cs223fig34.cdr
Friday, May 20, 2022 10:26:53 PM
Color profile: Disabled

Composite Default screen

Polynomial Evaluation

➪ **Polynomial degree**

➪ Calculate *p* at an x-value (a point)

$$p(x) = a_n x^{n} + \cdots + a_1 x + a_0$$

➪ Seems to involve

- ✎ Computing n terms of form $a_i c^i$
- ✎ Exponentiation of some constant

Definition naturally suggests multiplication as a basic operation.

➪ How well can we exponentiate?

cs223fig34.cdr
Friday, May 20, 2022 10:26:53 PM
Color profile: Disabled
Composite Default screen

Polynomial Evaluation
Brute Force Approach

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

Multiplications, M(n) = ?

Quiz
How many multiplications are needed for the example?

- ✎ How many per term? x^i, $a_i \times x^i$
- ✎ How many terms? Note decreasing exponent
- ✎ Resulting efficiency

Quiz
Is this approach favorable for a **multipoint** (c_1, c_2, … c_m) evaluation scenario?

Polynomial Evaluation
Representation Change

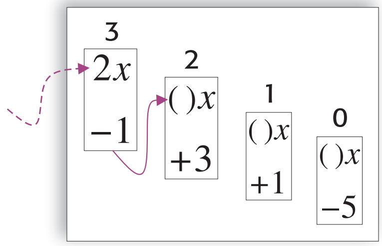
➪ **Horner's rule**

Exercise
Transform $p(x)$ to the alternate algebraic form.

Each column (nested factor) may correspond to an iteration in a *for*-loop.

$$p(x) = 2x^{\overset{n}{4}} - x^3 + 3x^2 + x - 5$$

$$= x(x(x(\underset{4}{2}x - 1) + 3) + 1) - 5$$



$$x^n = xx^{n-1}$$

Once $n-1$ power of x is obtained, do we really need to recompute it to get the next power?

➪ **Observation**

➪ **M(n) = ? By inspection? In general (guess)**

Helps to view an initial inner factor associated with coefficient a_n.

$$x(x(x((2)x - 1) \cdots$$

➪ **Pen-paper example (x=3)**

Focus on developing the procedure rather than getting a final result.

Polynomial Evaluation
Horner's Rule

Quiz
What's the efficiency if addition was chosen as basic operation?

Algorithm *Horner*
Input $P[0..n]$ coefficients $a_0 \cdots a_n$ of polynomial p, point x
Output Polynomial value $p(x)$

$p \leftarrow P[n]$
for $i \leftarrow n-1$ **downto** 0 **do**
    $p \leftarrow x \times p + P[i]$
return p

➪ Efficiency
➪ Applications

Representation Change
Binary Exponentiation

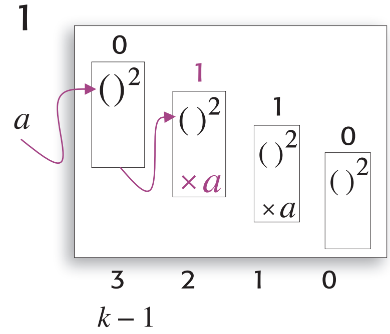
➪ **Successive squaring**

Exercise
Use the diagram to generate a^{22} (binary 10110) where *k* is #steps. Compare to the diagram depicting polynomial evaluation via Horner's rule.

➪ Key idea

✎ Successive squaring

✎ Simple examples



The calculation sequence suggested by Horner's rule + an older idea of exponentiation via successive squaring lead to algorithms that utilize a change of representation of the exponent.

➪ Pen-paper procedure

➪ Algorithms (next)

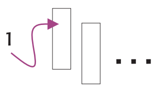
Binary Exponentiation
An Algorithm

Exercise
Compare to a decrease-by-constant factor based on the formula $(a^{n/2})^2$. **Hint**: write the recurrence.

Algorithm *LeftRightBinaryExponentiation*
Input Number a
Input Binary representation $b_l \cdots b_1 b_0$ of integer exponent $n > 0$
$l \leftarrow k$
Output a^n

$p \leftarrow a$
for $i \leftarrow l - 1$ **downto** 0 **do**
  $p \leftarrow p \times p$
  if $b_i = 1$ **then**
    $p \leftarrow p \times a$
return p

Essentially, iterate on an exponent's logarithm (rather than the exponent itself).



Exercise
Modify the pseudocode to initialize p with 1. Will performance change?

Efficiency?

Conclusions

Exercise
Write a recurrence for a divide-conquer exponentiation. Is it a good idea?
Hint: use *WolframAlpha* to check solution.

- **Exponentiation strategies**
- **Polynomial evaluation**

A representation change proves to be a better strategy than trying to improve exponentiation performance.

  - Via exponentiation
  - Using Horner Rule

Exercise
Lookup the FFT computation (problem, algorithms, compare efficiencies).

- **Multipoint scenario**

cs223fig34.cdr
Friday, May 20, 2022 10:26:56 PM
Color profile: Disabled
Composite Default screen

Styles of Iteration

Exercise
Write a summation or a recurrence for definition-based bottom-up and top-down decrease-by-1 algorithms to calculate a^n.

➪ Bottom-up: iterative typically

Build from basic case & work way up

➪ Top-down: recursive typically

Work way down to basic case

➪ Example: exponentiation

Binary vs. decrease-by-const-factor