Great Theoretical Ideas In Computer Science

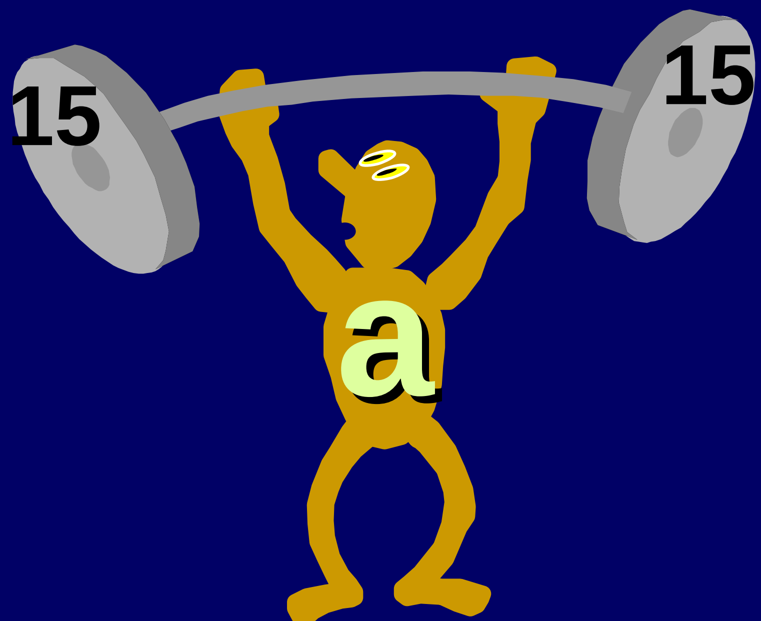
Steven Rudich

CS 15-251 Spring 2005

Lecture 5 Jan 25, 2005

Carnegie Mellon University

Ancient Wisdom: On Raising A Number To A Power



Rhind Papyrus (1650 BC)
70*13

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

17	1
34	2 *
68	4
136	8 *

184 48 14

Rhind Papyrus (1650 BC)

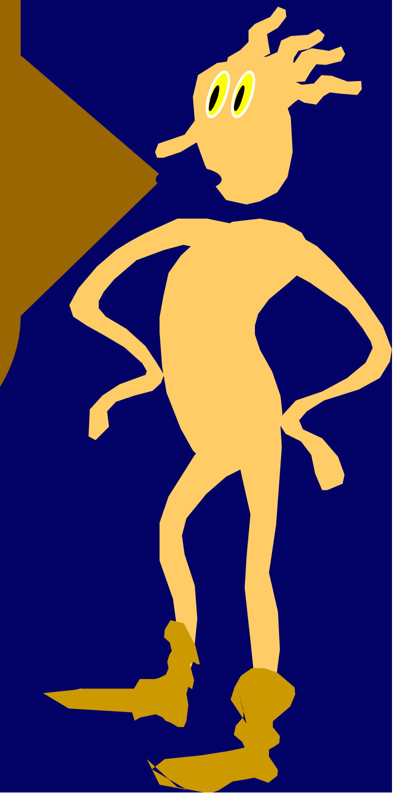
17	1
34	2 *
68	4
136	8 *

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

X * * * * * * * *

101

* * * * * * * *

* * * * * * * *

* * * * * * * * * * *

Egyptian Base 3

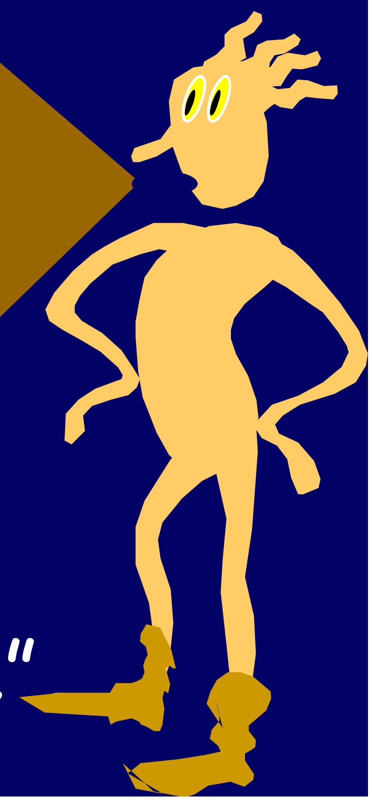
Convention Base 3:
Each digit can be 0, 1, or 2

Here is a strange new one:
Egyptian Base 3 uses -1, 0, 1

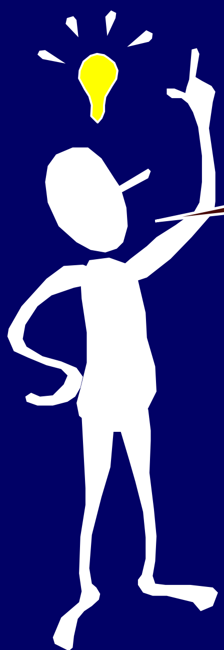
Example: 1 -1 -1 = 9 - 3 - 1 = 5

How could this be Egyptian? Historically, negative numbers first appear in the writings of the Hindu mathematician Brahmagupta (628 AD).

One weight for each power of 3.
Left = “negative”. Right = “positive”



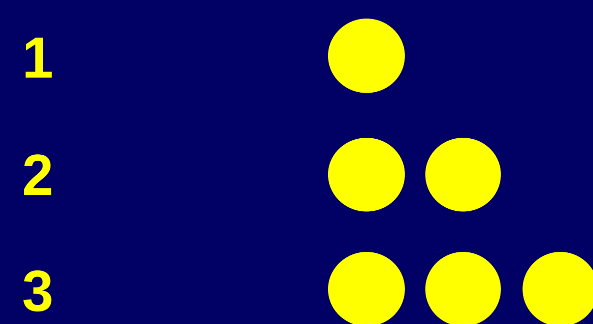
Our story so far .....



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

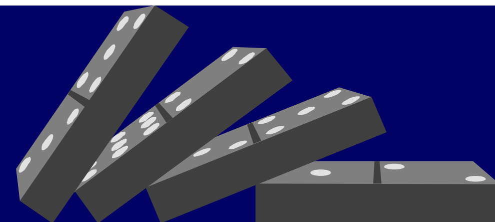
Representation:

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



So Far We Have Seen:

Induction is how we define and manipulate mathematical ideas.



Induction has many guises.
Master their interrelationship.

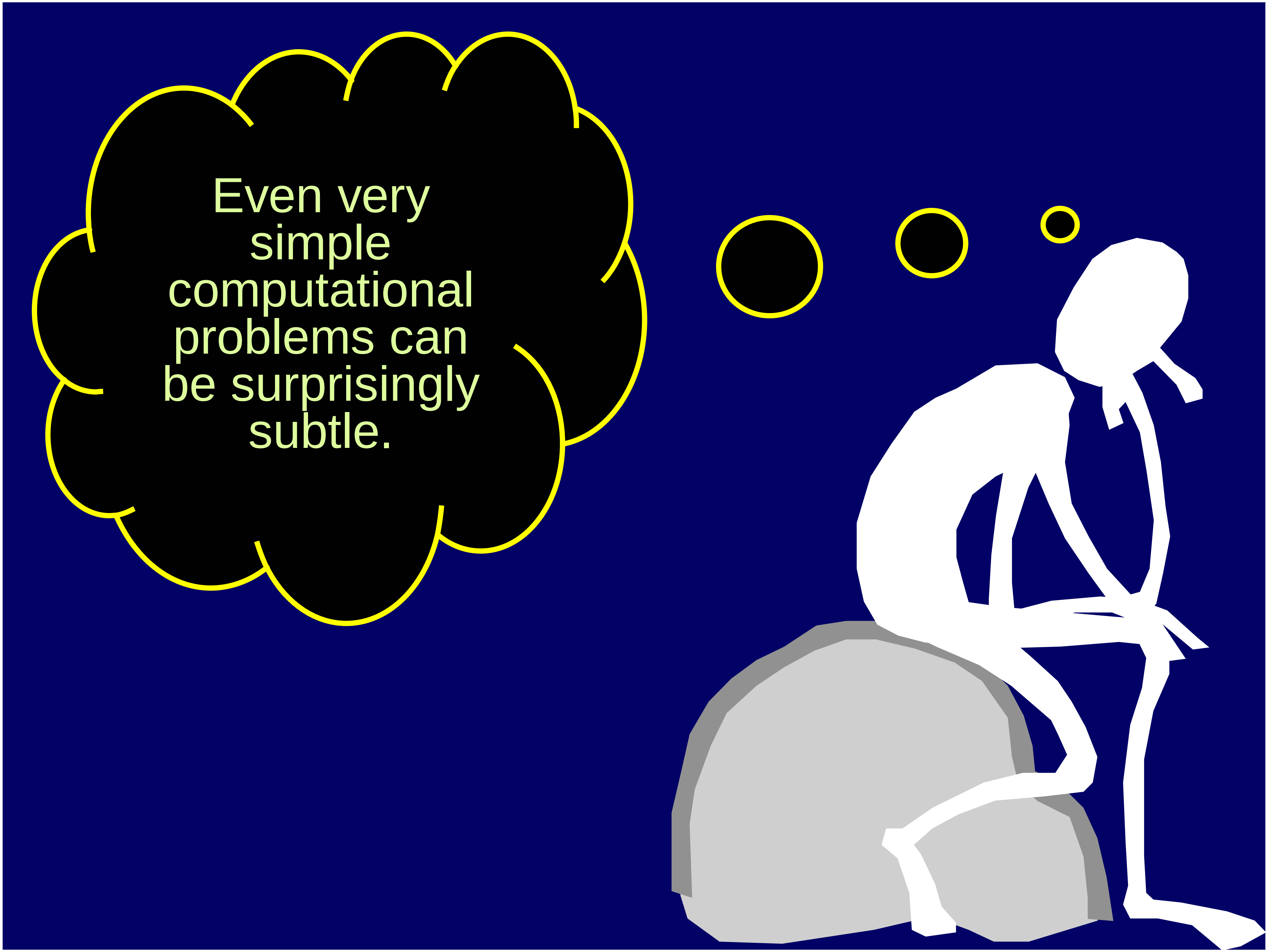
- Formal Arguments
- Loop Invariants
- Recursion
- Algorithm Design
- Recurrences

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

Powering By Repeated Multiplication

Input: a,n

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

Example

Input: $a, 5$

Output: a, a^2, a^3, a^4, a^5

or

Output: a, a^2, a^3, a^5

or

Output: a, a^2, a^4, a^5

Definition of M(n)

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

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

Exemplification:
Try out a problem or
solution on small examples.

Some Very Small Examples

What is M(1)?

- M(1) = 0   [a]

- What is M(0)?
  - M(0) is not clear how to define
- What is M(2)?
  - M(2) = 1   [a, a^2]

$M(8) = ?$

a, a^2, a^4, a^8 is a 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 a way to make a^8 in 3 multiplications. What does this tell us about the value of $M(8)$?

$$\mathbf{M(8) \leq 3}$$

Upper Bound

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

Lower Bound

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

Lower Bound

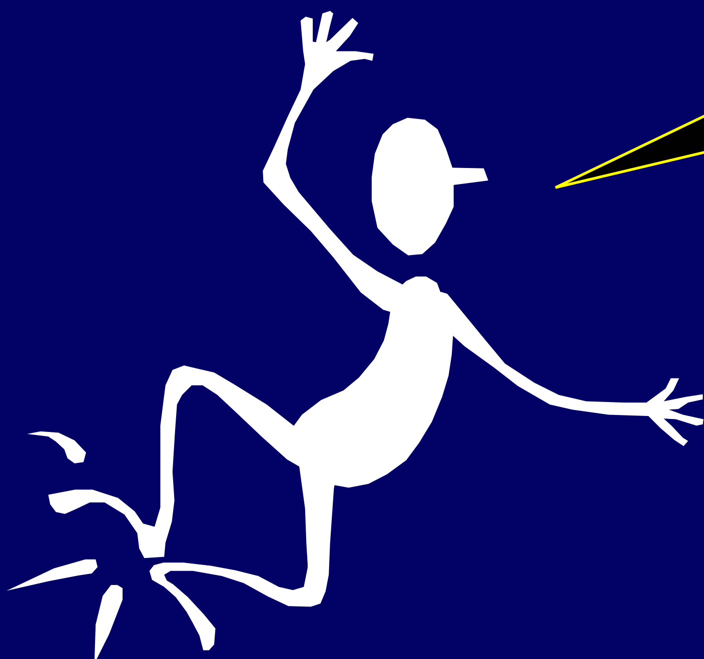
$$3 \le M(8)$$

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

$$a,\ a^2,\ a^3 \ \&\ a,\ a^2,\ a^4$$

$$3 \le M(8) \le 3$$

Lower Bound	Upper Bound

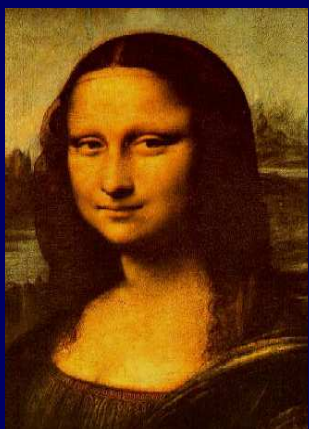


M(8) = 3

Applying Two Ideas

Abstraction:

Abstract away the inessential features of a problem or solution



=

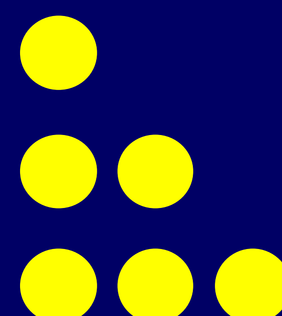
Representation:

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

1

2

3

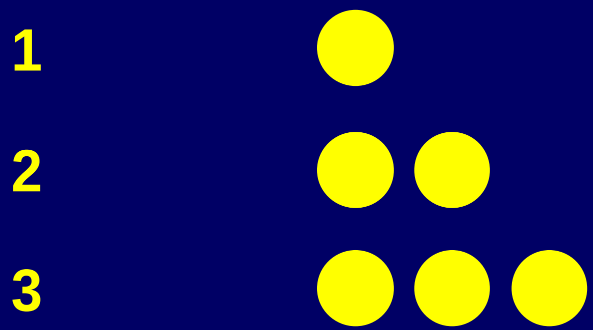


What is the more essential representation of M(n)?

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

=

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



The a is a **red herring**.

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

Everything besides the exponent is inessential. This should be viewed as a problem of repeated addition, rather than repeated multiplication.

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

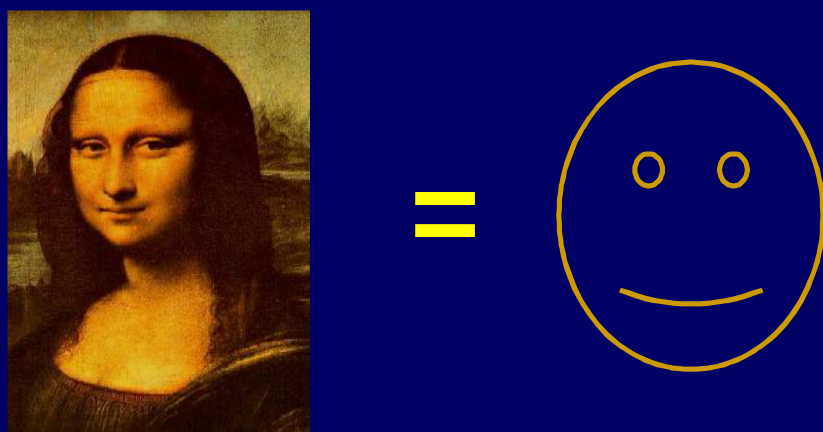
Minimal Addition Chain for 8:

1 2 4 8

Addition Chains Are A Simpler To Represent The Original Problem

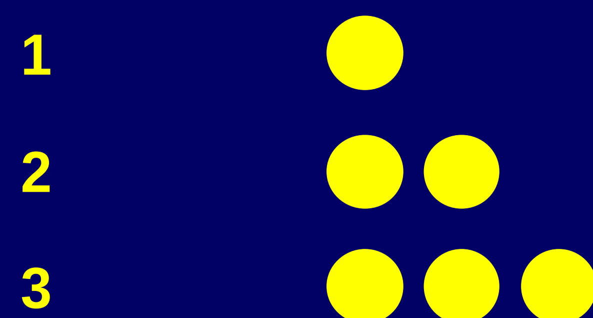
Abstraction:

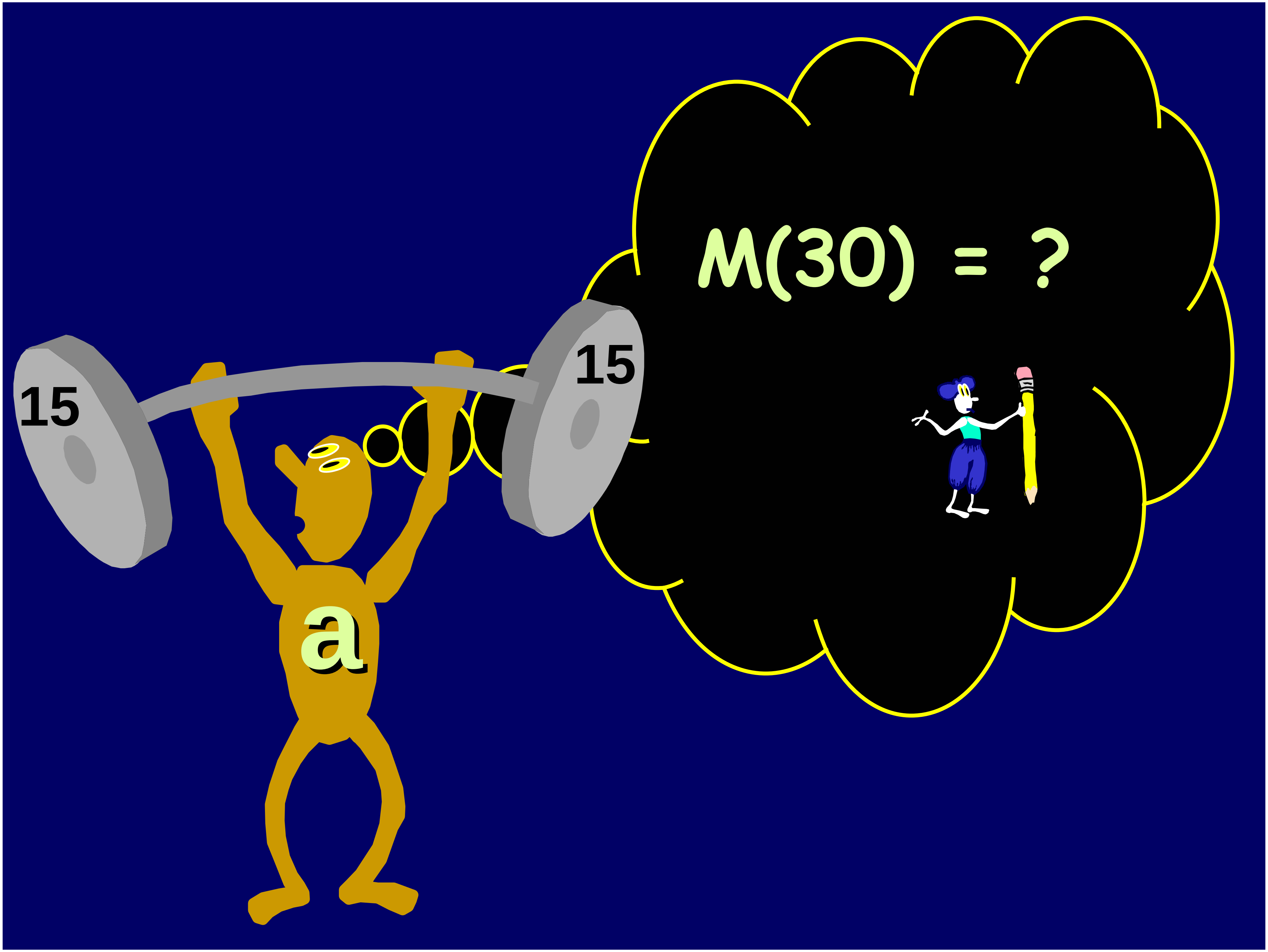
Abstract away the inessential features of a problem or solution



Representation:

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




M(30) = ?
15
15
a

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 1s in the binary representation of n. Ex: $B_5 = 2$ since 101 is the binary representation of 5

Proposition: $B_n \leqslant \lfloor \log_2(n) \rfloor + 1$

The length of the binary representation of n is bounded by this quantity.

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

	Binary
30	11110
Phase I	
1	1
2	10
4	100
8	1000
16	10000

Phase II: 6 14 30 (Cost: 7 additions)

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:

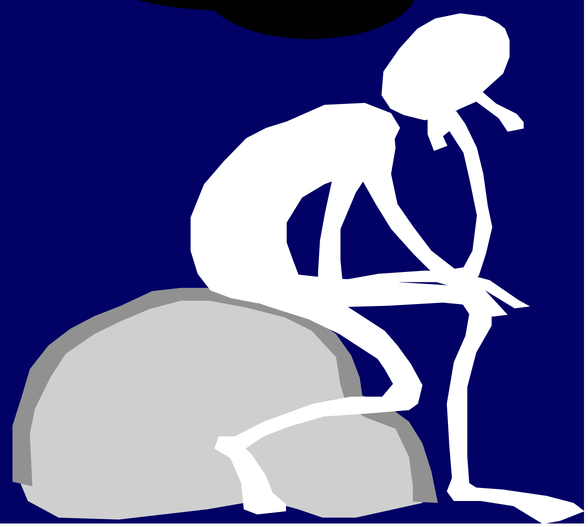
1 2 4 8 10 20 30

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.

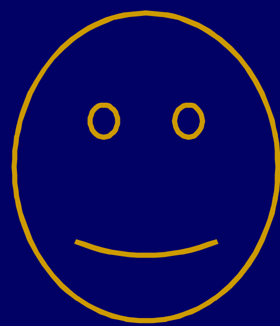
$$M(n) \le \lfloor \log_2 n \rfloor + B_n - 1 \le 2\lfloor \log_2 n \rfloor$$



Abstraction:

Abstract away the inessential
features of a problem or solution

=

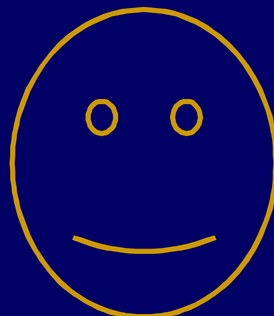


We saw that applying ABSTRACTION to the PROBLEM simplifies the issue.

PROBLEM = Raising A Number To A Power.

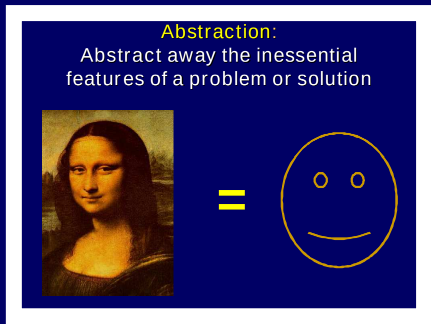
Abstraction:
Abstract away the inessential features of a problem or solution

=

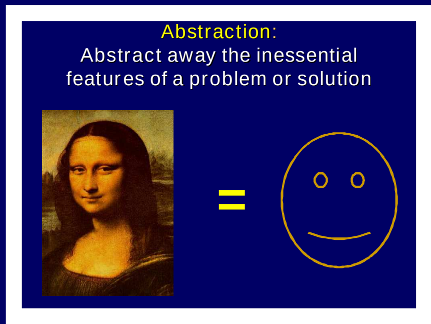


What about ABSTRACTION to the SOLTUTION ????

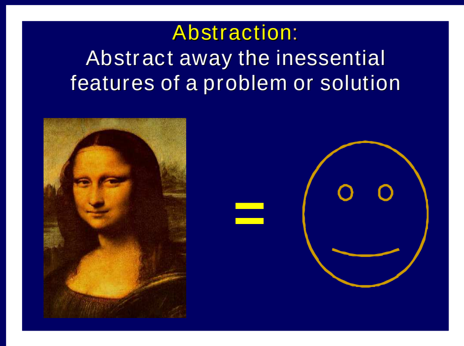
Let SOLUTION be the Repeated Squaring Algorithm.



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



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

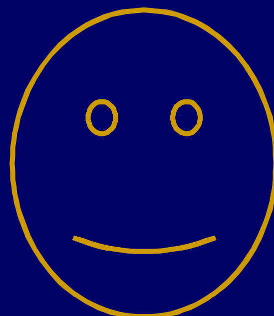


The repeated squaring method works for modular arithmetic and for raising a matrix to a power.

Abstraction:

Abstract away the inessential features of a problem or solution

=



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

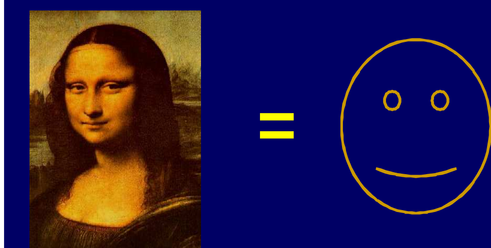
$(a*b)*c = a*(b*c)$

a^k is well defined

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

GENERALIZATION

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



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 . . .

Induction Proof

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

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.

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

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.

Proof By Invariant (Induction)

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 $m = \log_2(n)$.

Thus, All numbers obtainable in $\log_2(n)$ stages are bounded by n.

$$M(n) \geq \log_2(n)$$

In fact, $M(n) \geq \lceil \log_2(n) \rceil$

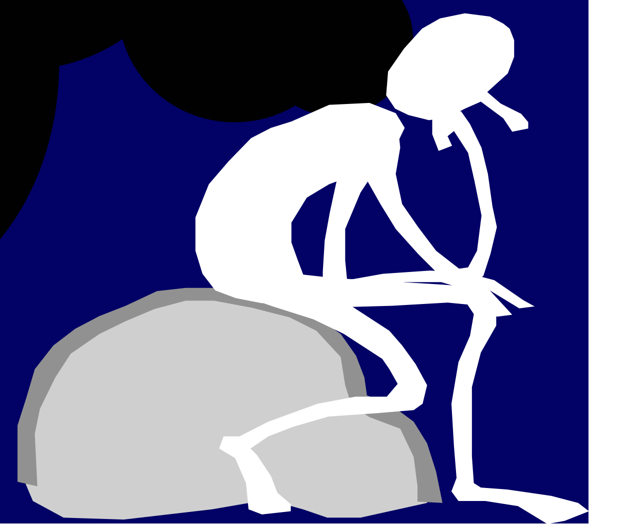
Theorem: 2^i is the largest number that can be made in i stages, and can only be made by repeated doubling

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.

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

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



5 < M(30)

Suppose that M(30)=5. At the last stage, we added two numbers x_1 and x_2 to get 30.

Without loss of generality (WLOG), we assume that $x_1 \geq x_2$.

Thus, $x_1 \geq 15$

By doubling bound, $x_1 \leq 16$

But x_1 can't be 16 since there is only one way to make 16 in 4 stages and it does not make 14 along the way.

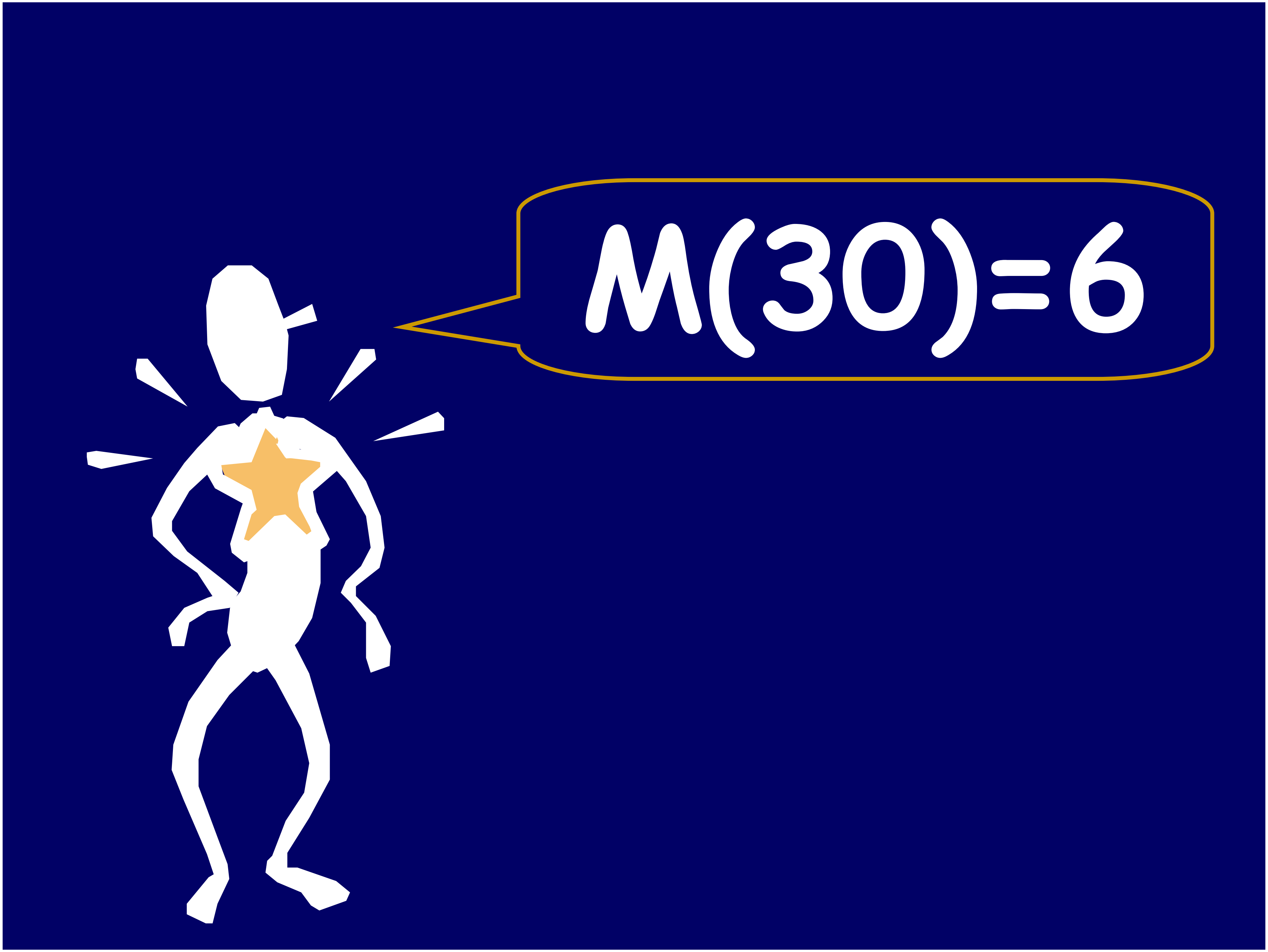
Thus, $x_1 = 15$ and M(15)=4

Suppose M(15) = 4

At stage 3, a number bigger than 7.5, but not more than 8 must have existed. There is only one sequence that gets 8 in 3 additions: 1 2 4 8

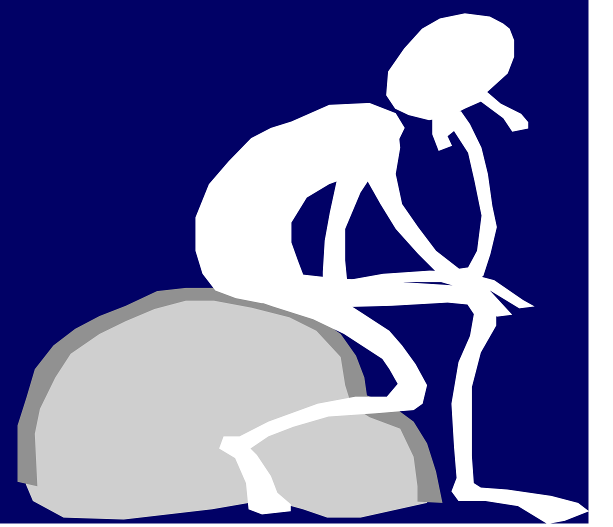
That sequence does not make 7 along the way and hence there is nothing to add to 8 to make 15 at the next stage.

Thus, M(15) > 4.   CONTRADICTION.


M(30)=6

$$M(30) = 6$$

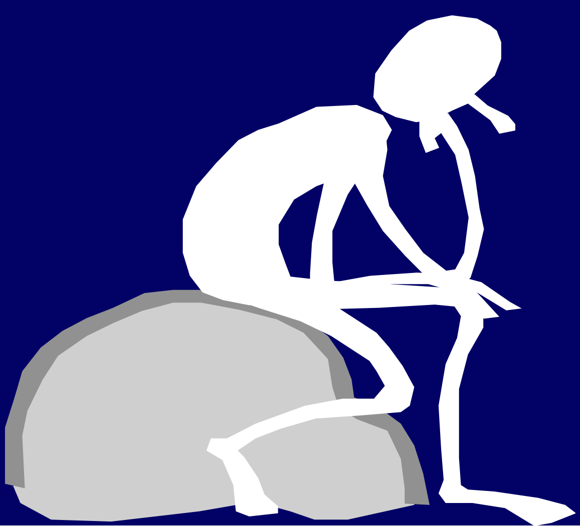
$$\log_2 n \leq M(n) \leq 2\lfloor \log_2 (n) \rfloor$$



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

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

Factoring Bound

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

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, every time the construction of b refers to a number x, use the number a times x.

Example

45 = 5 * 9

$M(5)=3$ [1 2 4 5]

$M(9)=4$ [1 2 4 8 9]

$M(45) \le 3+4$ [1 2 4 5 10 20 40 45]

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

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

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 125-126

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: **12,509**
[1 2 4 8 16 17 32 64 128 256 512 1024 1041 2082 4164 8328 8345 12509]

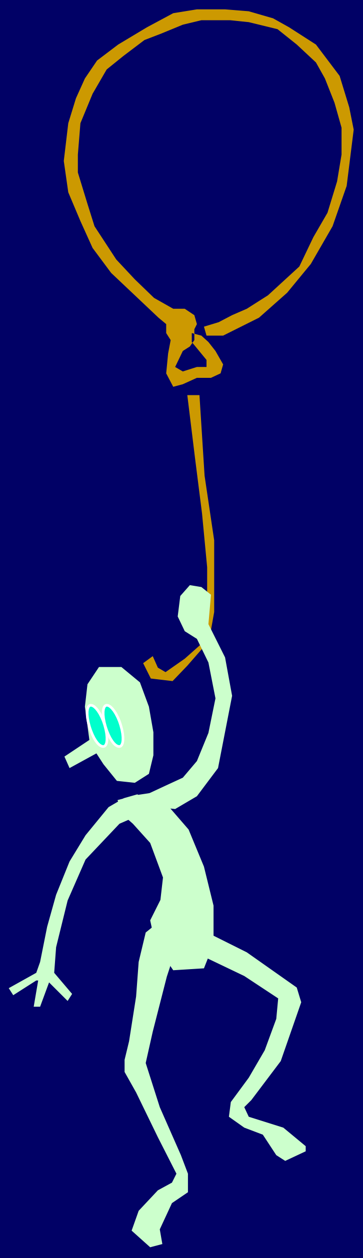
Open Problem

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

Raising To A Power
Minimal Addition Chain
Lower and Upper Bounds

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

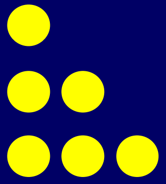
Study Bee

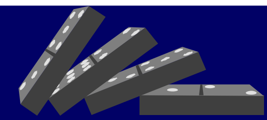
Representation:
Understand the relationship between different representations of the same information or idea

1

2

3



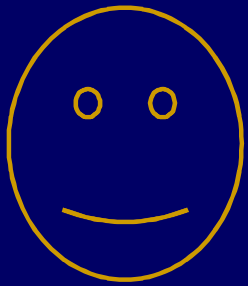


Induction has many guises.
Master their interrelationship.

- Formal Arguments
- Loop Invariants
- Recursion
- Algorithm Design
- Recurrences

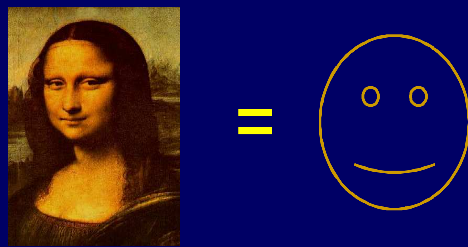
Abstraction:
Abstract away the inessential features of a problem or solution

=



Exemplification:
Try out a problem or solution on small examples.

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



Solution

GENERALIZE

Study Bee

REFERENCES

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