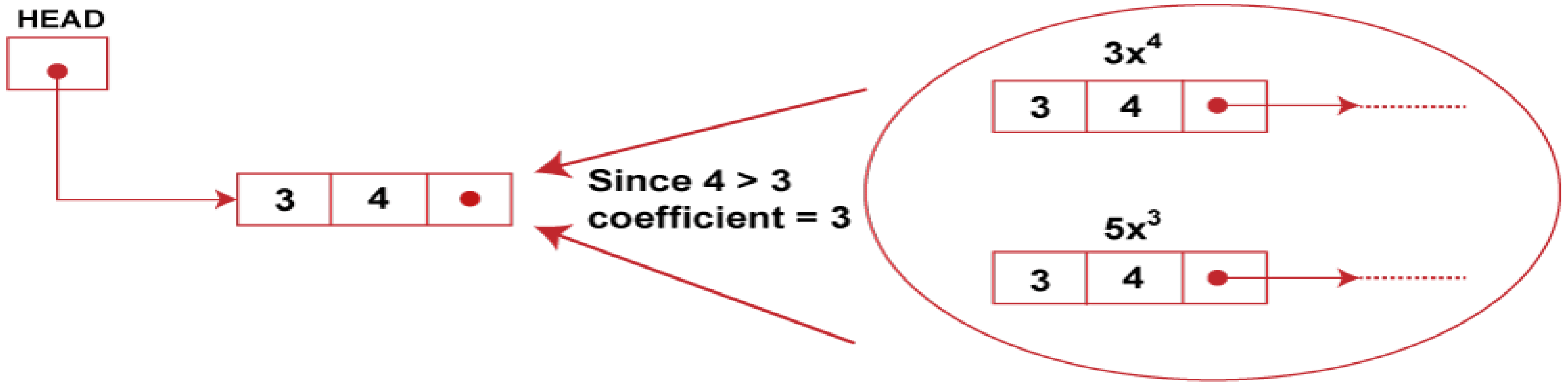


Example for Polynomial Addition

$P(x) = 3x^4 + 2x^3 - 4x^2 + 7$

$Q(x) = 5x^3 + 4x^2 - 5$

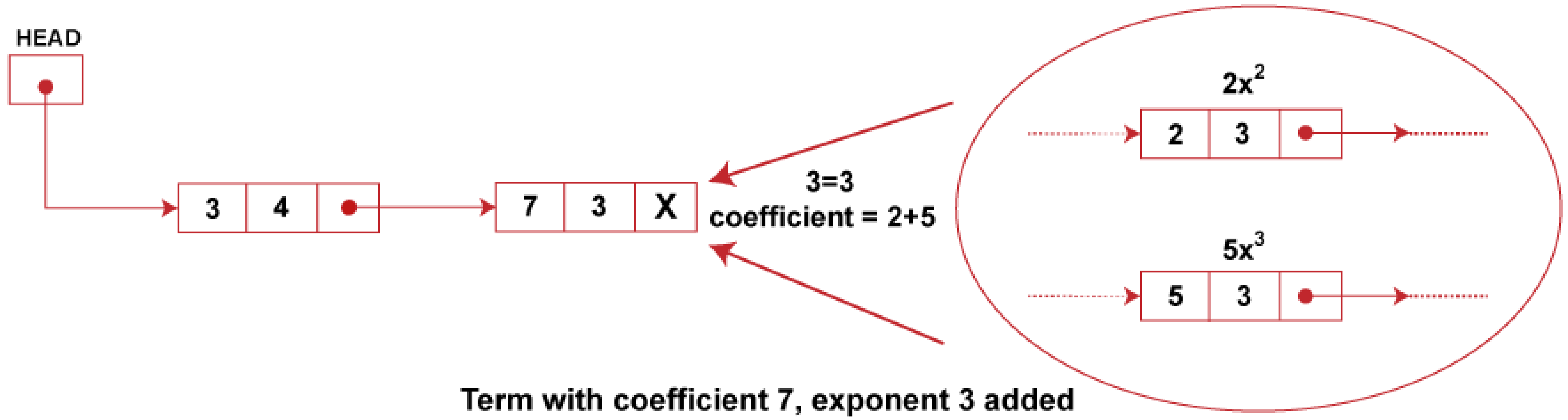
1. Traverse the two lists P and Q and examine all the nodes.
2. We compare the exponents of the corresponding terms of two polynomials



Term with coefficient 3, exponent 4 added

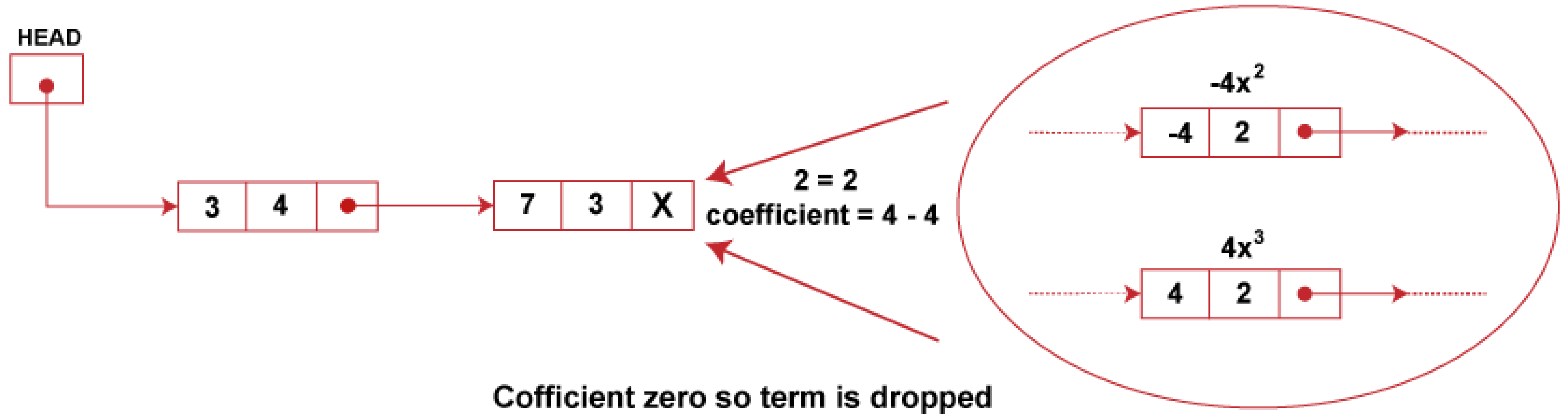
Example for Polynomial Addition

3. Compare the exponent of the next term of the list P with the exponents of the present term of list Q. Since the two exponents are equal, so their coefficients are added and appended to the new list as follows:



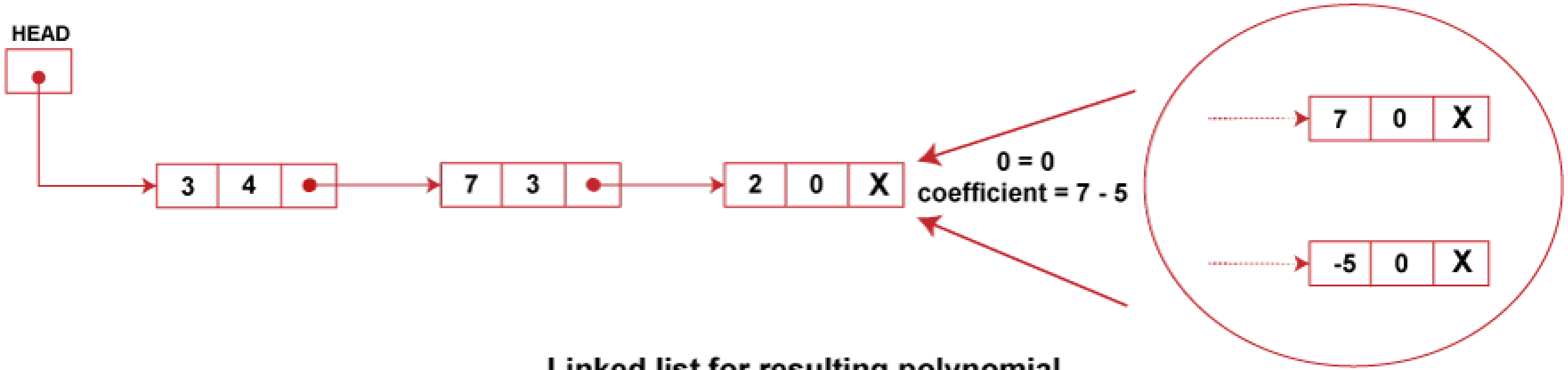
Example for Polynomial Addition

4. Then we move to the next term of P and Q lists and compare their exponents. Since exponents of both these terms are equal and after addition of their coefficients, we get 0, so the term is dropped, and no node is appended to the new list after this



Example for Polynomial Addition

5. Moving to the next term of the two lists, P and Q, we find that the corresponding terms have the same exponents equal to 0. We add their coefficients and append them to the new list for the resulting polynomial as shown below:



Linked list for resulting polynomial

Code to add two polynomials

```
Function AddPolynomials(P, Q):
    Initialize result ← empty linked list

    While P is not NULL and Q is not NULL:
        If P.exponent == Q.exponent:
            sum ← P.coefficient + Q.coefficient
            If sum ≠ 0:
                Insert (sum, P.exponent) into result
            Move P and Q to next nodes

        Else if P.exponent > Q.exponent:
            Insert (P.coefficient, P.exponent) into result
            Move P to next node
```

Code to add two polynomials

```
    Else:
        Insert (Q.coefficient, Q.exponent) into result
        Move Q to next node

While P is not NULL:
    Insert (P.coefficient, P.exponent) into result
    Move P to next node

While Q is not NULL:
    Insert (Q.coefficient, Q.exponent) into result
    Move Q to next node

Return result
```

Complete code here.

Practice Questions

1. Consider the polynomial $P1=7x^4+3x^2-1$ and $P2=5x^4-2x^2+6x$.

Find sum P1+P2

2. Given two polynomials:$P(x) = 4x^4 + 3x^3 + xQ(x) = 5x^4 - 3x^3 + 6$

Perform polynomial addition manually and draw the linked list structure for the result.

3. Suppose two polynomials have many terms with the same exponent. How can you modify your implementation to combine all terms with equal exponents?

Conclusion

- Using a linked list to represent and add polynomials is a classic example of how dynamic data structures can provide a flexible and efficient solution to a real-world problem.
- The ability to handle polynomials of varying sizes and the linear time complexity of the addition algorithm make this a superior approach compared to using static arrays, especially for sparse polynomials.
- This application highlights the power and versatility of linked lists in algorithm design.

THANK
YOU