# 資料結構報告

張哲維

July 30, 2024

CONTENTS

# CHAPTER 1

## 解題說明

用Polynomial來表示多項式，進行加法、乘法　處理多項式的輸入 輸出使用運算符號來方便這些操作。

## Ex:

兩個多項式 p1(x) = 4x^2 + 3x + 1 和 p2(x) = x^2 + 2，我們希望實現以下功能：

1. **加法**: p1 + p2 = (4+1)x^2 + 3x + (1+2) = 5x^2 + 3x + 3

2. **乘法**: p1 * p2 = (4x^2 + 3x + 1)( x^2 + 2) = 8x + x^2 + 4 = 3x^4 + 2x^3 + 13x^2 + 8x + 4

1

CHAPTER2

## 解題說明

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

class Poly; // 前向宣告

class Term {
    friend class Poly;
public:
    float coef; // 係數
    int exp;    // 指數
};
int main() {
    Poly p1, p2;

    cout << "輸入第一個多項式:\n";
    cin >> p1;
    cout << "輸入第二個多項式:\n";
    cin >> p2;

    cout << "多項式1: " << p1 << endl;
    cout << "多項式2: " << p2 << endl;

    Poly p3 = p1 + p2;

    cout << "(" << p1 << ") + (" << p2 << ") = " << p3 << endl;

    return 0;
}
```

```
void Poly::addTerm(float c, int e) {
    for (int i = 0; i < size; ++i) {
        if (terms[i].exp == e) {
            terms[i].coef += c;
            return;
        }
    }
    if (size == cap) {
        cap *= 2;
        Term* newTerms = new Term[cap];
        copy(terms, terms + size, newTerms);
        delete[] terms;
        terms = newTerms;
    }
    terms[size].coef = c;
    terms[size++].exp = e;
}

Poly Poly::operator+(const Poly& other) const {
    Poly res;
    int a = 0, b = 0;
    while (a < size && b < other.size) {
        if (terms[a].exp == other.terms[b].exp) {
            float t = terms[a].coef + other.terms[b].coef;
            if (t) res.addTerm(t, terms[a].exp);
            a++;
            b++;
        } else if (terms[a].exp < other.terms[b].exp) {
            res.addTerm(other.terms[b].coef, other.terms[b].exp);
            b++;
        } else {
```

```
Polynomial Polynomial::operator*(const Polynomial& other) const {
    Polynomial c;
    for (int i = 0; i < terms; ++i) {
        for (int j = 0; j < other.terms; ++j) {
            float newCoef = termArray[i].coef * other.termArray[j].coef;
            int newExp = termArray[i].exp + other.termArray[j].exp;
            c.NewTerm(newCoef, newExp);
        }
    }
```

```
float Polynomial::evaluate(float x) const {
    float result = 0.0;
    for (int i = 0; i < terms; ++i) {
        result += termArray[i].coef * pow(x, termArray[i].exp);
    }
    return result;
}
```

CHAPTER3

# 效能分析

## 時間複雜度

1. **AddTerm**: 時間複雜度為 O(n)， 因為可能需要遍歷所有項。
2. **Add**: 時間複雜度為O(n+m)， n 和 m 分別是兩個多項式的項數。
3. **Mult**: 時間複雜度為 O(n*m)， n 和 m 分別是兩個多項式的項數。
4. **Eval**: 時間複雜度為bigO(m) 。m是terms的長度

空間複雜度

1. **AddTerm: O(1)， 不需要額外的空間。**
2. **Add: O(n+m)， 需要儲存結果多項式。**
3. **Mult: O(n*m),**
4. **Eval: O(1)**

# CHAPTER 4

## 驗證與過程

驗證加法 (2x^2) + (1x^2 + 2x^1)是否等於3x^2 + 2x^1

乘法 (2x^2) * (1x^2 + 2x^1)　是否等於2x^4 + 4x^3

X 帶入2 的值是否正確

3x^2 + 2x^1帶入2是否是16

2x^4 + 4x^3帶入2是否是64

```
輸入項數: 1
輸入係數和指數: 2 2
輸入第二個多項式:
輸入項數: 2
輸入係數和指數: 2 1
輸入係數和指數: 1 2
多項式1: 2x^2
多項式2: 1x^2 + 2x^1
(2x^2) + (1x^2 + 2x^1) = 3x^2 + 2x^1
(2x^2) * (1x^2 + 2x^1) = 2x^4 + 4x^3
輸入 x 的值來計算多項式的值: 2
P1 在 x = 2 時的值: 8
P2 在 x = 2 時的值: 8
P1 + P2 在 x = 2 時的值: 16
P1 * P2 在 x = 2 時的值: 64
```

# 心得

因為這是暑修所以之前就又學過所以沒這麼生疏 還有一些可以找以前的程式 現在科技發達有GPT可以輔助學習 非常方便 在編譯上有任何錯誤都可以詢問但不一定能解決 所以書籍也是很重要的資料