Great Theoretical Ideas In Computer Science

Anupam Gupta CS 15-251 Fall 2005

Lecture 5 Sept 13, 2005 Carnegie Mellon University

Ancient Wisdom: On Raising A Number To A Power

Egyptian Multiplication

The Egyptians used decimal numbers but multiplied and divided in binary

Egyptian Multiplication a times b by repeated doubling

b has some n-bit representation: $b_{n-1}b_{n-2}\ldots b_1b_0$

$(101)_2 = 5$

Starting with a,
repeatedly double largest number so far to obtain: a, 2a, 4a, …., $2^{n-1}a$

$a, 2a, 4a$

Sum together all the $2^k a$ where $b_k = 1$

$a + 4a$

Egyptian Multiplication 15 times 5 by repeated doubling

b has some n-bit representation: $b_{n-1}b_{n-2}\ldots b_1b_0$

Starting with a,
repeatedly double largest number so far to obtain: a, 2a, 4a, …., $2^{n-1}a$

Sum together all the $2^k a$ where $b_k = 1$

$b = 13 = (1101)_2$

$a, 2a, 4a, 8a$

add them $= 13a$

Why does that work?

$b = b_0 2^0 + b_1 2^1 + b_2 2^2 + \ldots + b_{n-1}2^{n-1}$

$(1101)_2 = 1.2^0 + 0.2^1 + 1.2^2 + 1.2^3 = 13$

$ab = a.b_0.2^0 + a.b_1.2^1 + \cdots + a.b_{n-1}.2^{n-1}$

$= (a.2^0)b_0 + (a.2^1)b_1 + \cdots + (a.2^{n-1})b_{n-1}$

$= \sum_{i:\, b_i = 1} (a.2^i)$

Why does that work?

$b = b_0 2^0 + b_1 2^1 + b_2 2^2 + \ldots + b_{n-1}2^{n-1}$

$ab = b_0 2^0 a + b_1 2^1 a + b_2 2^2 a + \ldots + b_{n-1}2^{n-1}a$

If b_k is 1 then $2^k a$ is in the sum.
Otherwise that term will be 0.

Wait!
How did the Egyptians do the part where they converted b to binary?

They used repeated halving to do base conversion. Consider …

Egyptian Base Conversion

```
Output stream will print right to left.

Input X;
repeat {
    if (X is even)
        then print 0;
    else
        {X := X-1; print 1;}

    X := X/2 ;
} until X=0;
```

13 even? N → 1 ↑
6 even? Y 0
3 even? N 1
1 even? N 1

$(1101)_2 = 13.$

Sometimes the Egyptian combined the base conversion by halving and multiplication by doubling into a single algorithm

Rhind Papyrus (1650 BC)
70*13

			running sum
70	13	~~✓~~ ✓	70
140	6		
280	3	✓	350
560	1	✓	910

Answer: 910

Rhind Papyrus (1650 BC)
70*13

70	13 *	70
140	6	
280	3 *	350
560	1 *	910

Binary for 13 is 1101 = $2^3 + 2^2 + 2^0$
$70*13 = 70*2^3 + 70*2^2 + 70*2^0$

Rhind Papyrus (1650 BC)
Division: 184/17

17	1
34	2 *
68	4
136	8 *

184 48 14

Rhind Papyrus (1650 BC)
Division: 184/17

17	1
34	2 *
68	4
136	8 *

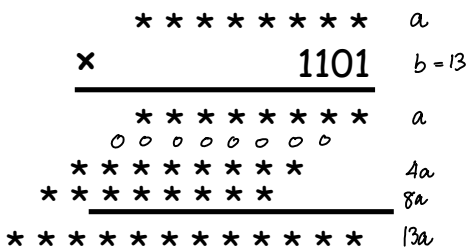
184 48 14

184 = 17*8 + 17*2 + 14
184/17 = 10 with remainder 14

This method is called "Egyptian Multiplication/Division" or "Russian Peasant Multiplication/Division".

Wow. Those Russian peasants were pretty smart.

Standard Binary Multiplication = Egyptian Multiplication



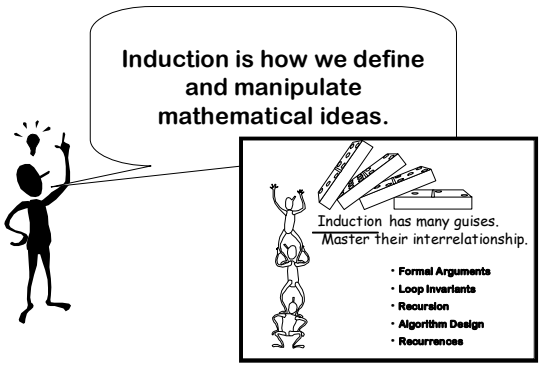
Our story so far…

We can view numbers in many different, but corresponding ways.

Representation:
Understand the relationship between different representations of the same information or idea

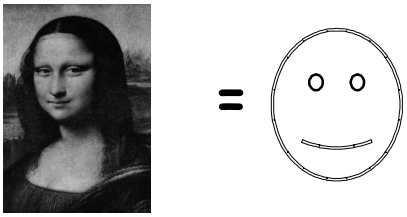
1	◯
2	◯◯
3	◯◯◯

Our story so far…

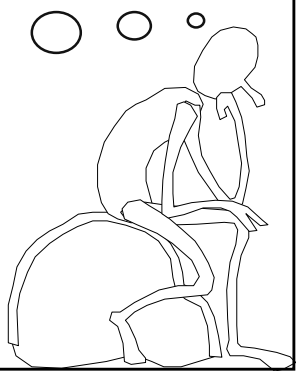


Let's Articulate A New One:

Abstraction:
Abstract away the inessential features of a problem or solution



Even very simple computational problems can be surprisingly subtle.



Compiler Translation

A compiler must translate a high level language (e.g., C) with complex operations (e.g., exponentiation) into a lower level language (e.g., assembly) that can only support simpler operations (e.g., multiplication).

$b:=a^8$

b:=a*a
b:=b*a
b:=b*a
b:=b*a
b:=b*a
b:=b*a
b:=b*a

b:=a*a
b:=b*b
b:=b*b

This method costs only 3 multiplications. The savings are significant if $b:=a^8$ is executed often.

General Version

Given a constant n,
how do we implement
$b:=a^n$
with the fewest number of multiplications?

Powering By Repeated Multiplication

Input: a, n

Output: A sequence starting with a, ending with a^n, such that each entry other than the first is the product of two previous entries.

Example

Input: a,5

a^5 : $(a \quad a^2 \quad a^3 \quad a^4 \quad a^5)$

4 multiplications

$(a \quad a^2 \quad a^4 \quad a^5)$

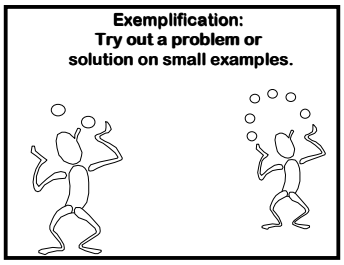
$(a \quad a^2 \quad a^3 \quad a^5)$

Definition of M(n)

M(n) = The minimum number of multiplications required to produce a^n from a by repeated multiplication

$? \leq M(n) \leq ?$

What is M(n)? Can we calculate it exactly? Can we approximate it?



Some Very Small Examples

M(n) = The minimum number of multiplications required to produce a^n from a by repeated multiplication

$M(1) = 0$ [a]

$M(2) = 1$ $[a \quad a^2]$

$M(0)$ undefined

$M(8) \leq 3$ $[a \quad a^2 \quad a^4 \quad a^8]$

M(8) = ?

a, a^2, a^4, a^8 is one way to make a^8 in 3 multiplications.

What does this tell us about the value of M(8)?

M(8) = ?

a, a^2, a^4, a^8 is one way to make a^8 in 3 multiplications.

What does this tell us about the value of M(8)?

$M(8) \leq 3$

Upper Bound

$? \leq M(8) \leq 3$

$? \leq M(8) \leq 3$

$3 \leq M(8)$

Exhaustive Search. There are only two sequences with 2 multiplications. Neither of them make 8:

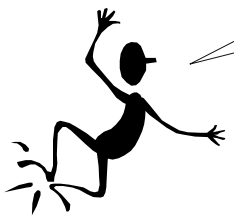
a, a^2, a^3 & a, a^2, a^4

$M(8) \geq 3$

$3 \leq M(8) \leq 3$

Lower Bound

Upper Bound



M(8) = 3

Applying Two Ideas

Abstraction:
Abstract away the inessential features of a problem or solution

=

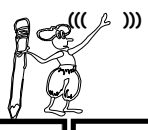
Representation:
Understand the relationship between different representations of the same information or idea

1 ○

2 ○○

3 ○○○

What is the more essential representation of M(n)?



Abstraction:
Abstract away the inessential features of a problem or solution

=

Representation:
Understand the relationship between different representations of the same information or idea

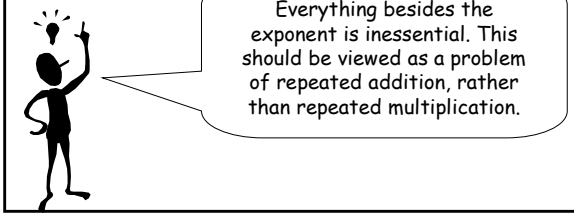
1 ○

2 ○○

3 ○○○

The "a" is a red herring.

$a^x \times a^y$ is a^{x+y}



Addition Chains

M(n) = Number of stages required to make n, where we start at 1 and in each subsequent stage we add two previously constructed numbers.

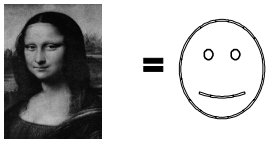
Examples

Addition Chain for 8:
1 2 3 5 8

Minimal Addition Chain for 8:
1 2 4 8 $M(8)=3$

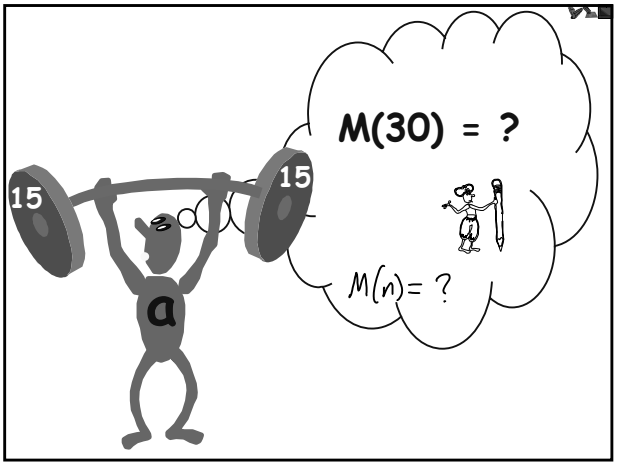
Addition Chains Are A Simpler Way To Represent The Original Problem

Abstraction:
Abstract away the inessential features of a problem or solution



Representation:
Understand the relationship between different representations of the same information or idea

1 ○
2 ○○
3 ○○○



Some Addition Chains For 30

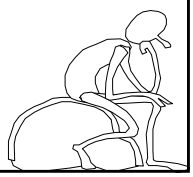
1 2 4 8 16 24 28 30 (7)
1 2 4 6 12 24 30 (6)
1 2 4 5 10 20 30 (6)
1 2 4 8 10 20 30 (6)
1 2 3 5 10 15 30 (6)

Some Addition Chains For 30

1	2	4	8	16	24	28	30
1	2	4	5	10	20	30	
1	2	3	5	10	15	30	
1	2	4	8	10	20	30	

$? \le M(30) \le 6$

$? \le M(n) \le ?$



Binary Representation

Let B_n be the number of "1"s in the binary representation of n.

E.g.: $B_5 = 2$ since 101 = binary representation of 5

$B_{13} = 3 \qquad (1101)_2 = 13$

Proposition: $B_n \le \lfloor \log_2 (n) \rfloor + 1$
(It is at most the number of bits in the binary representation of n.)

Binary Method
Repeated Squaring Method
(Repeated Doubling Method)

Phase I (Repeated Doubling)

For $\lfloor \log_2 n \rfloor$ **stages:**

Add largest so far to itself

(1, 2, 4, 8, 16, . . .)

Phase II (Make n from bits and pieces)

Expand n in binary to see how n is the sum of B_n powers of 2. Use $B_n - 1$ stages to make n from the powers of 2 created in phase I

Total cost: $\lfloor \log_2 n \rfloor + B_n - 1$

Binary Method Applied To 30

30 — Binary 11110

1 2 4 8 16

24 28 30

$\lfloor \log_2 30 \rfloor = 4$ additions

# of additions = (# of 1's in 11110) − 1

# of additions $\le \lfloor \log_2 n \rfloor + (B_n - 1) \le 2 \lfloor \log_2 n \rfloor$

Rhind Papyrus (1650 BC)
What is 30 times 5?

			running sum
5	30	✓	10
10	15	✓	30
20	7	✓	70
40	3	✓	150
80	1	✓	

$\lfloor \log_2 n \rfloor$

# of checks = B_n

$\Rightarrow$ # of additions = $B_n - 1$

total $\le \lfloor \log_2 n \rfloor + B_n - 1$

Rhind Papyrus (1650 BC)
What is 30 times 5?

1	5
2	10
4	20
8	40
16	80
24	120
28	140
30	150

30 by a chain of 7:

1 2 4 8 16 24 28 30

Repeated doubling is the same as the Egyptian binary multiplication

Rhind Papyrus (1650 BC)
Actually used faster chain for 30*5.

1	5
2	10
4	20
8	40
10	50
20	100
30	150

30 by a chain of 6:

1 2 4 8 10 20 30

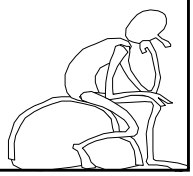
The Egyptian Connection

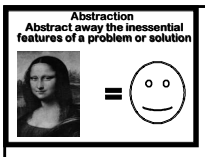
A shortest addition chain for n gives a shortest method for the Egyptian approach to multiplying by the number n.

The fastest scribes would seek to know M(n) for commonly arising values of n.

$$? \leq M(n) \leq \lfloor \log_2 n \rfloor + B_n - 1$$
$$\leq 2 \lfloor \log_2 n \rfloor$$

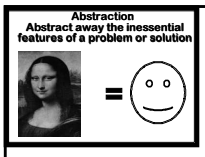
(B_n: # of 1's in the binary repr. of n)





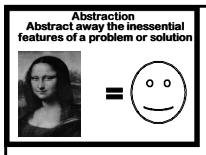
We saw that applying ABSTRACTION to the PROBLEM simplifies the issue.

PROBLEM = Raising A Number To A Power.



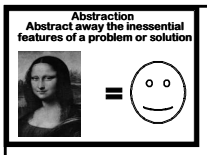
What about applying ABSTRACTION to the SOLUTION ?

Let SOLUTION be the Repeated Squaring Algorithm (RQA).

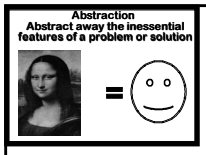


What features did our solution (RQA) actually make use of?

For example, does the RQA require the underlying objects to be numbers?



The repeated squaring method works for modular arithmetic and for raising a matrix to a power.



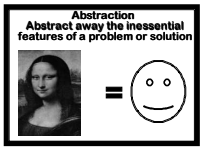
The repeated squaring method works for any notion of "multiplication" that is associative.

$(a*b)*c = a*(b*c)$

a^k is well defined

$a^x * a^y = a^{x+y}$

Generalization

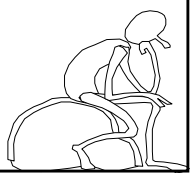


Solution

Always ask yourself what your solution actually requires.

$? \le M(30) \le 6$

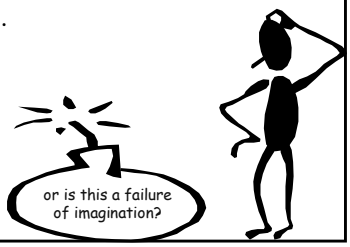
$? \le M(n) \le 2\lfloor \log_2 n \rfloor$



A Lower Bound Idea

You can't make any number bigger than 2^n in n steps.

1 2 4 8 16 32 64 . . .



Theorem: For all $n \geq 0$, no n stage addition chain will contain a number greater than 2^n.

S_k = "any addition chain with k additions can only produce a number $\leq 2^k$"

Thm: $\forall k \geq 0$, S_k.

Proof: By induction on k

Base Case: k=0, true.

Theorem: For all $n \geq 0$, no n stage addition chain will contain a number greater than 2^n.

Inductively assume that S_k is true. (Ind. Hypothesis)

To prove (using I.H.) that S_{k+1} is true.

~~Suppose I obtain a~~ Any number obtained by the $(k+1)^{th}$ addition must add together 2 numbers produced using at most k additions. By I.H, these are $\leq 2^k$. Hence the resulting number $\leq 2^k + 2^k \leq 2^{k+1}$. (Hence S_{k+1} holds.)

Let S_k be the statement that no k stage addition chain will contain a number greater than 2^k

Base case: k=0. S_0 is true since no chain can exceed 2^0 after 0 stages.

$\forall k \geqslant 0, \quad S_k \Rightarrow S_{k+1}$

At stage k+1 we add two numbers from the previous stage. From S_k we know that they both are bounded by 2^k. Hence, their sum is bounded by 2^{k+1}. No number greater than 2^{k+1} can be present by stage k+1.

Another Proof (using invariants, this time)

Invariant: All the numbers created by stage n are less than or equal to 2^n.

The invariant is true at the start.

Suppose we are at stage k. If the invariant is true, then the two numbers we decide to sum for stage k+1 are $\leq 2^k$ and hence create a number less than or equal to 2^{k+1}. The invariant is thus true at stage k+1.

Change Of Variable

All numbers obtainable in m stages are bounded by 2^m. Let $m = \log_2(n)$.

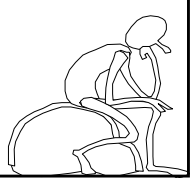
$\Rightarrow$ in $\log_2 n$ stages, any number produced $\leq n$

$\Rightarrow$ in $\lceil \log_2 n \rceil$ stages, numbers produced $\leq n$

$\Rightarrow M(n) \geq \lceil \log_2 n \rceil$

$? \leq M(30) \leq 6$

$\lceil \log_2 n \rceil \leq M(n) \leq 2 \lfloor \log_2 n \rfloor$



Theorem: 2^i is the largest number that can be made in i stages, and can only be made by repeated doubling

By induction

Base Case: ✓

Inductively: ~~2^{i+1}~~ to get 2^{i+1} in $(i+1)^{st}$ addition I must add $2^i + 2^i$. But inductively, I can only get 2^i by repeated doubling in all i stages. ⇒ implies the proof.

Theorem: 2^i is the largest number that can be made in i stages, and can only be made by repeated doubling

Base i = 0 is clear.

To make anything as big as 2^i requires having some X as big as 2^{i-1} in i-1 stages.

By I.H., we must have all the powers of 2 up to 2^{i-1} at stage i-1. Hence, we can only double 2^{i-1} at stage i.

The theorem follows.

5 < M(30) ?

Suppose M(30) = 5.

In addition #5, must add 15+15, or 16+14.

Case 1: 16+14. in 5th addition

⇒ first 4 additions, I got 14, 16.

but repeated doubling does not give 14.

not possible

Suppose M(15) = 4

Case 2: added 15+15 in 5th add.

M(15) = 4.

In addition #4, must added 8+7

⇒ must produce both 7, 8 in 3 additions.

Can't do.

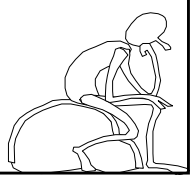
⇒ Case 2 not possible ⇒ Contradicts M(30)=5

⇒ M(30) ≥ 6.

M(30)=6

M(30) = 6

$\lceil \log_2 n \rceil \le M(n) \le 2 \lfloor \log_2 n \rfloor$



Rhind Papyrus (1650 BC)

1	5
2	10
4	20
8	40
10	50
20	100
30	150

30 = 1 2 4 8 10 20 30

Factoring Bound

$M(a \times b) \leq M(a) + M(b)$

Proof:

- Construct a in M(a) additions
- Using a as a unit follow a construction method for b using M(b) additions. In other words, each time the construction of b refers to a number y, use the number ay instead.

Corollary (Using Induction)

$M(a_1a_2a_3\ldots a_n) \leq M(a_1)+M(a_2)+\ldots+M(a_n)$

Proof: For n=1 the bound clearly holds. Assume it has been shown for up to n-1. Now apply theorem using $a= a_1a_2a_3\ldots a_{n-1}$ and $b=a_n$ to obtain:

$$M(a_1a_2a_3\ldots a_n) \leq M(a_1a_2a_3\ldots a_{n-1}) + M(a_n)$$

By inductive assumption,

$$M(a_1a_2a_3\ldots a_{n-1}) \leq M(a_1) + M(a_2) + \ldots + M(a_{n-1})$$

More Corollaries

Corollary: $M(a^k) \leq kM(a)$

Corollary: $\mathbf{M(p_1^{\alpha_1}p_2^{\alpha_2}p_3^{\alpha_3}\ldots p_n^{\alpha_n})}$

$\mathbf{\leq \alpha_1 M(p_1) + \alpha_2 M(p_2) + \ldots + \alpha_n M(p_n)}$

Does equality hold?

M(33) < M(3) + M(11)

M(3) = 2 [1 2 3]

M(11)= 5 [1 2 3 5 10 11]

M(3) + M(11) = 7

M(33) = 6 [1 2 4 8 16 32 33]

The conjecture of equality fails. There have been many nice conjectures. . . .

Conjecture: M(2n) = M(n) +1 (A. Goulard)

A fastest way to an even number is to make half that number and then double it.

Proof given in 1895 by E. de Jonquieres in L'Intermediere Des Mathematiques volume 2, pages 125-12

FALSE!
M(191)=M(382)=11
Furthermore, there are infinitely many such examples.

Open Problem

Is there an n such that

$M(2n) < M(n)$?

Conjecture

Each stage might as well consist of adding the largest number so far to one of the other numbers.

First Counter-example: **12509**
[1 2 4 8 16 17 32 64 128 256 512 1024 1041 2082 4164 8328 8345 12509]

Open Problem

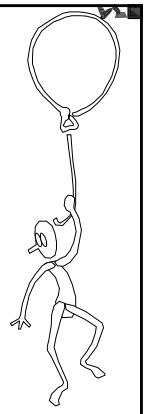
Prove or disprove the Scholz-Brauer Conjecture:

$$M(2^n-1) \leq n - 1 + B_n$$

(The bound that follows from this lecture is too weak: $M(2^n-1) \leq 2n - 1$)

High Level Point

Don't underestimate "simple" problems. Some "simple" mysteries have endured for thousand of years.



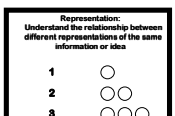
Study Bee

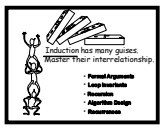
Egyptian Multiplication

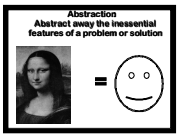
Raising To A Power
Minimal Addition Chain
Bracketing: Lower and Upper Bounds

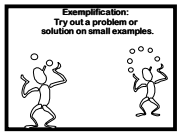
RQA [Repeated Squaring Algorithm]
RQA works for ANY binary operator

Study Bee









REFERENCES

The Art Of Computer Programming, Vol 2, pp. 444 – 466, by Donald Knuth