Introduction: Efficient Algorithms for the Problem of Computing Fibonocci Numbers

Analysis of Algorithms

Prepared by

John Reif, Ph.D.

Readings

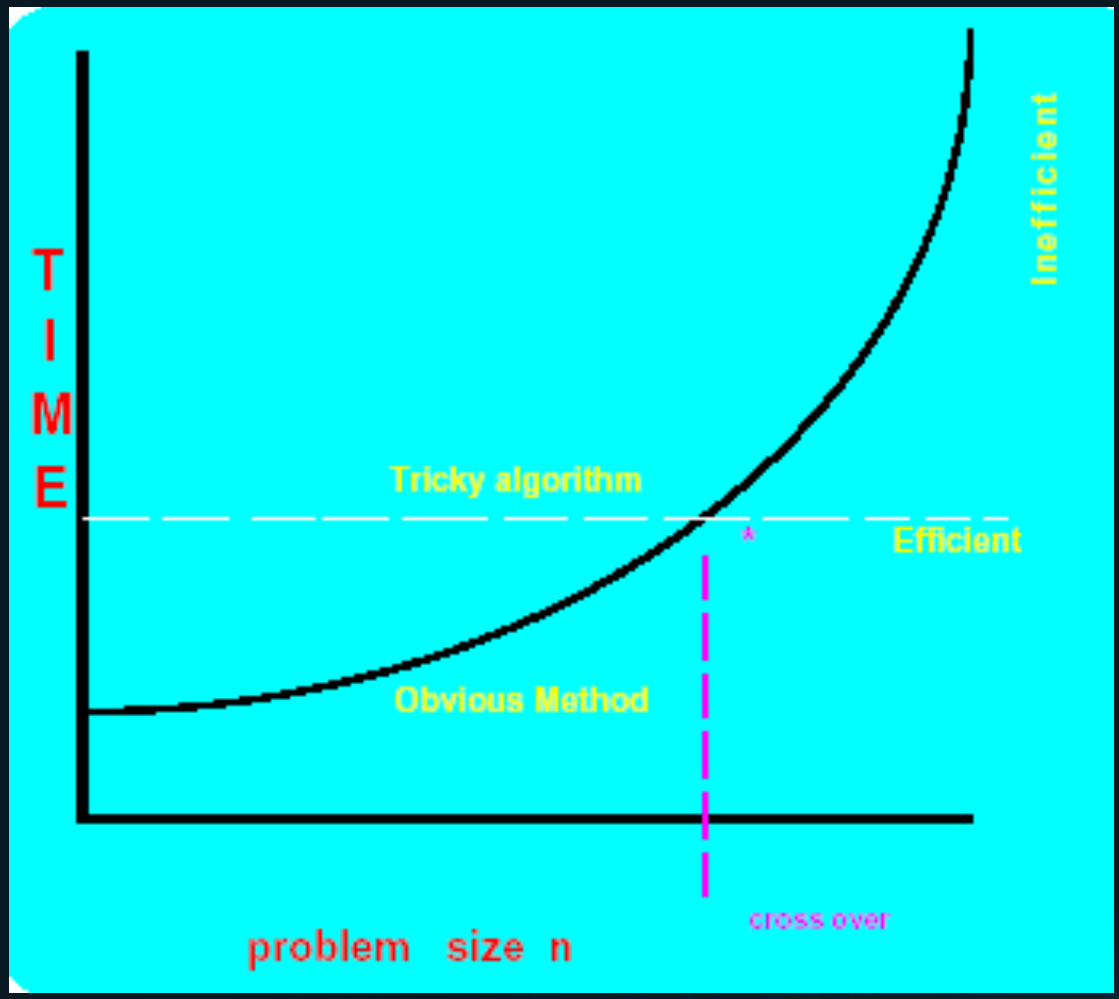
- Main Reading Selection:
  - CLR, Chapter 1

Goal

- Devise Algorithms to solve problems on a TI calculator

- *assume* each step
  (a multiplication, addition, or control branch)
  takes 1 μ sec = 10^{-6} sec
  on 16 bit words

Time Efficiency



Fibonocci Sequence

- 0, 1, 1, 2, 3, 5, 8, 13…
- Recursive Definition
  - F_n = if $n \leq 1$ then n
    else $F_{n-1} + F_{n-2}$
- Problem
  - Compute F_n for $n=10^9$ fast.

Fibonacci Growth

- Can show as n → ∞

$$F_n \sim \frac{\phi^n - (-\phi)^{-n}}{\sqrt{5}}$$

- Golden ratio

$$\phi = \frac{1+\sqrt{5}}{2} = 1.62...$$

- So $F_n \sim .45 \cdot 2^{.7n}$ is .7n bit number grows *exponentially*!

Obvious Method

- $F_n = \begin{cases} n & \text{if } n \leq 1 \\ F_{n-1} + F_{n-2} & \text{if } n > 1 \end{cases}$

costs at least $\frac{n}{2}$ adds of

$\frac{.7n}{2} = .35n$ bit numbers

$$\text{Total } \textit{Cost} \geq \left(\frac{n}{2}\text{ adds}\right)\left(\frac{.35n\text{ bits}}{16\text{ bits}}\right)$$

$\geq .01\ n^2$ steps

$\geq 10^{16}\ \mu$ sec for $n = 10^9$

$= 10^{10}$ sec ~ 317 years!

Wanted!

- An efficient algorithm for F_n
- Weapons:
  - Special properties of computational problems

  = combinatorics (in this case)

Theorem:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}$$

- Proof by Induction
  - Basis Step
    - Holds by definition for n=1
  - Inductive Step
    - Assume holds for some n>0

Inductive Step

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n+1} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}$$

$$= \begin{pmatrix} F_{n+1} + F_n & F_n + F_{n-1} \\ F_{n+1} & F_n \end{pmatrix}$$

$$= \begin{pmatrix} F_{n+2} & F_{n+1} \\ F_{n+1} & F_n \end{pmatrix}$$

Powering Trick

- Fix $M = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$
- To compute M^n when n is a power of 2

  For i=1 to $\lfloor \log n \rfloor$ do

$$M^{2^i} = \left(M^{2^{i-1}}\right) \times \left(M^{2^{i-1}}\right)$$

$$\text{gives } M, M^2, M^{2^2}, \ldots M^{2^{\lfloor \log n \rfloor}}$$

General Case of Powering

- Decompose $n = 2^{j_1} + 2^{j_2} + \ldots + 2^{j_k}$ as sum of powers of 2
- Compute:

$$M^n = \prod_{i=1,\ldots,k} M^{2^{j_i}}$$

- Example:

$$23 = 2^4 + 2^2 + 2^1 + 2^0 = 16 + 4 + 2 + 1 = 10111_2$$

$$M^{23} = M^{16+4+2+1} = M^{2^4+2^2+2^1+2^0} = M^{2^4} M^{2^2} M^{2^1} M^{2^0}$$

Algorithm

$$M = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

(1) compute $M, M^2, \ldots, M^{2^{\lfloor \log n \rfloor}}$ by power method

(2) if $n = 2^{j_1} + 2^{j_2} + \ldots + 2^{j_k}$ then $M^n = \prod_{i=1,\ldots,k} M^{2^{j_i}}$

(3) output F_n = upper right of M^n

Bounding Mults

- = 2 log n matrix products on symmetric matrices of size 2 x 2
- Each matrix product costs 6 integer mults
- Total Cost = 12 (log n) integer mults

  $\geq$ 360 for $n = 10^9$ integer mults

Time Analysis

- $F_n \sim .45\ 2^{.7n}$
- So F_n is m = .7n bit integer

- New method to compute F_n requires multiplying m bit number
- BUT
  - Grammar School Method for Mult takes m^2 *bool ops*
    - $= m^2/16$ steps
    - $\sim 10^{16}$ steps for $n = 10^9$
    - $\sim 10^{16}$ μsec
    - $\sim 10^{10}$ seconds
    - ~ 317 years!

Fast Multiplication

of m bit integers a,b by

- "Divide and Conquer"

$$a = a_L \cdot 2^k + a_R$$

$$b = b_L \cdot 2^k + b_R$$

a_L	a_R
b_L	b_R

$$\text{where } k = \frac{m}{\lfloor 2 \rfloor}$$

$$a \cdot b = a_L b_L 2^{2k} + (a_L b_R + a_R b_L) 2^k + a_R b_R$$

- seems to take 4 multiplications, but…

3 Mult Trick: Karatsubu Algorithm

$$(1)\ x = a_L \cdot b_L$$

$$(2)\ y = a_R \cdot b_R$$

$$(3)\ z = (a_L + a_R)\cdot(b_L + b_R) - (x + y)$$

$$= a_L b_R + a_R b_L$$

Requires only 3 *mults*

on $\frac{m}{2}$ - bit integers and 6 adds on m - bits

$$a \cdot b = a_L b_L 2^{2k} + (a_L b_R + a_R b_L)2^k + a_R b_R$$

$$= x\, 2^{2k} + z \cdot 2^k + y$$

Recursive Mult Algorithm

- Cost

$$C_m \text{ bit ops } C_1 = 1$$

$$C_m \le 3C_{\frac{m}{2}} + 6m \text{ for } m > 1$$

$$\le \sum_{i=0}^{\log m} \left(\frac{3}{2}\right)^i 6m$$

$$\le 2\left(\frac{3}{2}\right)^{\log m} 6m$$

$$\le 12m^{\log_2 3}$$

$$\sim 12m^{1.4} \text{ bit ops}$$

$$\sim 10^{13} \text{ bit ops if } m = .7\ 10^9$$

$$\sim \frac{10^{13}}{16} \text{ steps}$$

$$\sim .05 \text{ years}$$

Schonhage-Strassen Integer Multiplication Algorithm

$10m \log m \cdot \log \log m$ bit ops

$\sim .7 \cdot 10^{12}$ bit ops for $m = .7 \cdot 10^{9}$

$\sim 4.3 \cdot 10^{10}$ steps

$\sim 4.3 \cdot 10^{4}$ sec.

~3 days

- This is feasible to compute!

Improved Algorithm

- Computed F_n for $n = 10^9$
  using 360 = 12 log n integer mults
  each taking ~ 3 days

- Total time to compute F_n
  - ~ 3 years with 1 μsec/step

How Did We Get Even More Speed-Up?

1) New trick
   - a 2 log n mult algorithm (rather than n adds)
2) Old Trick
   - use known efficient algorithms for integer multiplication (rather than Grammar School mult)
3) Also used careful analysis of efficiency of algorithms to compare methods

Problem (to be pondered later…)

- How fast can 10^9 integers be sorted?
- What algorithms would be used?

Introduction: Efficient Algorithms for the Problem of Computing Fibonocci Numbers

Analysis of Algorithms

Prepared by

John Reif, Ph.D.