Algorithms and Complexity, simple examples $\boxed{\text{algor}}$

Problem: evaluate a polynomial p at a point x.

compute $y = p(x)$

Representations of a poly

Ex $p(x) = \frac{15}{4} - \frac{9}{4}x + \frac{1}{2}x^2$

$$= a_1 + a_2 x + a_3 x^2$$

expansion in "monomial basis"

Generally $p(x) = a_1 + a_2 x + \cdots + a_{n+1} x^n$

Ex 1 $\quad p(x) = 2 - \frac{1}{4}(x-1) + \frac{1}{2}(x-1)(x-3)$

Shifted form

$$= c_1 + c_2(x-r_1) + c_3(x-r_1)(x-r_2)$$

Generally $\quad p(x) = c_1 + c_2(x-r_1) +$

$$\cdots + c_{n+1}\underbrace{(x-r_1)(x-r_2)\cdots(x-r_n)}_{\text{product of } n \text{ factors}}$$

> Fact: same poly in each example. a degree-n poly p can be expressed relative to any set of base points $r_1, r_2, \ldots, r_n$

Let's evaluate

$$p(x) = 3.99x^4 + 7.34x^3 - 2.03x^2 + 9.1x - 5.277$$

Naive way

$$\left.\begin{aligned} p(x) = {} & 3.99 * x * x * x * x \\ & + 7.34 * x * x * x \\ & - 2.03 * x * x \\ & + 9.1 * x \\ & - 5.277 \end{aligned}\right\} \quad \begin{array}{l} \text{10 multiplications} \\ \text{4 additions} \\ \boxed{\text{14 flops}} \end{array}$$

Easy to do a bit better w/ recursion

$x2 = x * x$

$x3 = x * x2$

$x4 = x * x3$

$p(x) = 3.99 * x4 + 7.34 * x3 - 2.03 * x2 + 9.1 * x - 5.277$

7 multiplications, 4 additions

$\boxed{\text{11 flops}}$

Even better (use <u>nested</u> form)

$$p(x) = -5.277 + 9.1x - 2.03x^2 + 7.34x^3 + 3.99x^4$$

$$= -5.277 + x(9.1 - 2.03x + 7.34x^2 + 3.99x^3)$$

$$= -5.277 + x(9.1 + x(-2.03 + 7.34x + 3.99x^2))$$

$$= -5.277 + x \underline{*} (9.1 + x \underline{*} (-2.03 + x \underline{*} (7.34 + x \underline{*} 3.99)))$$

$$\left.\begin{array}{l}\text{4 multiplications}\\ \text{4 additions}\end{array}\right\} \boxed{\text{8 flops}}$$

organization of computation affects flop count

Consider

$$p(x) = 2 - \frac{1}{4}(x-1) + \frac{1}{2}(x-1)(x-3)$$

$$= 2 + (x-1)\left(-\frac{1}{4} + (x-3)\frac{1}{2}\right)$$

$$= c_1 + (x-r_1)\left(c_2 + (x-r_2)c_3\right)$$

to compute $y = p(x)$:

$y = c_3$

$y = y * (x - r_2) + c_2$

$y = y * (x - r_1) + c_1$

really

$y \leftarrow y * (x - r_2) + c_2$

"computer reassignment"

General case

$$p(x) = c_1 + (x-r_1)(c_2 + (x-r_2)\,c_3) \qquad n=2$$

$$p(x) = c_1 + (x-r_1)(c_2 + (x-r_2)(c_3 + (x-r_3)c_4)) \qquad n=3$$

$$p(x) = c_1 + (x-r_1)(c_2 + (x-r_2)(c_3 + \cdots + (x-r_n)c_{n+1}\underbrace{)))))}_{\substack{n-1 \\ \text{parentheses}}}$$

Horner's method

```
y = c_{n+1}
for k = n down to 1
    y = y.*(x - r_k) + c_k
end
```

for arrays in MATLAB

Cost:

$m = \text{length}(x)$

$3mn$ flops

For n fixed, Horner's method is an $O(n)$ algorithm (linear complexity)

$$\lim_{n \to \infty} \frac{\text{total cost of algorithm}}{n} = K \text{ (constant)}$$

(there is a more precise defn but this is ok here)

If

$$\lim_{n \to \infty} \frac{\text{total cost of algorithm}}{n^2} = K \text{ (const)}$$

e.g. total cost $= 7n^2 - 5n + 2$

$O(n^2)$ algorithm (quadratic complexity)

<u>EX</u> $A \in \mathbb{R}^{n \times n}$, $\vec{v} \in \mathbb{R}^n$

compute $\vec{w} = A\vec{v}$

$$w_j = \sum_{k=1}^{n} a_{jk} v_k$$

2n-1 flops

$$= a_{j1} v_1 + a_{j2} v_2 + \cdots + a_{jn} v_n$$

$\vec{w}$ has n-components

total cost $2n^2 - n = O(n^2)$