Algorithms and Data Structures

Examples

Edgar Pek

10.10.2008.

Definition (Polynomial)

Polynomial is an expression constructed from one or more variables and constants, using only the operations of addition, subtraction, multiplication, and constant positive whole number exponents.

Example

$P(x) = x^3 + 3 \cdot x - 16$

Definition (Univariate polynomial)

$$P(x) = \sum_{k=0}^{n} a_k x^k = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

- Naive polynomial evaluation algorithm

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

Example

Evaluate $P(x) = 2x^3 - 3x^2 + 5x - 7$ for $x = 3$.

Example

Evaluate $P(x) = 2x^3 - 3x^2 + 5x - 7$ for $x = 3$.

$$\begin{aligned} P(x) &= 2 \cdot 3^3 - 3 \cdot 3^2 + 5 \cdot 3 - 7 \\ &= 3^3 + 8 \\ &= 35 \end{aligned}$$

Naive algorithm

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

1. Pseudo-code

Naive algorithm

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

1. Pseudo-code

   NAIVE-POLY-EVAL(A, x)

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

1. Pseudo-code

NAIVE-POLY-EVAL(A, x)

1 $y \leftarrow 0$

Naive algorithm

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

1. Pseudo-code

   NAIVE-POLY-EVAL(A, x)

   $y \leftarrow 0$

   for $k \leftarrow 1$ **to** $length(A)$

Naive algorithm

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

1. Pseudo-code

   NAIVE-POLY-EVAL(A, x)

   $y \leftarrow 0$
   for $k \leftarrow 1$ **to** $length(A)$
       do $p \leftarrow 1$

Naive algorithm

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

1. Pseudo-code

   NAIVE-POLY-EVAL(A, x)

   ```
   y ← 0
   for k ← 1 to length(A)
        do p ← 1
           for j ← 1 to k
               do p ← p · x
           y ← y + a_k · p
   ```

2. Asymptotic running time

Naive algorithm

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

1. Pseudo-code

   NAIVE-POLY-EVAL(A, x)

   ```
   y ← 0
   for k ← 1 to length(A)
         do p ← 1
            for j ← 1 to k
                do p ← p · x
            y ← y + a_k · p
   ```

2. Asymptotic running time $T(n) = \Theta(n^2)$

- Java
- Python

- William George Horner - 1819
- Isaac Newton - 1669
- ...
- *The Nine Chapters of the Mathematical Art*
    - Han Dynasty (202 BC - 220 AD)

$$P(x) = a_0 + x(a_1 + x(a_2 + \ldots + x(a_{n-1} + xa_n)\ldots)$$

We evaluate polynomial at a specific value of x, e.g. x_0 as follows:

$$\begin{aligned} y_0 &= a_n \\ y_1 &= a_{n-1} + y_0 x_0 \\ y_2 &= a_{n-2} + y_1 x_0 \\ &\vdots \\ y_n &= a_0 + y_{n-1} x_0 \end{aligned}$$

At the end, y_n is the value of $P(x_0)$.

Example

Evaluate $P(x) = 2x^3 - 3x^2 + 5x - 7$ for $x = 3$.

Example

Evaluate $P(x) = 2x^3 - 3x^2 + 5x - 7$ for $x = 3$.

$$P(x) = -7 + x(5 + x(-3 + 2x))$$

Example

Evaluate $P(x) = 2x^3 - 3x^2 + 5x - 7$ for $x = 3$.

$$
\begin{aligned}
P(x) &= -7 + x(5 + x(-3 + 2x)) \\
P(3) &= -7 + x(5 + 3x) \\
&= -7 + 14x \\
&= 35
\end{aligned}
$$

1. Pseudo-code

Horner's rule

Algorithm

1. Pseudo-code

   HORNER-POLY-EVAL(A)

   ```
   y ← 0
   k ← n
   while k ≥ 0
         do y ← a_k + x · y
            k ← k − 1
   ```

2. Asymptotic running time

Horner's rule

Algorithm

1. Pseudocode

   HORNER-POLY-EVAL(A)

   $y \leftarrow 0$
   $k \leftarrow n$
   while $k \geq 0$
       do $y \leftarrow a_k + x \cdot y$
           $k \leftarrow k - 1$

2. Asymptotic running time $T(n) = \Theta(n)$

- Java
- Python