Algorithms
Professor John Reif

ALG 1.0

Introduction:

Efficient Algorithms For The Problem Of Computing Fibonocci Numbers

Main Reading Selection:
CLR, Chapter 1

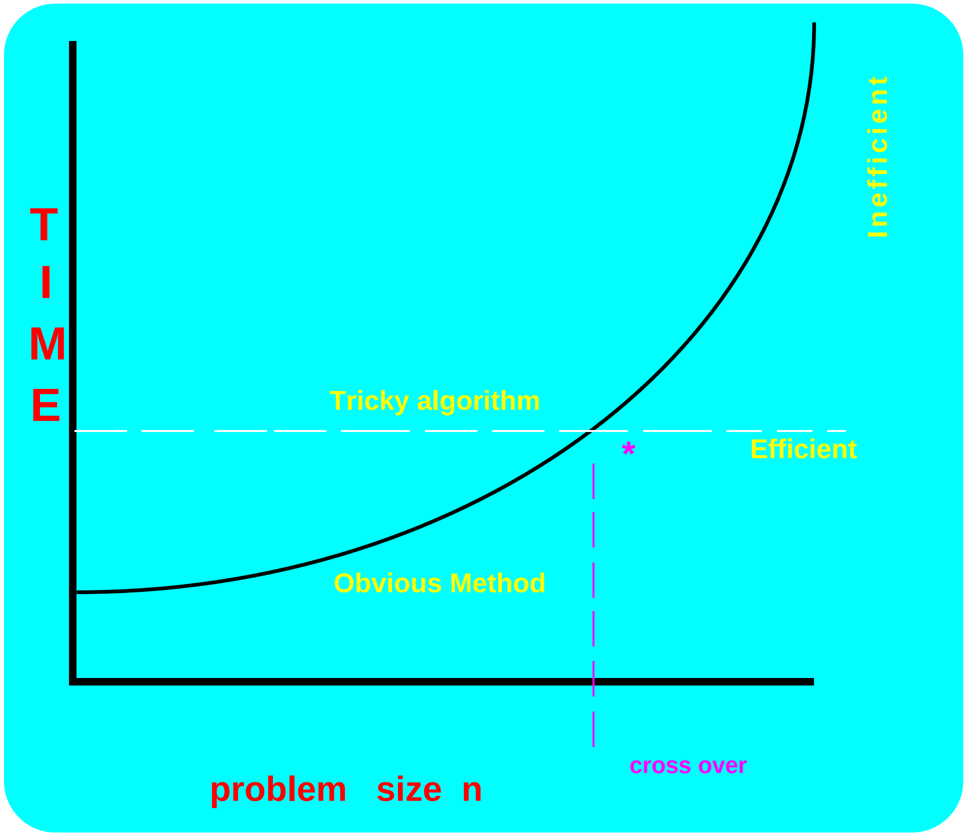
Auxillary Reading Selection:
BB Chapter 1 and Section 4.7

GOAL

devise algorithms to solve problems on Sun machines, e.g. Sun-2

assume **each step**
(a mult, add, or control branch)

takes 1 μsec = 10^{-6} sec
on 16 bit words



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 $F_{1,000,000,000}$

fast on a SUN-2

Can show as $n \to \infty$

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

golden ratio

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

So $F_n \sim .45 \cdot 2^{.7n}$ is

.7n bit number

grows *exponentially!*

Obvious Method

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

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

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

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

$\geq .01\, n^2$ steps

$\geq 10^{16}$ μ sec for $n = 10^9$

$= 10^{10}$ sec

~ 317 years !

WANTED!

An Efficient Algorithm for F_n

Weapons:

--Special Properties of computational problem

= 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

Then :

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

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

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

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

"Powering Trick"

To compute M^n when n is a power of 2

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

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

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

In general case

decompose $n = 2^{j1} + 2^{j2} + \ldots + 2^{jk}$

as sum of powers of 2

example

$$23 = 2^4 + 2^2 + 2^1 + 2^0$$

$$= 16 + 4 + 2 + 1$$

$$= 10111_2$$

compute

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

New 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 $n = 2^{j1} + 2^{j2} + \ldots + 2^{jk}$

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

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

cost

= 2 log n matrix products
on symmetric matrices
of size 2x2

Each matrix product costs
6 integer mults

Total Cost

= 12(log n) integer mults

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

Recall $F_n \sim .45 \; 2^{.7n}$
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*

$= \frac{m^2}{16}$ steps on Sun-2

$\sim 10^{16}$ steps for $n = 10^9$
$\sim 10^{16}$ µ sec
$\sim 10^{10}$ 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

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

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

seems to take *4 mults* , but....

"3 Mult Trick"

(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 bit ops $\quad C_1 = 1$

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

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

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

$$\le 6m^{\log_2 3}$$

$\sim 6m^{1.4}$ bit ops

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

$\sim \frac{10^{13}}{16}$ steps of SUN-2

$\sim$ .05 years on SUN-2

Schonhage - Strassen Integer Multiplication Algorithm

(See AHU-Data page 270-274)

10m log m · 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 on Sun-2

$\sim 4.3 \cdot 10^4$ sec. on Sun-2

$\sim$ 3 days on Sun-2

This is feasible to Compute!

New Algorithm

computed F_n for $n = 10^9$
using 360 = 12 log n integer mults
each taking ~ 3 days on Sun-2

Total Time to compute F_n

~ 3 years on single Sun-2 with 1 μ sec/step

(or 6 days on 5 nano sec/step Cray)

How did we get sequential 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 on a Sun-2?

⇒

What algorithms would be used?