

Work Three

劉向榮

2025 1 7

Homework 3

[*Programming Project*] Develop a C++ class *Polynomial* to represent and manipulate univariate polynomials with integer coefficients (use circular linked lists with header nodes). Each term of the polynomial will be represented as a node. Thus, a node in this system will have three data members as below:

coef	exp	link
------	-----	------

Each polynomial is to be represented as a circular list with header node. To delete polynomials efficiently, we need to use an available-space list and associated functions as described in Section 4.5. The external (i.e., for input or output) representation of a univariate polynomial will be assumed to be a sequence of integers of the form: $n, c_1, e_1, c_2, e_2, c_3, e_3, \dots, c_n, e_n$, where e_i represents an exponent and c_i a coefficient; n gives the number of terms in the polynomial. The exponents are in decreasing order— $e_1 > e_2 > \dots > e_n$.

Write and test the following functions:

- istream*& **operator**>>(*istream*& *is*, *Polynomial*& *x*): Read in an input polynomial and convert it to its circular list representation using a header node.
- ostream*& **operator**<<(*ostream*& *os*, *Polynomial*& *x*): Convert *x* from its linked list representation to its external representation and output it.
- Polynomial*::*Polynomial*(**const** *Polynomial*& *a*) [Copy Constructor]: Initialize the polynomial ***this** to the polynomial *a*.
- const** *Polynomial*& *Polynomial*::**operator**=(**const** *Polynomial*& *a*) **const** [Assignment Operator]: Assign polynomial *a* to ***this**.
- Polynomial*::~*Polynomial*() [Destructor]: Return all nodes of the polynomial ***this** to the available-space list.
- Polynomial* **operator**+(**const** *Polynomial*& *b*) **const** [Addition]: Create and return the polynomial ***this** + *b*.
- Polynomial* **operator**-(**const** *Polynomial*& *b*) **const** [Subtraction]: Create and return the polynomial ***this** - *b*.
- Polynomial* **operator***(**const** *Polynomial*& *b*) **const** [Multiplication]: Create and return the polynomial ***this** * *b*.
- float** *Polynomial*::*Evaluate*(**float** *x*) **const**: Evaluate the polynomial ***this** at *x* and return the result.

效能分析

空間複雜度分析

1. 鏈表節點的存儲：

- 每個多項式用一個環形鏈表來存儲，每個節點包含一個系數、一個指數和一個指向下一節點的指針。
- 若多項式包含 nnn 個非零項，則需要 $O(n)O(n)O(n)$ 的空間來存儲節點。
- 假設有兩個多項式 $p_1p_1p_1$ 和 $p_2p_2p_2$ ，則總空間需求為 $O(n+m)O(n+m)O(n+m)$ ，其中 nnn 和 mmm 是 $p_1p_1p_1$ 和 $p_2p_2p_2$ 的項數。

2. 結果多項式的存儲：

- 加法和減法：最多會生成包含 $n+mn+mn+m$ 項的結果多項式，空間需求為 $O(n+m)O(n+m)O(n+m)$ 。
- 乘法：最壞情況下，結果多項式的項數可能高達 $n \cdot m \cdot mn \cdot m$ （例如兩個完全展開的多項式相乘），因此空間複雜度為 $O(n \cdot m)O(n \cdot m)O(n \cdot m)$ 。

3. 輔助變數：

- 只使用了一些指針（如 Node^* ），以及一些臨時變數來遍歷鏈表，這些都是常數級別的空間需求 $O(1)O(1)O(1)$ 。

總空間複雜度：

- 加法/減法： $O(n+m)O(n+m)O(n+m)$
- 乘法： $O(n \cdot m)O(n \cdot m)O(n \cdot m)$

時間複雜度分析

1. 輸入與輸出（ $\text{operator}>>$ 和 $\text{operator}<<$ ）：

- 需要遍歷所有節點，輸入和輸出的時間複雜度均為 $O(n)O(n)O(n)$ 和 $O(m)O(m)O(m)$ 。

2. 插入節點 (Insert) :

- 插入新項時，必須遍歷節點以找到正確的位置。對於 nnn 項的多項式，單次插入的時間複雜度為 $O(k)O(k)O(k)$ ，其中 kkk 是節點數。
- 若插入 nnn 次（整個多項式輸入），總時間為 $O(n^2)O(n^2)O(n^2)$ （假設每次都在末尾插入）。

3. 加法與減法 (operator+ 和 operator-) :

- 遍歷兩個多項式，合併節點時僅需一次遍歷。
- 時間複雜度為 $O(n+m)O(n+m)O(n+m)$ ，其中 nnn 和 mmm 是兩個多項式的項數。

4. 乘法 (operator*) :

- 每個項 aia_iai 與另一多項式 bbb 的所有項進行相乘，總共進行 $n \cdot m \cdot k$ 次乘法操作。
- 插入結果時可能需要處理項的合併（在最壞情況下可能花費 $O(k)O(k)O(k)$ ，其中 kkk 是當前結果多項式的項數）。
- 最壞情況下的時間複雜度為 $O(n \cdot m \cdot k)O(n \cdot m \cdot k)O(n \cdot m \cdot k)$ ，但通常 $k \leq n \cdot m$ 。

5. 評估 (Evaluate) :

- 逐項計算 $c_i \cdot x^{e_i}$ ，對 nnn 項的多項式來說，時間複雜度為 $O(n)O(n)O(n)$ 。

總時間複雜度：

- 輸入/輸出： $O(n)O(n)O(n)$
- 插入： $O(n^2)O(n^2)O(n^2)$ （對於多項式建構）
- 加法/減法： $O(n+m)O(n+m)O(n+m)$
- 乘法： $O(n \cdot m \cdot k)O(n \cdot m \cdot k)O(n \cdot m \cdot k)$ （最壞情況）

測試與驗證

舉例測資：

假設用戶輸入以下多項式：

- Polynomial 1: $2x^3 + 3x^2 - x + 5$
- Polynomial 2: $x^2 - 4x + 6$

其實際運算結果為

```
Enter polynomial 1 (format: n c1 e1 c2 e2 ...): 4 2 3 3 2 -1 1 5 0
Enter polynomial 2 (format: n c1 e1 c2 e2 ...): 3 1 2 -4 1 6 0
Polynomial 1:  $2x^3 + 3x^2 - 1x + 5$ 
Polynomial 2:  $1x^2 - 4x + 6$ 
Sum:  $2x^3 + 4x^2 - 5x + 11$ 
Difference:  $2x^3 + 2x^2 + 3x - 1$ 
Product:  $2x^5 - 5x^4 - 1x^3 + 27x^2 - 26x + 30$ 
Enter a value to evaluate polynomial 1:
```

申論及心得

實作這個多項式類讓我深入理解了鏈表的靈活性與挑戰，並體驗到運算符重載提升程式可讀性的好處。此外，過程中學到了如何平衡時間與空間效率，並加強了對 C++ 面向對象設計和內存管理的掌握。