cs681fig19-20.cdr
Thursday, April 25, 2024 9:53:49 PM
Color profile: Disabled
Composite Default screen

Motivation

> “… some people consider it one of the most important algorithmic discoveries of all time.” Levitin, 3rd

➪ Fast Fourier Transform (FFT)

Involve evaluation of typically large degree polynomials at a large number of points.

- ✎ The problem (computationally)

Little or no effort to reduce steps, just use a more powerful computer to run faster!

- ✎ Brute force approaches next

➪ Final project: a case study

- ✎ Research component
- ✎ Evaluation *exercise some practical skills*

cs681fig19-20.cdr
Thursday, April 25, 2024 9:53:49 PM
Color profile: Disabled

Composite Default screen

Polynomial Evaluation

➪ Polynomial degree

Evaluate the polynomial at a point $x = c$, a fundamental compuatation.

- ➪ **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?**

cs681fig19-20.cdr
Thursday, April 25, 2024 9:53:50 PM
Color profile: Disabled

Composite Default screen

Thinking Review
Exponentiation

To reduce run time, rely on computer power rather than algorithm efficiency (which deals with the run time growth).

Brute force approach, solutions

Directly apply definition, do all possible steps or try all possible alternatives

The definition calls for n−1 mults (#operands less one).

BF exponentiation **efficiency? obviously**

Exercise WolframAlpha
Write recurrence for the first two. **Hint**: one mult is performed in each recursive iteration.

Exercise
Determine the efficiency of divide-conquer exponentiation. Is it a good idea?

Alternatives?

Check the recurrences

x^n

$$x \times x^{n-1}$$

$$(x^{n/2})^2$$

$$x^{n/2} \times x^{n/2}$$

Polynomial Evaluation
Brute Force Approach

$$2x^4 - 3x^3 + 4x^2 - 2x + 1$$

Quiz
How many multiplications are needed for the example? Determine *M*(4) by inspection first.

Multiplications, *M*(n) = ? *M*(4)?

✎ How many per term? x^i, $a_i \times x^i$

Write the sequence in general (*i* runs from 1 to *n*), then a summation.

✎ How many terms? Note decreasing exponent

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

✎ Resulting efficiency

© 2024 Dr. Muhammad Al-Hashimi

cs681fig19-20.cdr
Thursday, April 25, 2024 9:53:50 PM
Color profile: Disabled

Composite Default screen

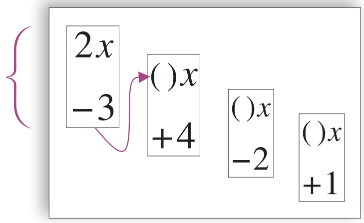
Polynomial Evaluation
Representation Change

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

$$p(x) = 2x^{\overset{n}{\blacktriangledown}4} - 3x^3 + 4x^2 - 2x + 1$$

$$= x\,(x\,(x\,(\underbrace{2x - 3}) + 4) - 2) + 1$$

Each column (nested factor) may correspond to an iteration in a *for*-loop (note processing order of coefficients).



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

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

➪ **Insight**

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

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

➪ **Pen-paper example *x* = 3, *next***

© 2024 Dr. Muhammad Al-Hashimi

cs681fig19-20.cdr
Thursday, April 25, 2024 9:53:50 PM
Color profile: Disabled

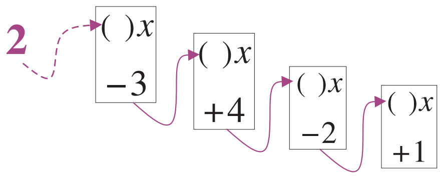
Composite Default screen

Polynomial Evaluation Example

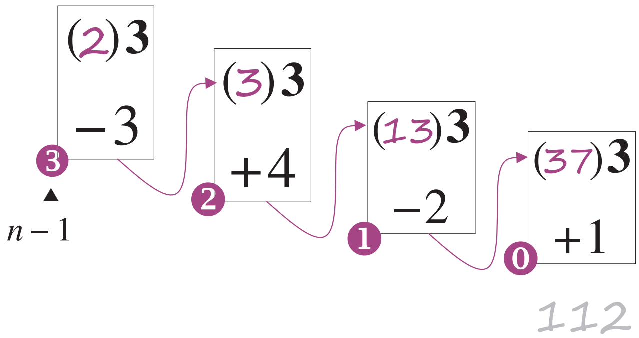
Helps to view an initial inner factor associated with coefficient a_n.

Down the ladder (n = 4)

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



Design iteration based on natural coefficient specification (see general form), therefore, `P[0…n]` where `P[0]` ← a_0. For example, `{+1,-2,+4,-3,+2}`.

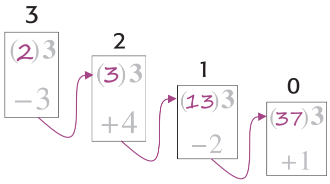


© 2024 Dr. Muhammad Al-Hashimi

cs681fig19-20.cdr
Thursday, April 25, 2024 9:53:50 PM
Color profile: Disabled
Composite Default screen

Polynomial Evaluation
Horner's Method

Input array index follows term exponent



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

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

➪ Efficiency

➪ Applications

cs681fig19-20.cdr
Thursday, April 25, 2024 9:53:50 PM
Color profile: Disabled

Composite Default screen

Conclusions

- **Exponentiation strategies**

Quiz
Compare efficiency (the 3 discussed algorithms).

- **Polynomial evaluation**
  - ✎ Via exponentiation
  - ✎ Using Horner's method

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

Can we do better? In terms of sorting, it is like we can only do bubble or selection sort class computations!

- **Multipoint scenario efficiency**

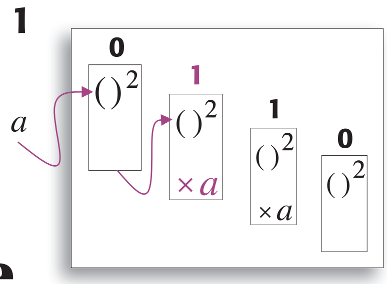
© 2024 Dr. Muhammad Al-Hashimi

Representation Change
Binary Exponentiation

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

⇨ Key idea

- ✎ Successive squaring
- ✎ Simple examples *next*



Exercise
Use figure to generate a^{22} (binary exponent 10110). Compare to the one depicting Horner's method.

⇨ Pen-paper procedure

⇨ Algorithms (later)

© 2024 Dr. Muhammad Al-Hashimi

Binary Exponentiation Examples

The number of steps needed to calculate a^n coincides with the subscript of the left-most bit of the exponent if bits were (classically) labeled right-to-left starting from 0.

Steps = binary length – 1

$a^8 \Leftrightarrow 2^3 \Leftrightarrow 1\,000$

1	$a \cdot a = a^2$
2	$a^2 \cdot a^2 = a^4$
3	$a^4 \cdot a^4 = a^8$

Exactly k=3 steps in general for 2^k

3 2 1 0

$a^{13} \Leftrightarrow 1\,101$

1	$a \cdot a\ (a) = a^2$
2	$a^3 \cdot a^3 = a^6$
3	$a^6 \cdot a^6\ (a) = a^{13}$

Still k=3 steps but $k = \lfloor \log_2 n \rfloor$

cs681fig19-20.cdr
Thursday, April 25, 2024 9:53:51 PM
Color profile: Disabled
Composite Default screen

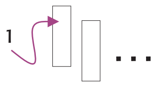
Binary Exponentiation
An Algorithm

Exercise
Compare to a decrease-by-constant factor solution based on the formula $(a^{n/2})^2$.

Algorithm *binaryExponentiation*
Input Number a
Input Binary representation $b_k \cdots b_1 b_0$ of integer exponent $n > 0$
Output a^n

```
p ← a
for i ← k − 1 downto 0 do
    p ← p × p
    if b_i = 1 then
        p ← p × 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?

cs681fig19-20.cdr
Thursday, April 25, 2024 9:53:51 PM
Color profile: Disabled

Composite Default screen

Exercise

© 2024 Dr. Muhammad Al-Hashimi