**COEFFICIENT ARRAYS**

Typical polynomials look like:

\[ x^5 + 10x^2 + 7x - 15 \]
\[ 2x^2 + 3x + 5 \]
\[ 10x^5 + 5x^3 + 3x^2 + 20x \]

The highest power of the polynomial is its **degree**.

A polynomial can be stored in memory using a *coefficient array* – an array \( \text{coef} \) in which \( \text{coef}[k] \) contains the coefficient of \( x^k \).

**Example**
The polynomial \( x^5 + 10x^2 + 7x - 15 \) has degree of 5. It is stored as:

| coef | \(-15\) | 7 | 10 | 0 | 0 | 1 |

**Polynomial Evaluation**
A polynomial \( P(x) \) is evaluated by substituting a value for \( x \) and performing the requisite computations.

**Example**
Given \( P(x) = 2x^3 - 4x^2 + 6x - 10 \)

\[ P(5) = 2 \cdot 5^3 - 4 \cdot 5^2 + 6 \cdot 5 - 10 = 170 \]

When done by calculating the powers, the number of multiplication operations needed to evaluate \( P(x) \) is at most \( \sum_{k=1}^{d} k = 1 + 2 + \ldots + d = \frac{d(d+1)}{2} \), where \( d \) is the degree of \( P(x) \).

**Examples**
\( P(x) = 2x^3 - 4x^2 + 6x - 10 \) has degree of 3. Its evaluation requires \( \leq \frac{3(4)}{2} = 6 \) multiplications.

\( Q(x) = x^5 + 10x^2 + 7x - 15 \) has degree of 3. Its evaluation requires \( \leq \frac{5(6)}{2} = 15 \) multiplications.
**Horner’s Method**

William George Horner (1786–1837) invented a much faster way to evaluate a polynomial, which is called *Horner’s method* in his honor. It works by factoring the polynomial. This polynomial of degree $n$:

$$a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \ldots + a_2 x^2 + a_1 x + a_0$$

Appears as follows after factoring $x$ out of the first $n$ terms:

$$(a_n x^{n-1} + a_{n-1} x^{n-2} + a_{n-2} x^{n-3} + \ldots + a_2 x + a_1) x + a_0$$

Factoring another $x$ out the first $n-1$ terms gives:

$$((a_n x^{n-2} + a_{n-1} x^{n-3} + a_{n-2} x^{n-4} + \ldots + a_2) x + a_1) x + a_0$$

And so on until yielding:

$$(((\ldots((a_n x + a_{n-1}) x + a_{n-2}) x + \ldots + a_2) x + a_1) x + a_0$$

**Examples**

$P(x) = 2x^3 - 4x^2 + 6x - 10 = ((2x - 4)x + 6)x - 10$

$Q(x) = x^5 + 10x^2 + 7x - 15 = x^5 + 0x^4 + 0x^3 + 10x^2 + 7x - 15 = (((x + 0)x + 0)x + 10)x + 7)x - 15$

The polynomial is evaluated by setting the initial result to $a_n$, multiplying it by $x$’s value, adding $a_{n-1}$ and repeating the process until reaching $a_0$. In the algorithm below, $p$ is the coefficient array.

```plaintext
Horner’s Method
Given $P(x) = p_n x^n + \ldots + p_0$, evaluate $P(c)$

x = c
val = p_n
for k = n down to 1 do
    val = (val * x) + p_k
end for
```
**Example**

Given $P(x) = 2x^3 - 4x^2 + 6x - 10$, here’s its coefficient array:

And a trace of the evaluation of $P(5)$:

<table>
<thead>
<tr>
<th>Loop Cycle</th>
<th>$x$</th>
<th>$n$</th>
<th>$k$</th>
<th>$(\text{val} \times x) + p[k-1]$</th>
<th>val</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td>$2 \times 5 - 4$</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td>$6 \times 5 + 6$</td>
<td>36</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td>$36 \times 5 - 10$</td>
<td>170</td>
</tr>
</tbody>
</table>

Horner’s method performs faster than the method that calculates the powers. To evaluate $P(x)$ whose degree is $d$, Horner’s method requires $d$ multiplication operations.

**Examples**

$P(x) = 2x^3 - 4x^2 + 6x - 10$ has degree of 3. Horner’s method requires 3 multiplications to evaluate it.

$Q(x) = x^5 + 10x^2 + 7x - 15$ has degree of 3. Horner’s method requires 5 multiplications to evaluate it.

**Polynomial Addition**

The sum of two polynomials is a third polynomial created by adding the coefficients of like terms and merging the remaining terms.

**Example**

\[
\frac{x^3 - 10x^2 + 15}{x^3 - 9x^2 - 3x + 17}
+ \frac{x^2 - 3x + 2}{x^2}
\]
When implemented using coefficient arrays, you can implement polynomial addition simply by adding corresponding entries of the array.

**Example**
The coefficient array for $x^3 - 10x^2 + 15$:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>p</strong></td>
<td>15</td>
<td>0</td>
<td>-10</td>
<td>1</td>
</tr>
</tbody>
</table>

The coefficient array for $x^2 - 3x + 2$:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>q</strong></td>
<td>2</td>
<td>-3</td>
<td>1</td>
</tr>
</tbody>
</table>

Adding corresponding entries of these two arrays yields this array:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>sum</strong></td>
<td>17</td>
<td>-3</td>
<td>-9</td>
<td>1</td>
</tr>
</tbody>
</table>

Which is the coefficient array of $x^3 - 9x^2 - 3x + 17$.

---

**Polynomial Addition**
Given $P(x) = p_nx^n + \ldots + p_0$ and $Q(x) = q_mx^m + \ldots + q_0$, calculate $P(x) + Q(x)$

- create a new coefficient array $c$, which is a copy of the larger of arrays $p$ and $q$
- for $i = 0$ up to the smaller of $n$ and $m$ do
  - $c[i] = p[i] + q[i]$
- end for
Distribution of a Term over a Polynomial

Distribution is the process of multiplying a single term by each term in the polynomial.

Example

\[
2x^2(x^2 - 3x + 2) = 2x^3(x^2) - 2x^3(3x) + 2x^3(2) = 2x^5 - 6x^4 + 4x^3
\]

As can be seen in the example, for each term in the polynomial, you multiply its coefficient by the coefficient of the single term and add its exponent onto the exponent of the single term; e.g. \(2x^3(3x) = 2 \cdot 3(x^3 \cdot x) = 6(x^{3+1}) = 6x^4\). With coefficient arrays, the algorithm becomes:

```
Distribution of a Term over a Polynomial
Given \(ax^k\) and \(p_nx^n + \ldots + p_0\), calculate \(ax^k[p_nx^n + \ldots + p_0]\)
create a new, larger coefficient array \(c\) of size \(k+n\)
for \(i = 0\) to \(k-1\) do
    \(c[i] = 0\)
end for
for \(i = k\) to \(n\) do
    \(c[i] = a \cdot p_{i-k}\)
end for
```

```
Example
Given \(2x^3(x^2 - 3x + 2)\). The coefficient array for \(x^2 - 3x + 2\) is:
The answer is \(2x^5 - 6x^4 + 4x^3\). Here’s a trace of the algorithm:
```

<table>
<thead>
<tr>
<th>Loop Cycle</th>
<th>a</th>
<th>k</th>
<th>n</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>First loop</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Second loop</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
```
Polynomial Multiplication
The product of two polynomials is a third polynomial created by distributing each term of the first polynomial over the second polynomial and then adding the resulting polynomials.

Example

\[
\begin{array}{c}
2x^3 - 10x^2 + 15 \\
x^2 - 3x + 2
\end{array}
\times
\begin{array}{c}
2x^3(x^2 - 3x + 2) \\
+ (-10x^2(x^2 - 3x + 2)) \\
+ (15(x^2 - 3x + 2))
\end{array}
= 2x^5 - 6x^4 + 4x^3 \\
+ (-10x^4 + 30x^3 - 20x^2) \\
+ (15x^2 - 45x + 30)
\]

\[
= 2x^5 - 16x^4 + 34x^3 - 5x^2 - 45x + 30
\]

As you can see from the example, polynomial multiplication makes use of the previous two algorithms.

Polynomial Multiplication

Given \( P(x) = p_nx^n + \ldots + p_0 \) and \( Q(x) = q_mx^m + \ldots + q_0 \), calculate \( P(x) \times Q(x) \)

create an empty array \( c \) that can hold \( n+1 \) polynomials

for \( k = 0 \) up to \( n \) do

calculate the polynomial \( p_kx^k[q_mx^m + \ldots + q_0] \) using the distribution algorithm and store the result into \( c_k \)

end for

create a new coefficient array \( d = c_0 \)

for \( k = 1 \) up to \( n \) do

add \( c_k \) onto \( d \) using the polynomial addition algorithm

end for
Exercises

1. Implement and test the following Java method that displays a polynomial stored as a coefficient array.

```
public void display( double [] p )
    // Display the polynomial 'p'.
```

To test your method, write a test wrapper that initializes a coefficient array and passes it to the method. For example:

```
double v = { -3, 2, -5, 0, 1 };
display( v );
```

Should display:

```
x^4 - 5x^2 + 2x - 3
```

Observe that your method must:

- Suppress the printing of 1 as a coefficient or exponent
- Suppress the printing of terms equaling 0
- Print the coefficient signs, either positive or negative

2. Implement Horner’s Method as the following Java method.

```
public double eval( double [] p, double x )
    // Given polynomial P(x) store in the coefficient array 'p' and the value 'x', evaluate P(x).
    // Return the calculated value.
```

Test your method in a manner similar to that of problem 1.
3. Implement polynomial addition as the following Java method.

   ```java
   public double[] add(double[] p, double[] q)
   // Given polynomials P(x) and Q(x) stored in the
   // coefficient arrays 'p' and 'q', calculate P(x)+Q(x).
   // Return the coefficient array of the result.
   ```

   Test your method in a manner similar to that of problem 1.

4. Implement distribution of a term over a polynomial as the following Java method.

   ```java
   public double[] dist(double a, int k, double[] p)
   // Given term ax^k and polynomial P(x), calculate ax^k[P(x)].
   // The term is stored in arguments 'a' and 'k'.
   // P is stored in the coefficient array 'p'.
   // Return the coefficient array of the result.
   ```

   Test your method in a manner similar to that of problem 1.

5. Implement polynomial multiplication as the following Java method.

   ```java
   public double[] times(double[] p, double[] q)
   // Given polynomials P(x) and Q(x) stored in the
   // coefficient arrays 'p' and 'q', calculate P(x)*Q(x).
   // Return the coefficient array of the result.
   ```

   Test your method in a manner similar to that of problem 1.
6. Implement and test the following Java method that builds a polynomial from user input and stores it into a coefficient array.

   ```java
double [] input( double [] p )
    // Build the polynomial p from user input.
```

For example, user entry of the polynomial $x^5 + 10x^2 - 7x + 15$ could go something like this (user input is underlined):

   Enter the polynomial's degree: 5

   Enter each term, coefficient first.
   You need not enter zero terms.
   You may enter terms in any order,
   EXCEPT you MUST enter the DEGREE LAST.
   For $cx^e$, enter c and e: 15 0
   For $cx^e$, enter c and e: 10 2
   For $cx^e$, enter c and e: -7 1
   For $cx^e$, enter c and e: 1 5