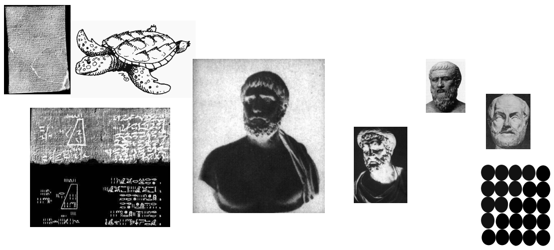
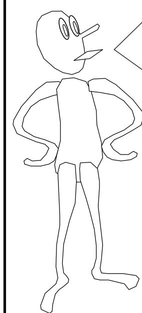
Great Theoretical Ideas In Computer Science

Steven Rudich — CS 15-251 — Spring 2005

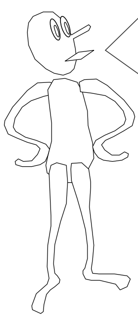
Lecture 14 — Feb 24, 2005 — Carnegie Mellon University

# Choose Your Representation!





We have seen that the same idea can be represented in many different ways.



Natural Numbers

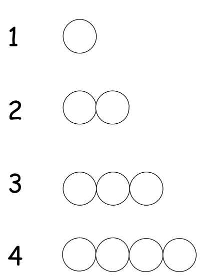
Unary
Binary
Base b

## Mathematical Prehistory: 30,000 BC

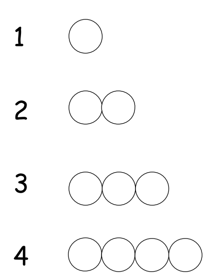
Paleolithic peoples in Europe record unary numbers on bones.

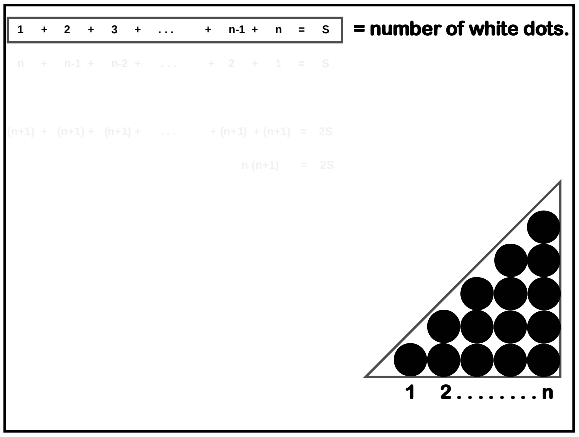
1 represented by 1 mark
2 represented by 2 marks
3 represented by 3 marks
4 represented by 4 marks
…

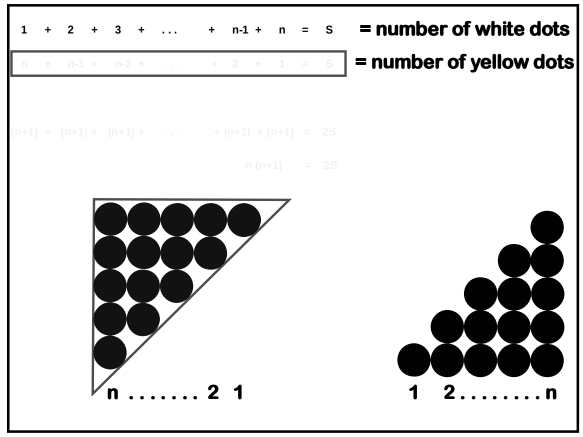
## Prehistoric Unary

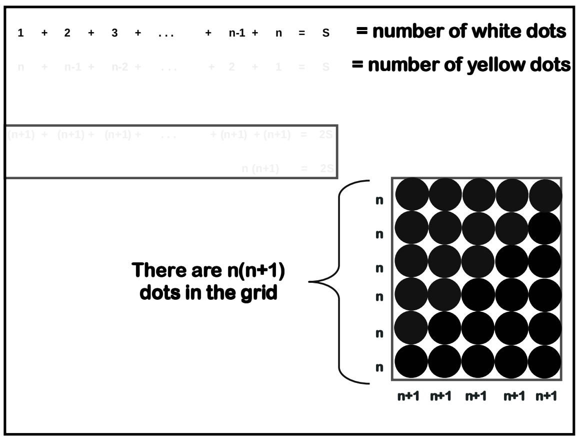


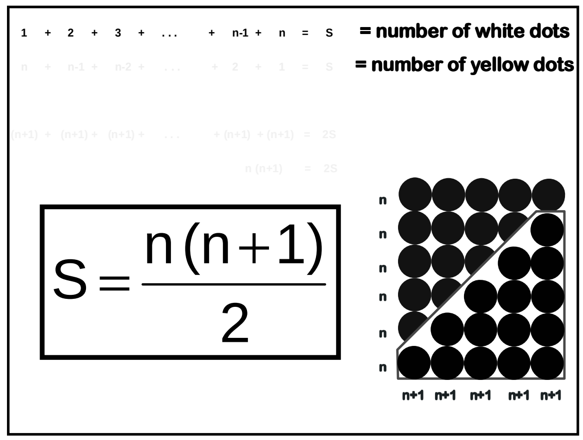

## PowerPoint Unary

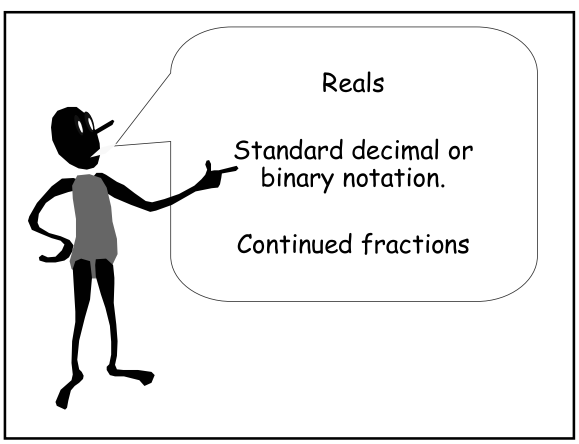




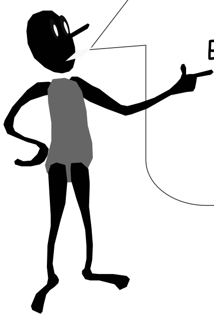



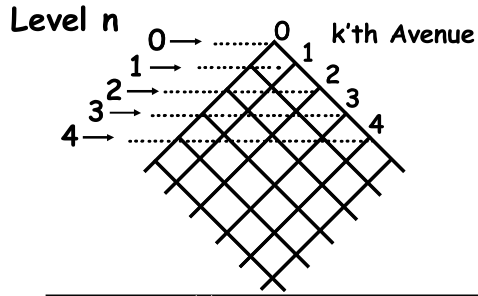

## A Periodic CF

$$\frac{3+\sqrt{13}}{2} = 3+\cfrac{1}{3+\cfrac{1}{3+\cfrac{1}{3+\cfrac{1}{3+\cfrac{1}{3+\cfrac{1}{3+\cfrac{1}{3+\cfrac{1}{3+\cfrac{1}{3+....}}}}}}}}}$$

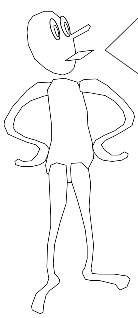


Choice Trees
Block Walking Model
Binomial Expansion
Vectors

## Manhattan

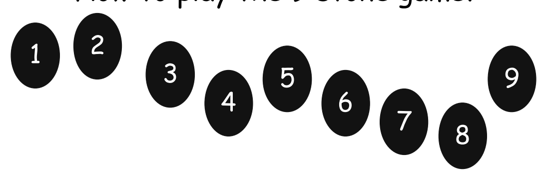



There are $\binom{n}{k}$ shortest routes from (0,0) to Level $n$ and $k^{th}$ Avenue.

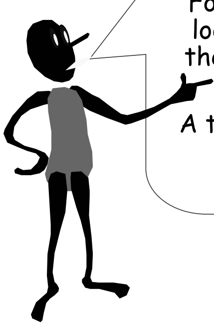


Multiple Representations means that we have our CHOICE of which one we will use.

## How to play the 9 stone game?



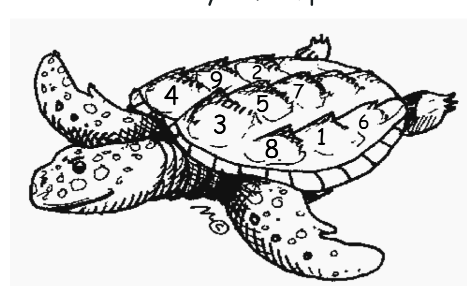

9 stones, numbered 1-9
Two players alternate moves.
Each move a player gets to take a new stone
Any subset of 3 stones adding to 15, wins.



For enlightenment, let's look to ancient China in the days of Emperor Yu.

A tortoise emerged from the river Lo…

Magic Square: Brought to humanity on the back of a tortoise from the river Lo in the days of Emperor Yu



Magic Square: Any 3 in a vertical, horizontal, or diagonal line add up to 15.

| | | |
|---|---|---|
| 4 | 9 | 2 |
| 3 | 5 | 7 |
| 8 | 1 | 6 |

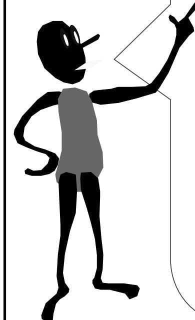
Conversely, any 3 that add to 15 must be on a line.

| | | |
|---|---|---|
| 4 | 9 | 2 |
| 3 | 5 | 7 |
| 8 | 1 | 6 |

TIC-TAC-TOE on a Magic Square Represents The Nine Stone Game

Alternate taking squares 1-9. Get 3 in a row to win.

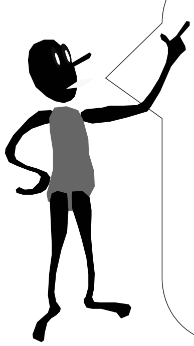
| | | |
|---|---|---|
| 4 | 9 | 2 |
| 3 | 5 | 7 |
| 8 | 1 | 6 |



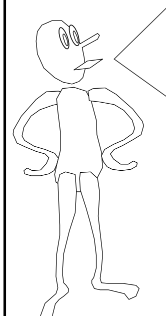
BIG IDEA!

Don't stick with the representation in which you encounter problems!

Always seek the more useful one!



This IDEA takes practice, practice, practice to understand and use.



<u>Natural Numbers</u>

Unary

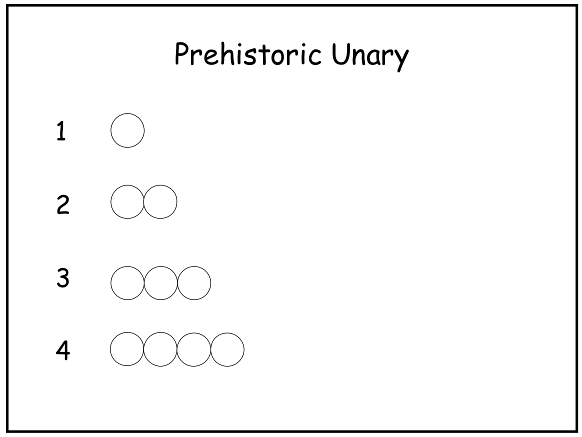
Binary

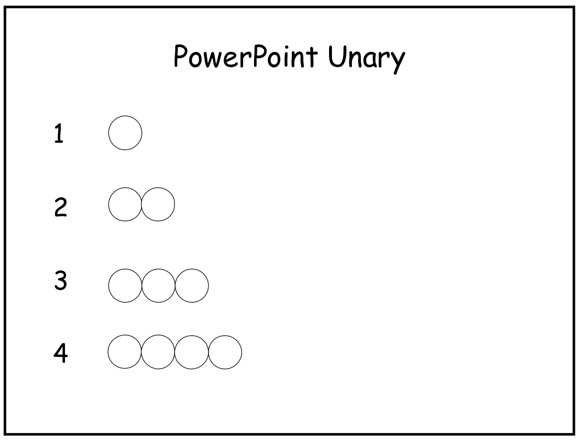
Base b

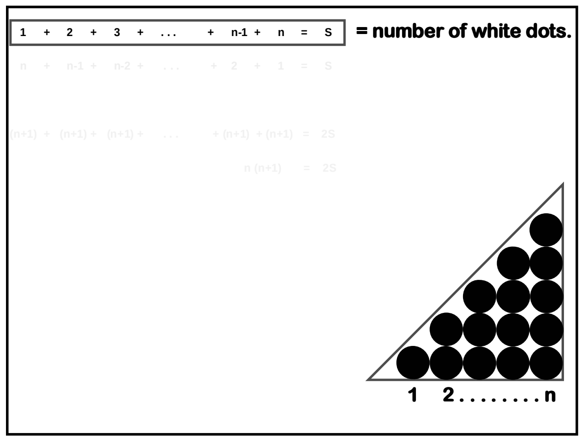
Mathematical Prehistory:
30,000 BC

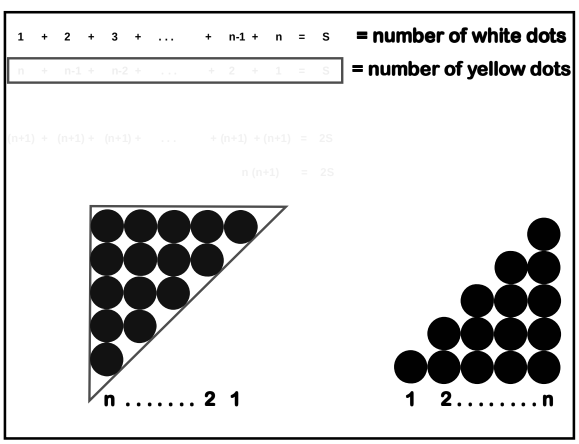
Paleolithic peoples in Europe record unary numbers on bones.

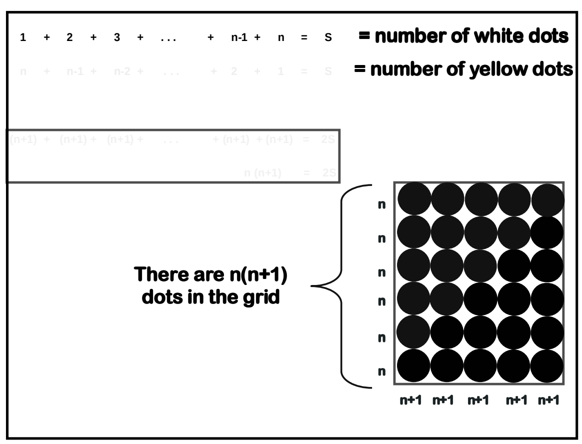
1 represented by 1 mark
2 represented by 2 marks
3 represented by 3 marks
4 represented by 4 marks
...



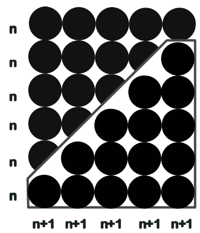
1 + 2 + 3 + ... + n-1 + n = S = number of white dots

n + n-1 + n-2 + ... + 2 + 1 = S = number of yellow dots

(n+1) + (n+1) + (n+1) + ... + (n+1) + (n+1) = 2S

n (n+1) = 2S

$$S = \frac{n(n+1)}{2}$$




A case study.

Anagram Programming Task.

You are given a 70,000 word dictionary. Write an anagram utility that given a word as input returns all anagrams of that word appearing in the dictionary.

Examples

Input: CAT
Output: ACT, CAT, TAC

Input: SUBESSENTIAL
Output: SUITABLENESS

Impatient Hacker
(Novice Level Solution)

Loop through all possible ways of rearranging the input word

Use binary search to look up resulting word in dictionary.

If found, output it

Performance Analysis
Counting without executing

On the word "microphotographic", we loop $17! \approx 3 * 10^{14}$ times.

Even at 1 microsecond per iteration, this will take $3 * 10^8$ seconds.

Almost a decade!

(There are about $\pi$ seconds in a nanocentury.)

"Expert" Hacker

Module ANAGRAM(X,Y) returns TRUE exactly when X and Y are anagrams.
(Works by sorting the letters in X and Y)

Input X
Loop through all dictionary words Y
If ANAGRAM(X,Y) output Y

The hacker is satisfied
and reflects no futher

Comparing an input word with each of 70,000 dictionary entries takes about 15 seconds.

The master keeps trying
to <u>refine</u> the solution.

The master's program runs in less than 1/1000 seconds.

Master Solution

Don't keep the dictionary in sorted order!

Rearranging the dictionary into anagram classes will make the original problem simple.

Suppose the dictionary was the list below.

ASP
DOG
LURE
GOD
NICE
RULE
SPA

After each word, write its "signature" (sort its letters)

| | |
|---|---|
| ASP | APS |
| DOG | DGO |
| LURE | ELRU |
| GOD | DGO |
| NICE | CEIN |
| RULE | ELRU |
| SPA | APS |

Sort by the signatures

| | |
|---|---|
| ASP | APS |
| SPA | APS |
| NICE | CEIN |
| DOG | DGO |
| GOD | DGO |
| LURE | ELRU |
| RULE | ELRU |

## Master Program

Input word W

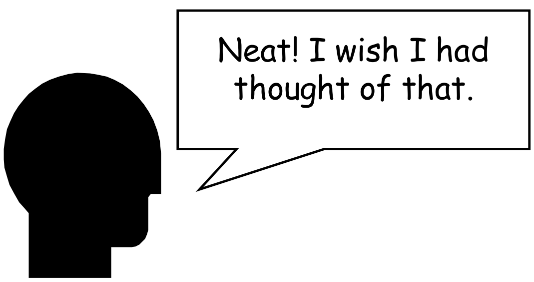
X := signature of W

Use binary search to find the anagram class of W and output it.

About $\log_2(70{,}000) \times 25$ microseconds

$\approx$ .0004 seconds

Of course, it takes about 30 seconds to create the dictionary, but it is perfectly fair to think of this as programming time. The building of the dictionary is a one-time cost that is part of writing the program.

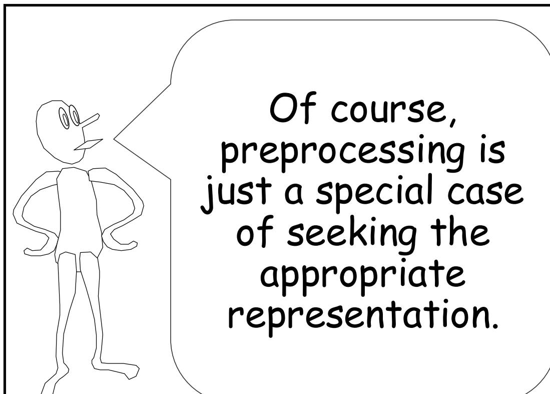


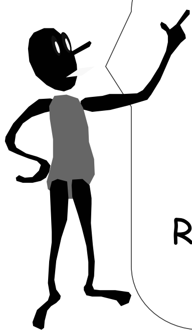

## Name Your Tools

Whenever you see something you wish you had thought of, try and formulate the minimal and most general lesson that will insure that you will not miss the same thing the next time. Name the lesson to make it easy to remember.

## NAME: <u>Preprocessing</u>

It is sometimes possible to pay a reasonable, one-time preprocessing cost to reorganize your data in such a way as to use it more efficiently later. The extra time required to preprocess can be thought of as additional programming effort.





Don't let the representation choose you,

CHOOSE THE REPRESENTATION!

## Vector Programs

Let's define a (parallel) programming language called VECTOR that operates on possibly infinite vectors of numbers. Each variable $\vec{V}$ can be thought of as:

< * , * , * , * , *, *, . . . . . . . . . >

0 1 2 3 4 5 . . . . . . . . .

## Vector Programs

Let k stand for a scalar constant
<k> will stand for the vector <k,0,0,0,…>

<0> = <0,0,0,0,….>
<1> = <1,0,0,0,…>

$\vec{V}+\vec{T}$ means to add the vectors position-wise.

<4,2,3,…> + <5,1,1,….> = <9,3,4,…>

## Vector Programs

RIGHT($\vec{V}$) means to shift every number in $\vec{V}$ one position to the right and to place a 0 in position 0.

RIGHT( <1,2,3, …> ) = <0,1,2,3,. …>

## Vector Programs

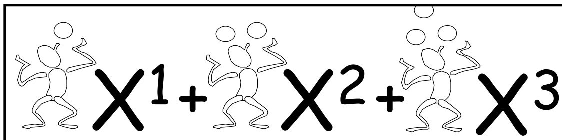
| Example: | Stare |
|---|---|
| $\vec{V}$ := <6>; | $\vec{V}$ = <6,0,0,0,..> |
| $\vec{V}$ := RIGHT($\vec{V}$) + <42>; | $\vec{V}$ = <42,6,0,0,..> |
| $\vec{V}$:= RIGHT($\vec{V}$) + <2>; | $\vec{V}$ = <2,42,6,0,..> |
| $\vec{V}$ := RIGHT($\vec{V}$) + <13>; | $\vec{V}$= <13,2,42,6,.> |

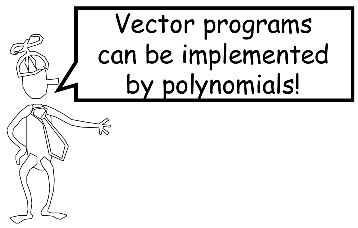
$\vec{V}$ = < 13, 2, 42, 6, 0, 0, 0, . . . >

## Vector Programs

| Example: | Stare |
|---|---|
| $\vec{V}$ := <1>; | $\vec{V}$ = <1,0,0,0,..> |
| Loop n times: | $\vec{V}$ = <1,1,0,0,..> |
| $\vec{V}$ := $\vec{V}$ + RIGHT($\vec{V}$); | $\vec{V}$ = <1,2,1,0,..> |
| | $\vec{V}$= <1,3,3,1,.> |

$\vec{V}$ = $n^{th}$ row of Pascal's triangle.




## Programs -----> Polynomials

The vector $\vec{V} = \langle a_0, a_1, a_2, \ldots \rangle$ will be represented by the polynomial:

$$P_V = \sum_{i=0}^{i=\infty} a_i X^i$$

## Formal Power Series

The vector $\vec{V} = \langle a_0, a_1, a_2, \ldots \rangle$ will be represented by the formal power series:

$$P_V = \sum_{i=0}^{i=\infty} a_i X^i$$

$\vec{V} = \langle a_0, a_1, a_2, \ldots \rangle$

$$P_V = \sum_{i=0}^{i=\infty} a_i X^i$$

| | |
|---|---|
| $\langle 0 \rangle$ is represented by | 0 |
| $\langle k \rangle$ is represented by | k |
| $\vec{V} + \vec{T}$ is represented by | $(P_V + P_T)$ |
| RIGHT($\vec{V}$) is represented by | $(P_V X)$ |

## Vector Programs

Example:

$\vec{V} := \langle 1 \rangle;$ $\quad P_V := 1;$

Loop n times:
$\vec{V} := \vec{V} + \text{RIGHT}(\vec{V});$ $\quad P_V := P_V + P_V X;$

$\vec{V}$ = $n^{th}$ row of Pascal's triangle.

## Vector Programs

Example:

$\vec{V} := \langle 1 \rangle;$ $\quad P_V := 1;$

Loop n times:
$\vec{V} := \vec{V} + \text{RIGHT}(\vec{V});$ $\quad P_V := P_V (1 + X);$

$\vec{V}$ = $n^{th}$ row of Pascal's triangle.

## Vector Programs

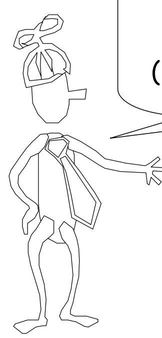
Example:

$V^{\rightarrow}$ := <1>;

Loop n times:
$V^{\rightarrow} := V^{\rightarrow} + \text{RIGHT}(V^{\rightarrow})$;

$P_V = (1+X)^n$

$V^{\rightarrow}$ = $n^{th}$ row of Pascal's triangle.

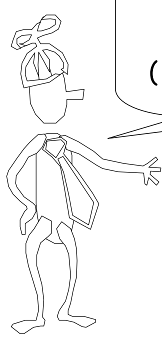


What is the coefficient of $X^k$ in the expansion of:

$( 1 + X + X^2 + X^3 + X^4 + \ldots )^n$ ?

Each path in the choice tree for the cross terms has n choices of exponent $e_1, e_2, \ldots, e_n \geq 0$. Each exponent can be any natural number.

Coefficient of $X^k$ is the number of non-negative solutions to:
$e_1 + e_2 + \ldots + e_n = k$



What is the coefficient of $X^k$ in the expansion of:

$( 1 + X + X^2 + X^3 + X^4 + \ldots )^n$ ?

$$\binom{n+k-1}{n-1}$$

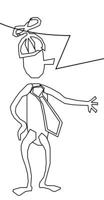
$( 1 + X + X^2 + X^3 + X^4 + \ldots )^n =$

$$\frac{1}{(1-X)^n} = \sum_{k=0}^{\infty} \binom{n+k-1}{n-1} X^k$$

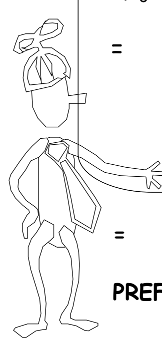
What is the coefficient of $X^k$ in the expansion of:

$(a_0 + a_1X + a_2X^2 + a_3X^3 + \ldots) ( 1 + X + X^2 + X^3 + \ldots )$

$= (a_0 + a_1X + a_2X^2 + a_3X^3 + \ldots) / (1 - X)$ ?



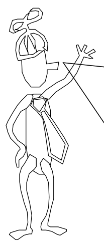
$a_0 + a_1 + a_2 + .. + a_k$



$(a_0 + a_1X + a_2X^2 + a_3X^3 + \ldots) / (1 - X)$

$$= \sum_{k=0}^{\infty} \left( \sum_{i=0}^{i=k} a_i \right) X^k$$

= **PREFIXSUM**$(a_0 + a_1X + a_2X^2 + a_3X^3 + \ldots)$



Let's add an instruction called PREFIXSUM to our VECTOR language.

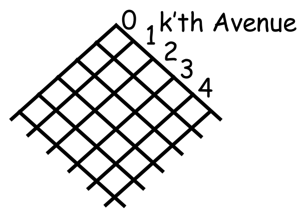
$W^{\rightarrow}$ := PREFIXSUM($V^{\rightarrow}$)

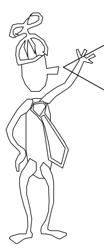
means that the $i^{th}$ position of W contains the sum of all the numbers in V from positions 0 to i.

What does this program output?

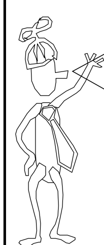
$V^{\rightarrow}$ := $1^{\rightarrow}$ ;

Loop k times: $V^{\rightarrow}$ := PREFIXSUM($V^{\rightarrow}$) ;



Can you see how PREFIXSUM can be represented by a familiar polynomial expression?



$W^{\rightarrow}$ := PREFIXSUM($V^{\rightarrow}$)

is represented by

$P_W = P_V / (1-X)$

$= P_V (1+X+X^2+X^3+ \ldots)$

## Al-Karaji Program

Zero_Ave := PREFIXSUM(<1>);

First_Ave := PREFIXSUM(Zero_Ave);

Second_Ave :=PREFIXSUM(First_Ave);

Output:=

First_Ave + 2*RIGHT(Second_Ave)

$OUTPUT^{\rightarrow}$ = <1, 4, 9, 25, 36, 49, …. >

## Al-Karaji Program

Zero_Ave = $1/(1-X)$;

First_Ave = $1/(1-X)^2$;

Second_Ave = $1/(1-X)^3$;

Output =

$1/(1-X)^2 + 2X/(1-X)^3$

$(1-X)/(1-X)^3 + 2X/(1-X)^3$

$= (1+X)/(1-X)^3$

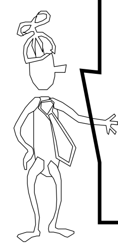
## $(1+X)/(1-X)^3$

```
Zero_Ave    := PREFIXSUM(<1>);
First_Ave   := PREFIXSUM(Zero_Ave);
Second_Ave :=PREFIXSUM(First_Ave);

Output:=
     RIGHT(Second_Ave) + Second_Ave
```

Second_Ave = <1, 3, 6, 10, 15,.
RIGHT(Second_Ave) = <0, 1, 3, 6, 10,.
Output = <1, 4, 9, 16, 25

---

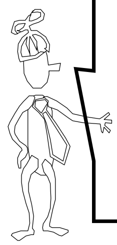


$(1+X)/(1-X)^3$ outputs <1, 4, 9, ..>

$X(1+X)/(1-X)^3$ outputs <0, 1, 4, 9, ..>

The $k^{th}$ entry is $k^2$

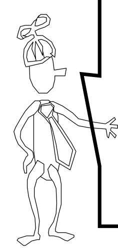
---



$X(1+X)/(1-X)^3 = \sum k^2 X^k$

What does $X(1+X)/(1-X)^4$ do?

---

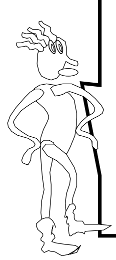


$X(1+X)/(1-X)^4$ expands to :

$\sum S_k X^k$

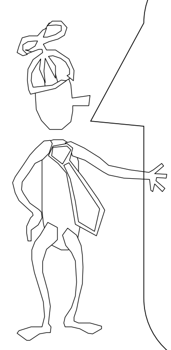
where $S_k$ is the sum of the first k squares

---



Aha! Thus, if there is an alternative interpretation of the $k^{th}$ coefficient of $X(1+X)/(1-X)^4$ we would have a new way to get a formula for the sum of the first k squares.

---



Using pirates and gold we found that:

$$\frac{1}{(1-X)^n} = \sum_{k=0}^{\infty} \binom{n+k-1}{n-1} X^k$$

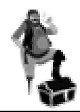
**THUS:**

$$\frac{1}{(1-X)^4} = \sum_{k=0}^{\infty} \binom{k+3}{3} X^k$$

Coefficient of $X^k$ in $P_V = (X^2+X)(1-X)^{-4}$ is the sum of the first k squares:

$$\frac{X^2+X}{(1-X)^4} = (X^2+X)\sum_{k=0}^{\infty}\binom{k+3}{3}X^k$$

$$= \sum_{k=0}^{\infty}\left(\binom{k+2}{3}+\binom{k+1}{3}\right)X^k$$



$$\frac{1}{(1-X)^4} = \sum_{k=0}^{\infty}\binom{k+3}{3}X^k$$

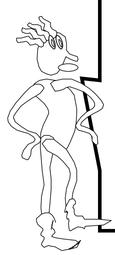
---

Vector programs -> Polynomials
-> Closed form expression

$$\frac{X^2+X}{(1-X)^4} = \sum_{k=0}^{\infty}\left(\binom{k+2}{3}+\binom{k+1}{3}\right)X^k$$

$$\sum_{i=0}^{i=n} i^2 = \binom{n+2}{3}+\binom{n+1}{3}$$

---



Expressions of the form

Finite Polynomial / Finite Polynomial

are called <u>Rational Polynomial Expressions</u>.

Clearly, these expressions have some deeper interpretation as a programming language.

---

## References


*The Heritage of Thales*, by W. S. Anglin and F. Lambek

*The Book Of Numbers*, by J. Conway and R. Guy

*Programming Pearls*, by J. Bentley

*History of Mathematics, Histories of Problems,* by The Inter-IREM Commission