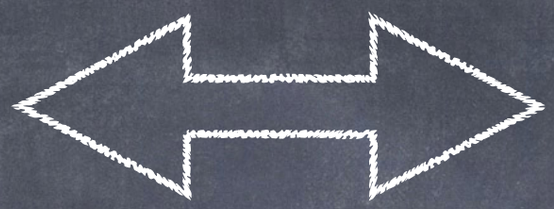
How to convince ourselves that we are NOT stupid...

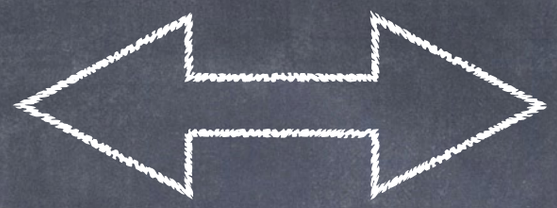
Prerona Chatterjee

STCS @ TIFR

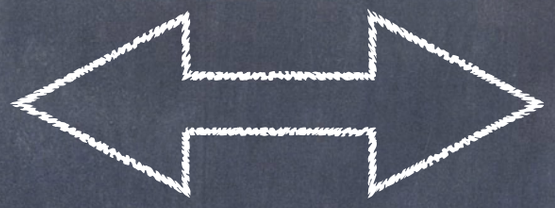
Connecting Cities



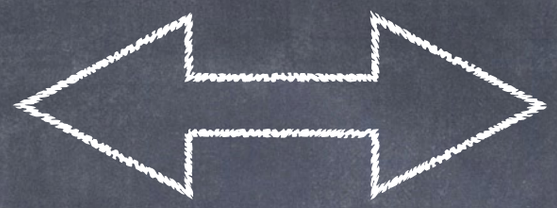
Volunteers



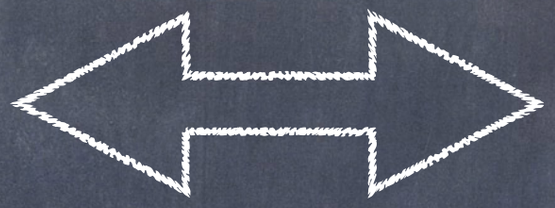
Volunteers



USB Cables

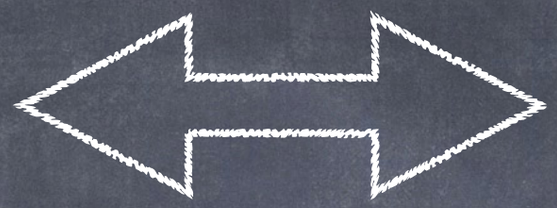


Volunteers



USB Cables

Every pair of cities
should be connected



Volunteers

USB Cables

Every pair of cities
should be connected

Are 5 cables enough?

NO

NO

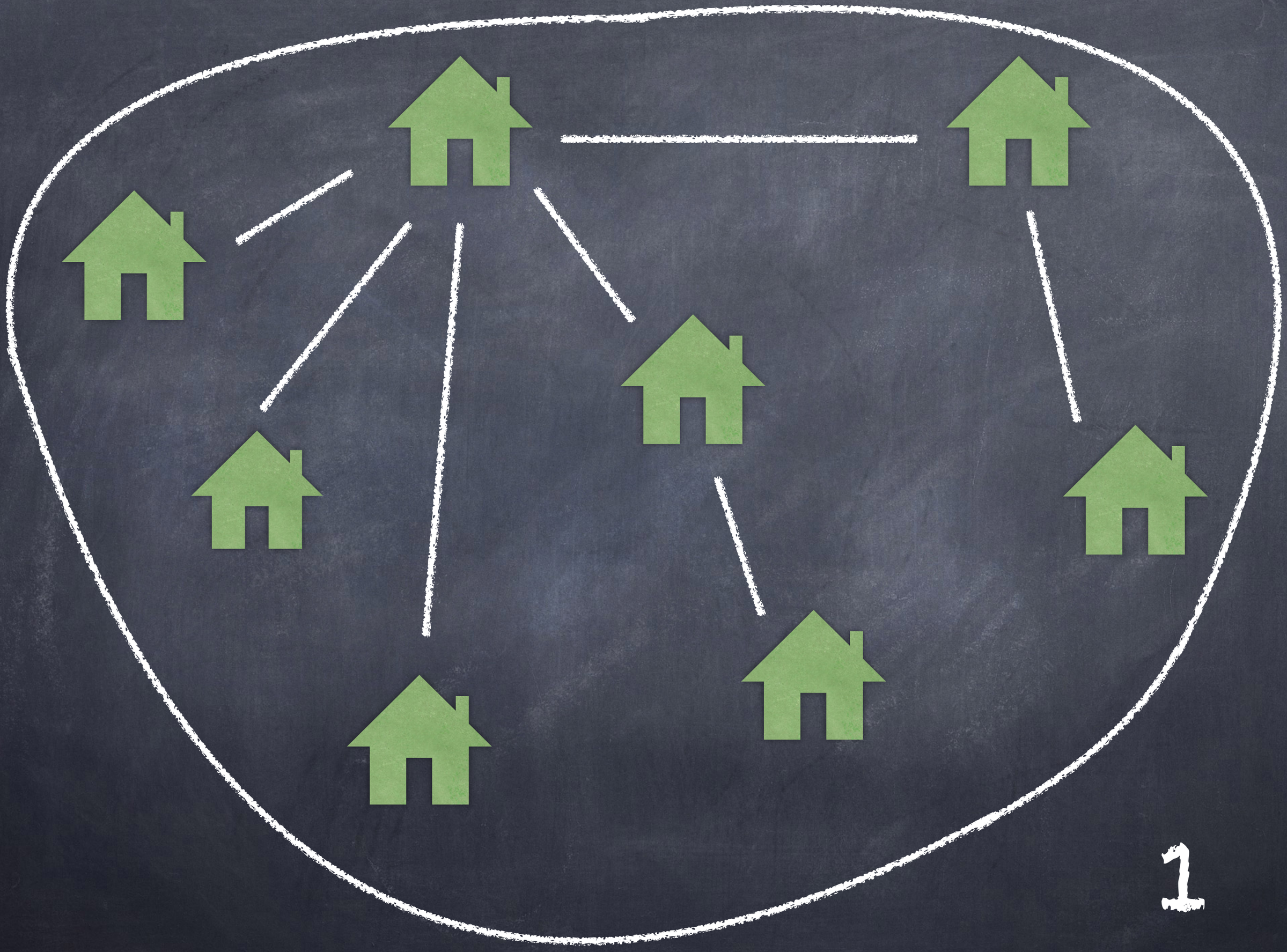
7 cables are required

NO

7 cables are required!

But why?

7



Proof

No. of groups @ the beginning = n

No. of groups in the end = 1

Proof

No. of groups @ the beginning = n

No. of groups in the end = 1

At each step the no. of groups reduce by at most 1

Proof

No. of groups @ the beginning = n

No. of groups in the end = 1

At each step the no. of groups reduce by at most 1

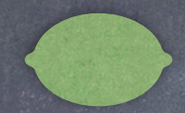
So, the no. of steps reqd. $\geq n-1$

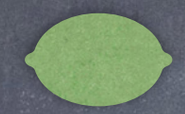
Sorting

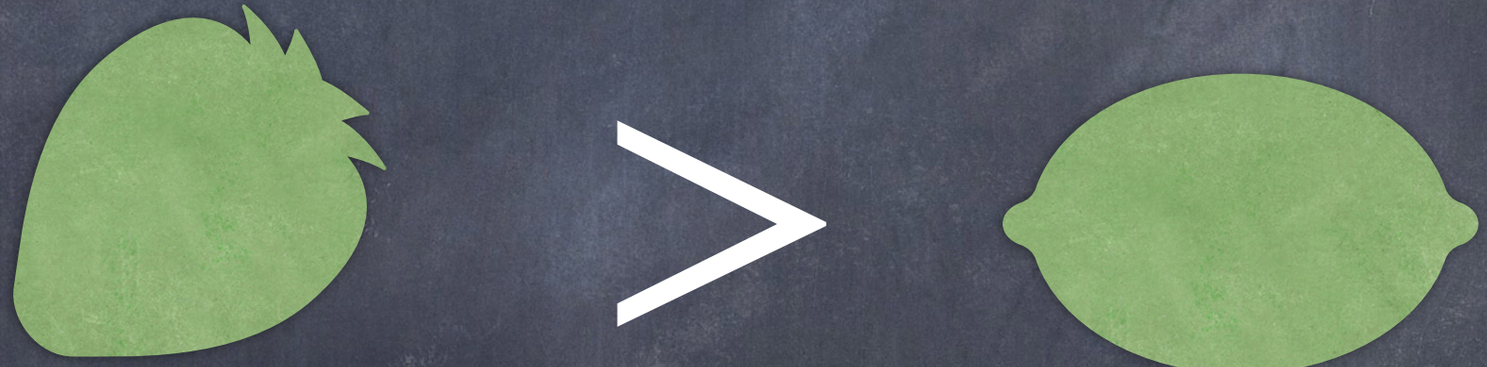
1	2	3	4	5	6	7	8	9	10
61	73	38	2	54	89	92	44	13	29

1	2	3	4	5	6	7	8	9	10
61	73	38	2	54	89	92	44	13	29

38 > 13

1	2	3	4	5	6	7	8	9	10
									

1	2	3	4	5	6	7	8	9	10
									

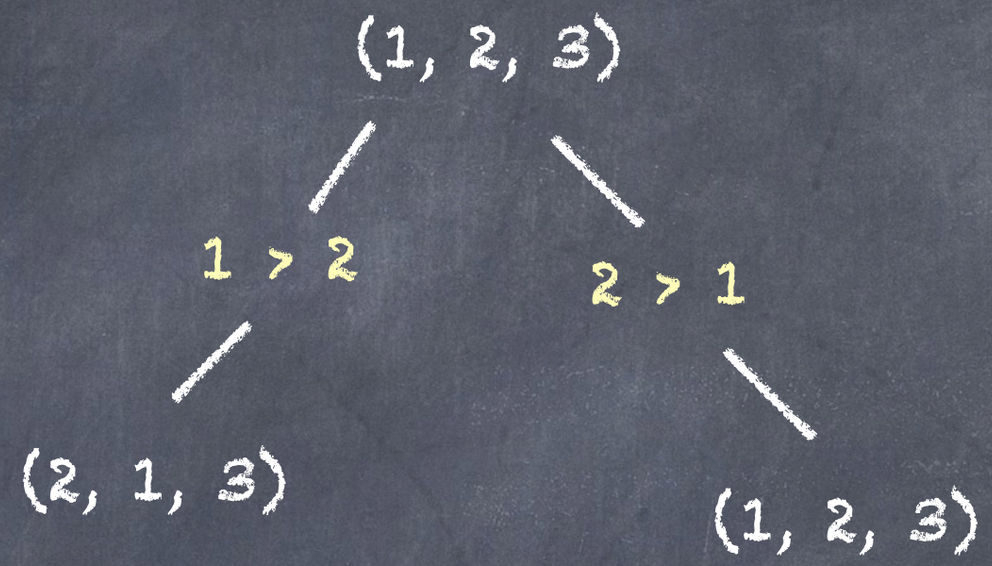


Bubble Sort

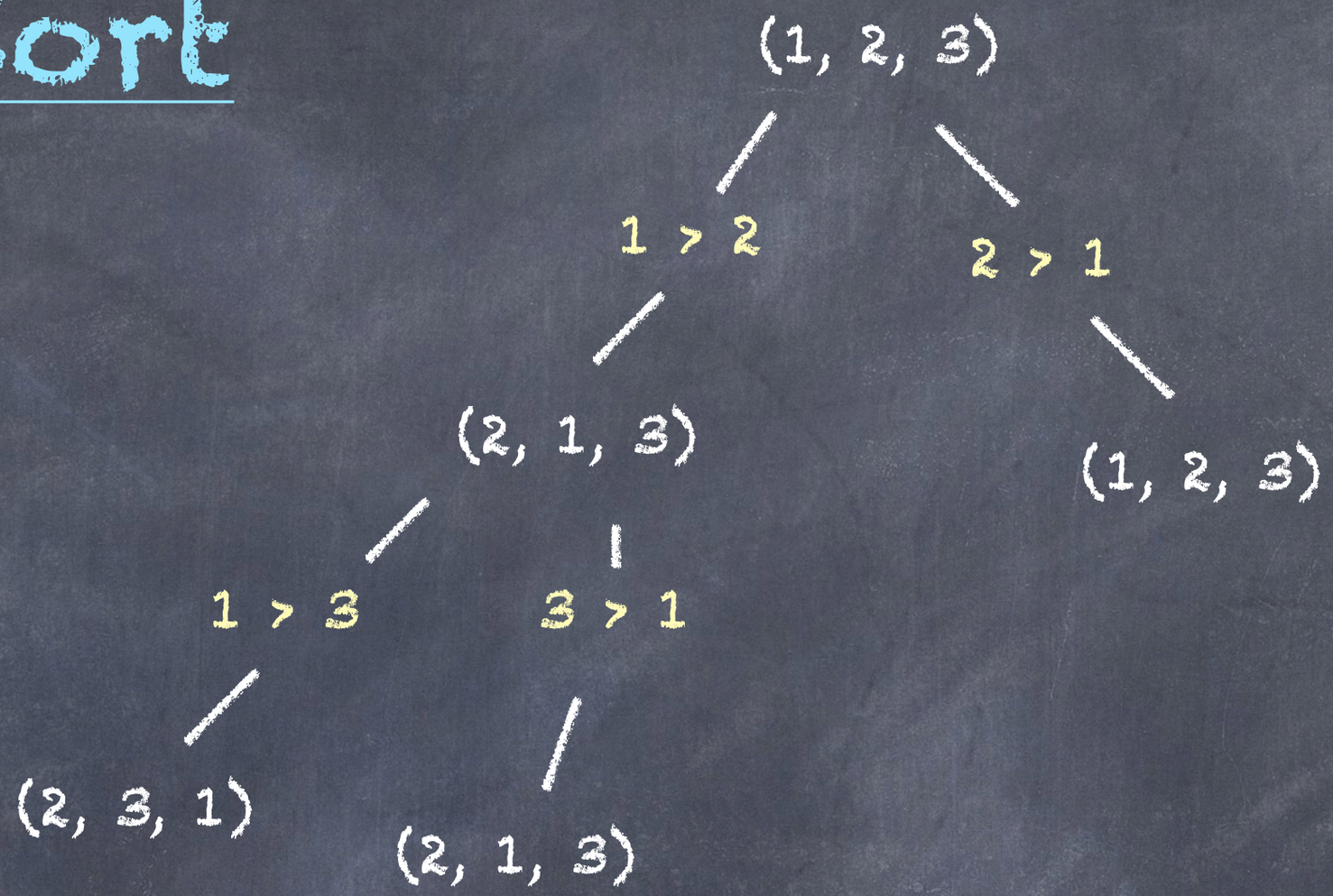
1	2	3
61	73	38

⟺ (1, 2, 3)

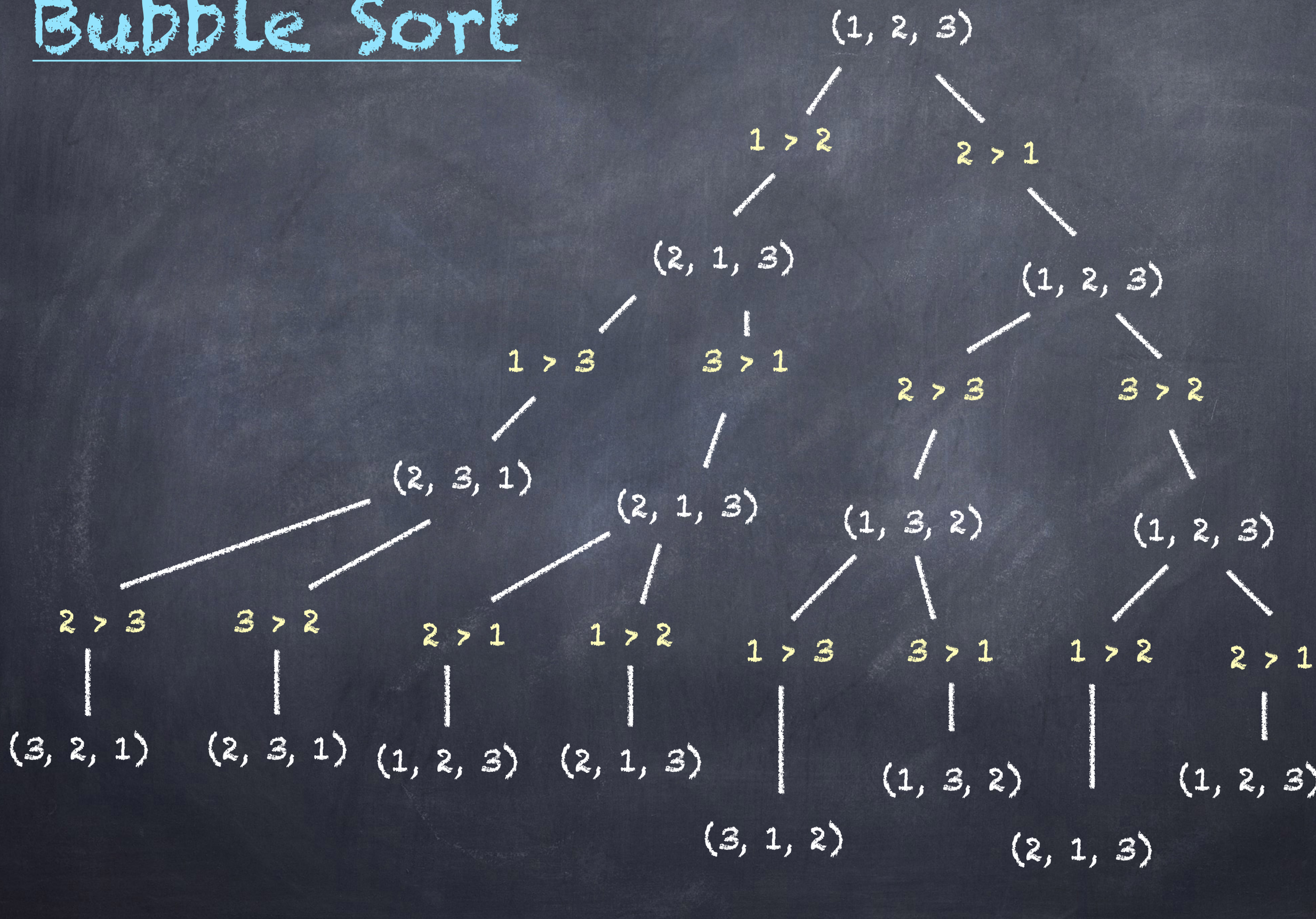
Bubble Sort



Bubble Sort

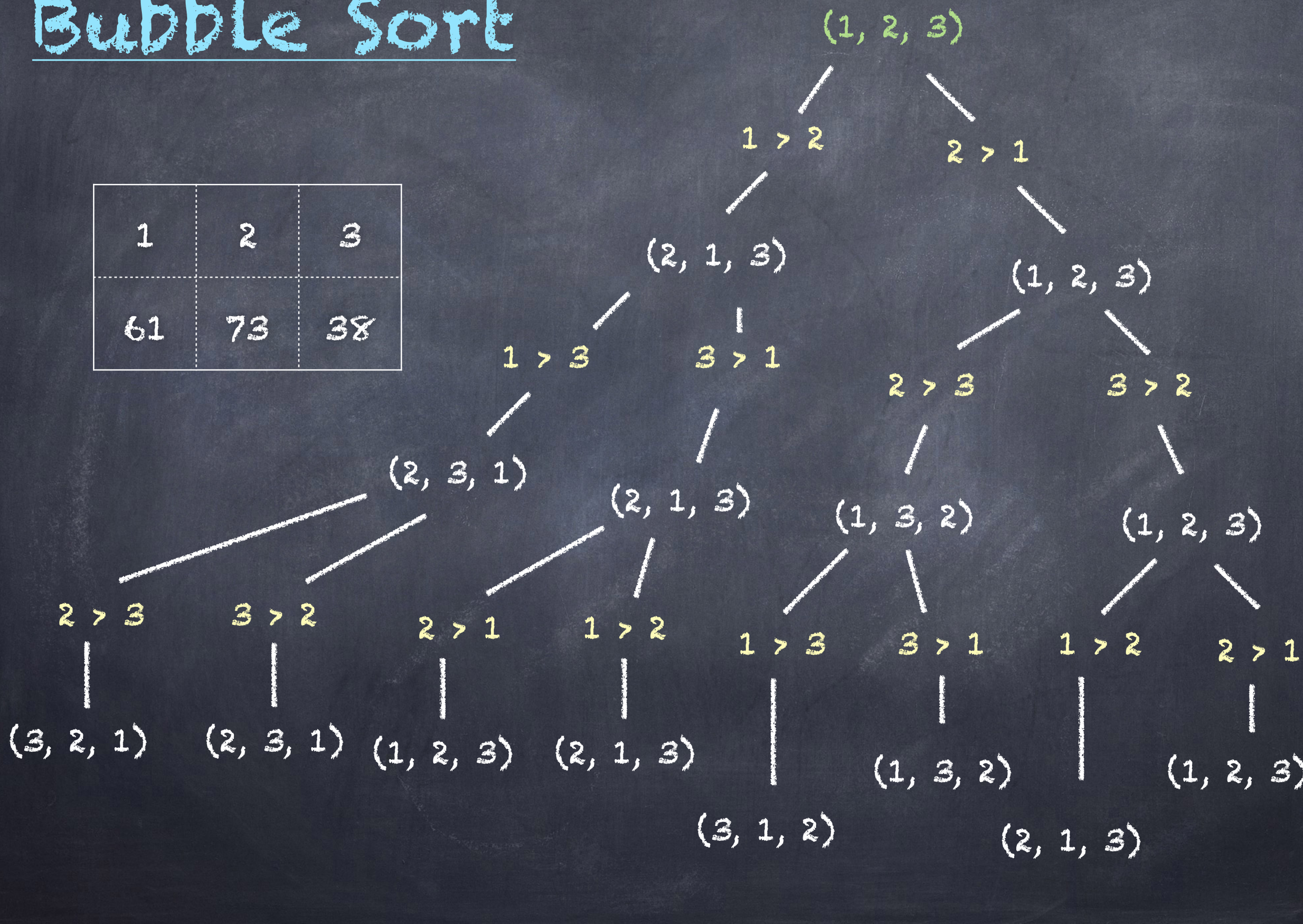


Bubble Sort


(1, 2, 3)
1 > 2
2 > 1
(2, 1, 3)
(1, 2, 3)
1 > 3
3 > 1
2 > 3
3 > 2
(2, 3, 1)
(2, 1, 3)
(1, 3, 2)
(1, 2, 3)
2 > 3
3 > 2
2 > 1
1 > 2
1 > 3
3 > 1
1 > 2
2 > 1
(3, 2, 1)
(2, 3, 1)
(1, 2, 3)
(2, 1, 3)
(3, 1, 2)
(1, 3, 2)
(2, 1, 3)
(1, 2, 3)

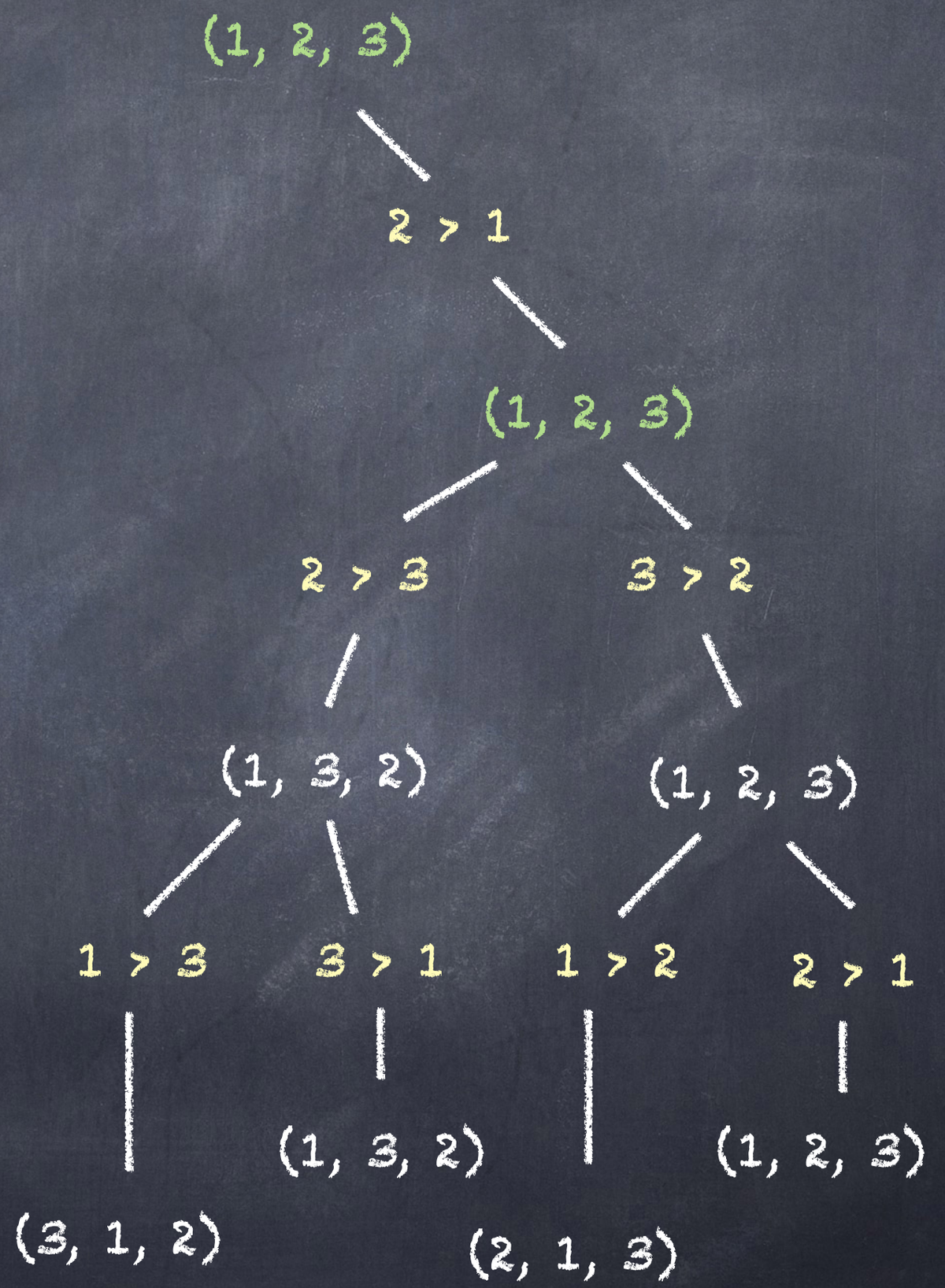
Bubble Sort

1	2	3
61	73	38



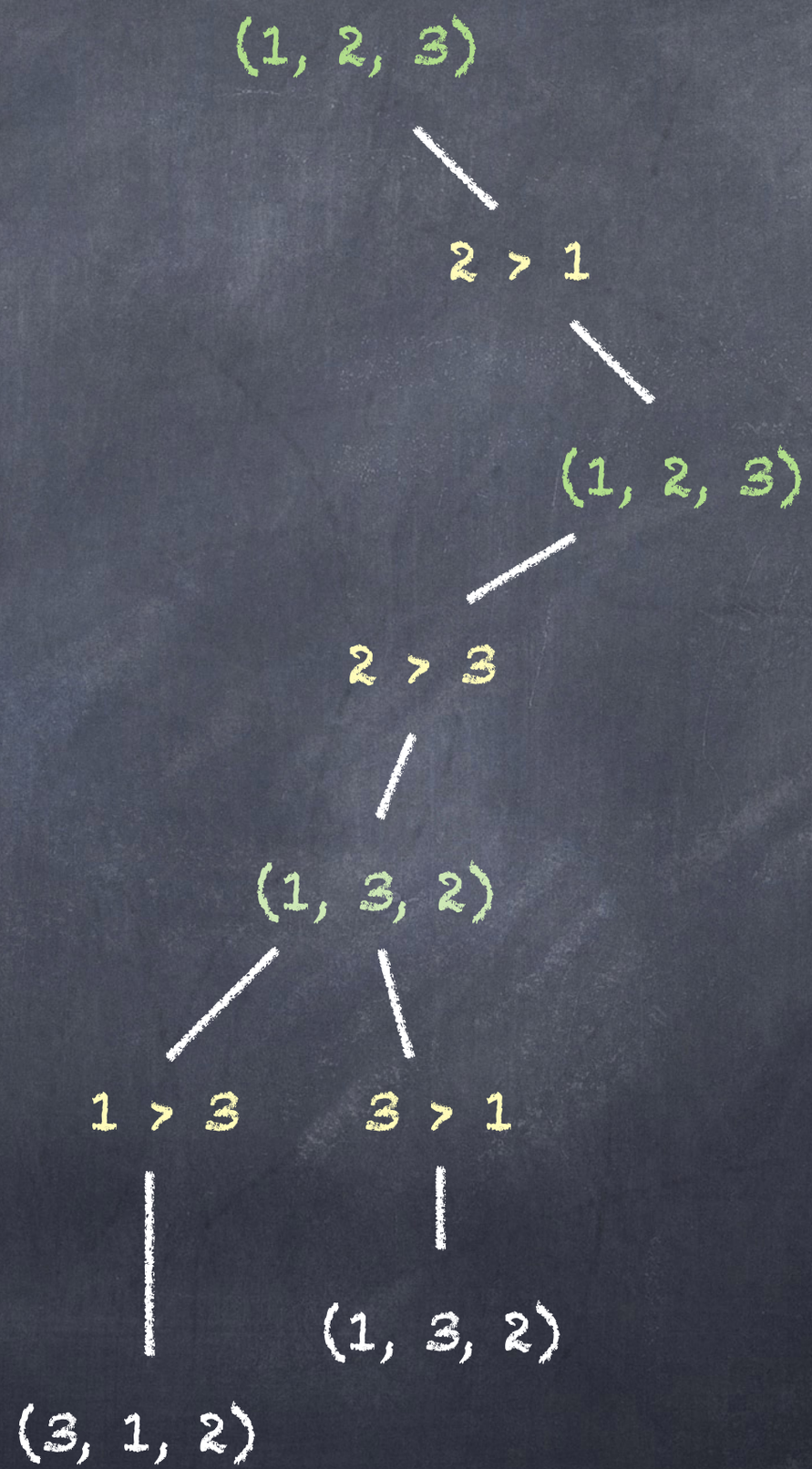
Bubble Sort

1	2	3
61	73	38



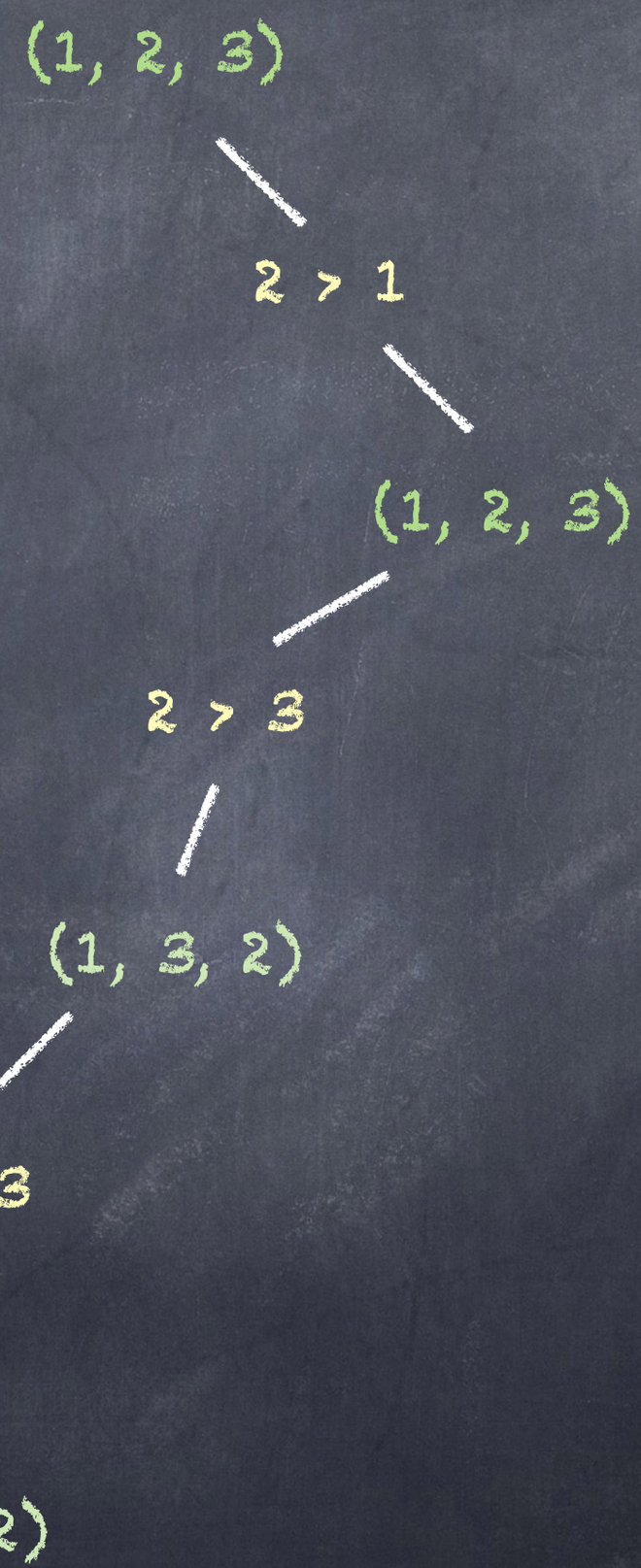
Bubble Sort

1	3	2
61	38	73



Bubble Sort

3	1	2
38	61	73

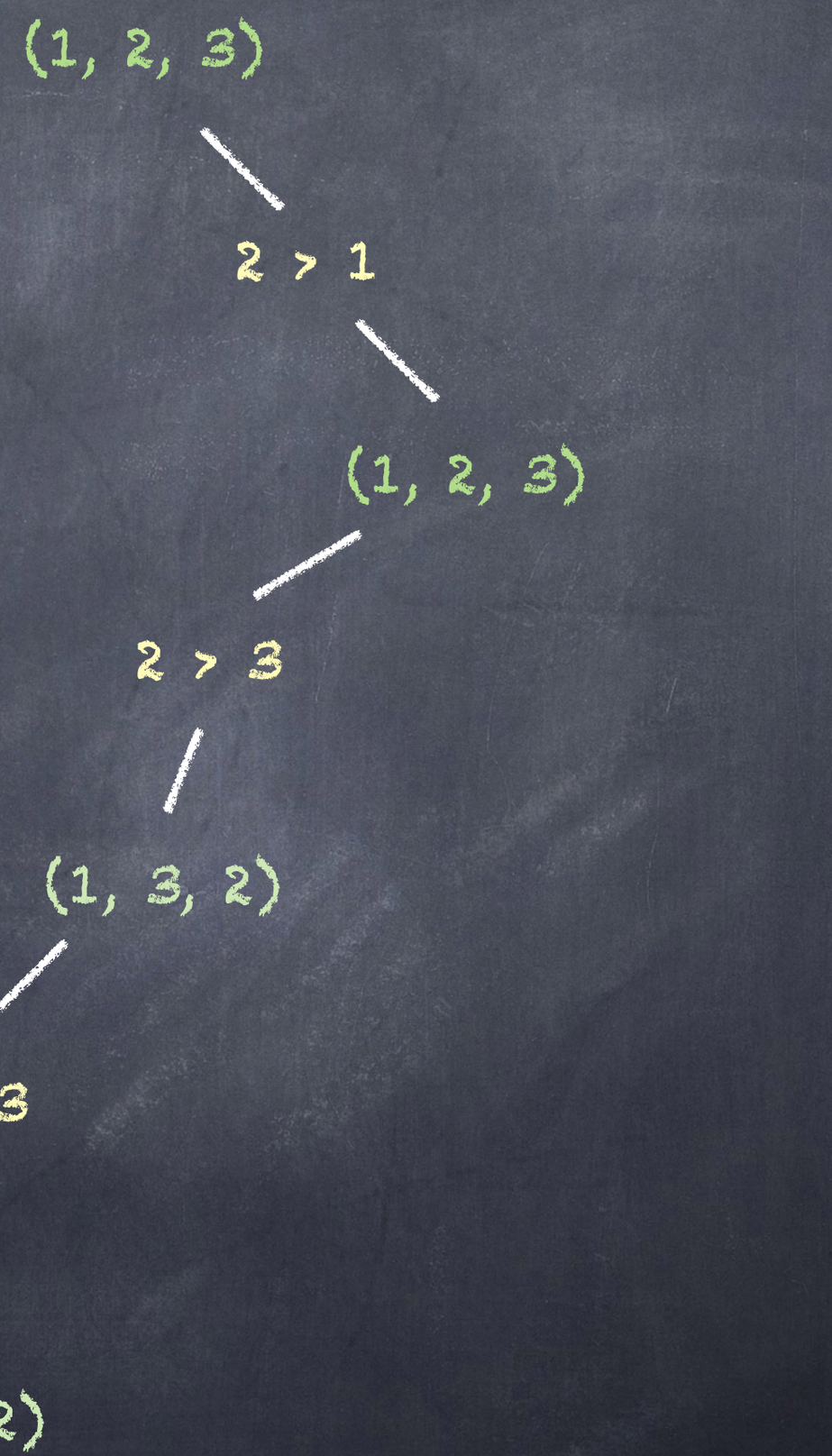


Bubble Sort

3	1	2
38	61	73

#comparisons

= 2 + 1 = 3



Bubble Sort

In general, the running time is:

$$(n-1)+(n-2)+\cdots+1=\frac{n(n-1)}{2}=\Theta(n^2)$$

Algorithms	Running Time
Bubble Sort	$O(n^2)$
Selection Sort	$O(n^2)$
Insertion Sort	$O(n^2)$
Merge Sort	$O(n \log n)$
Quick Sort	$O(n \log n)$
Heap Sort	$O(n \log n)$
.	
.	
.	

Can we do any better?

Can we do any better?

- Nothing is known about the kind of data
- Only comparisons allowed

Can we do any better?

- Nothing is known about the kind of data
- Only comparisons allowed

Nothing better is possible

Can we do any better?

- Nothing is known about the kind of data
- Only comparisons allowed

Nothing better is possible

But why?

A sorting algorithm

Binary tree

A sorting algorithm

Binary tree

Depth

No. of comparisons

A sorting algorithm

Binary tree

Depth

No. of comparisons

Leaf of the binary tree

Final answer

Correct permutation

Proof

Total number of correct permutations possible = n!

Proof

Total number of correct permutations possible = $n!$

$\leq$ the no. of leaves in the tree

Proof

Total number of correct permutations possible = $n!$

$\leq$ the no. of leaves in the tree

The no. of comparisons required = the depth of the tree

Proof

Total number of correct permutations possible = n!

≤ the no. of leaves in the tree

The no. of comparisons required = the depth of the tree

and this is AT LEAST log (n!) = $\Theta(n \log n)$

Matrix Multiplication

The Question:

How many multiplication operations are required?

The Question:

How many multiplication operations are required?

What is currently known:

The trivial algorithm takes $O(n^3)$

The Question:

How many multiplication operations are required?

What is currently known:

The trivial algorithm takes $O(n^3)$

Strassen showed it is $O(n^{\log_2 7}) = O(n^{2.807})$

The Question:

How many multiplication operations are required?

What is currently known:

The trivial algorithm takes $O(n^3)$

Strassen showed it is $O(n^{\log_2 7}) = O(n^{2.807})$

The current best algorithm takes $O(n^{2.3728639})$

The Question:

How many multiplication operations are required?

What is currently known:

The current best upper bound = $O(n^{2.3728639})$

The current best lower bound = $O(n^2)$

Computability
&
Complexity Theory

Problem

Easy?

Hard?

Decision Problems

L : Set of Strings

x : Input string

Question : Does x belong to L?

Decision Problems

L : Set of Strings

x : Input string

Question : Does x belong to L?

Computability:

Is there an algorithm to solve this?

Complexity Theory :

If it does have an algorithm, then how much resource does it require?

Complexity Theory :

If it does have an algorithm, then how much resource does it require?

Resources we are interested in:

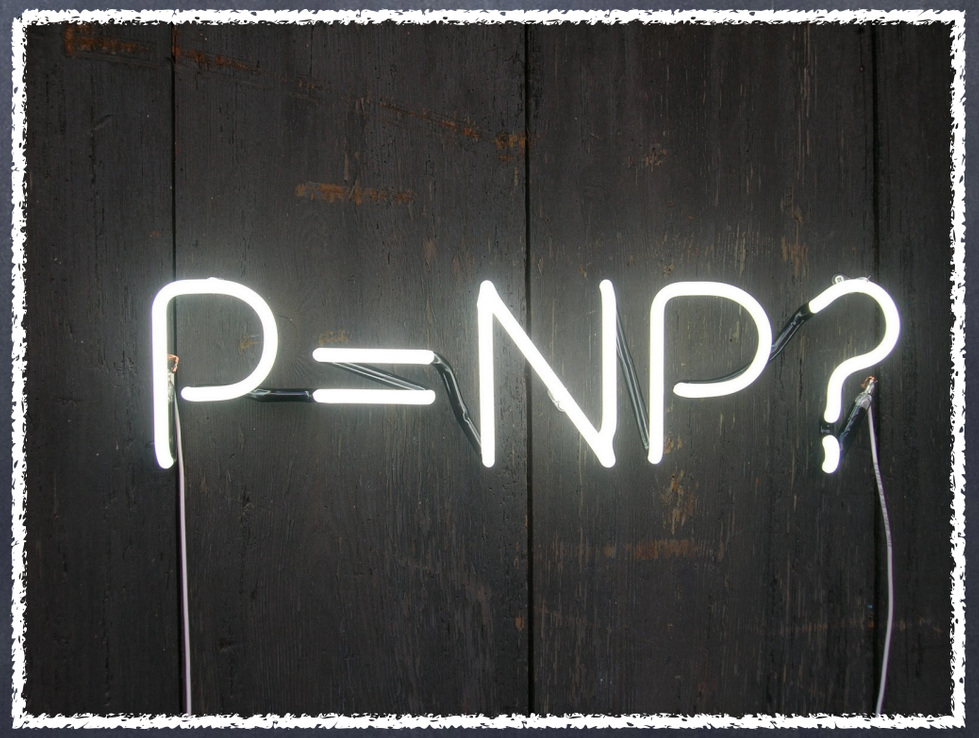
1. Time required
2. Space required

Complexity Theory :

If it does have an algorithm, then how much resource does it require?

Resources we are interested in:

1. Time required
2. Space required



P vs. NP

P: Polynomial time algorithms

P vs. NP

Input Length : n

P: Polynomial time algorithms

Run time: $O(n^k)$ for some const. k

P vs. NP

Input Length : n

P: Polynomial time algorithms

Run time: $O(n^k)$ for some const. k

NP: Non-deterministic Polynomial time algorithms

P vs. NP

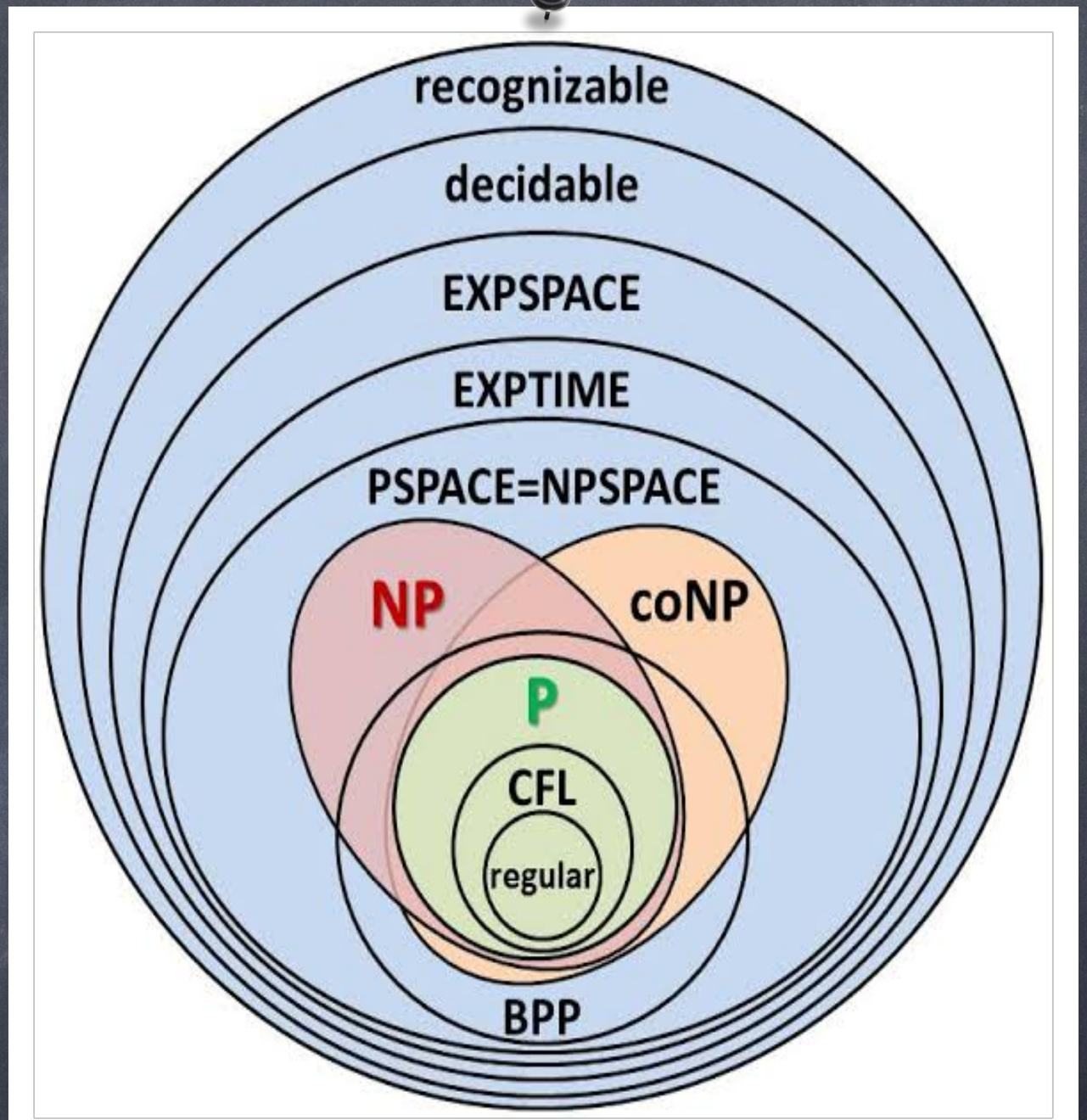
Input Length : n

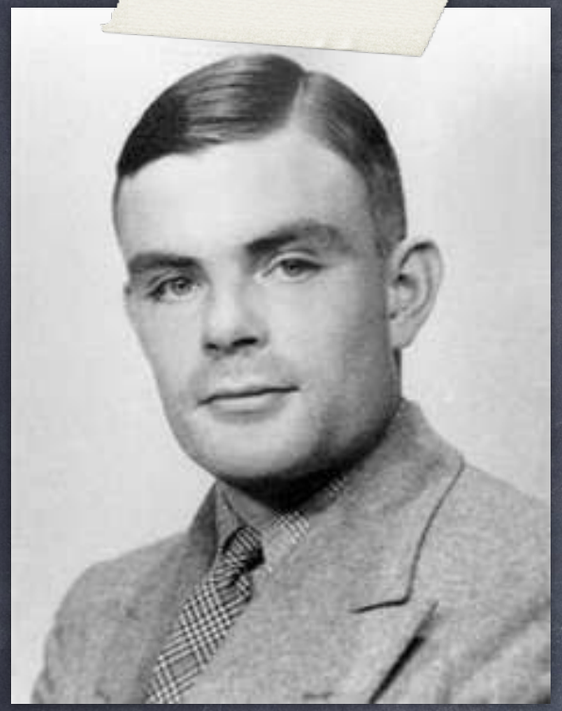
P: Polynomial time algorithms

Run time: $O(n^k)$ for some const. k

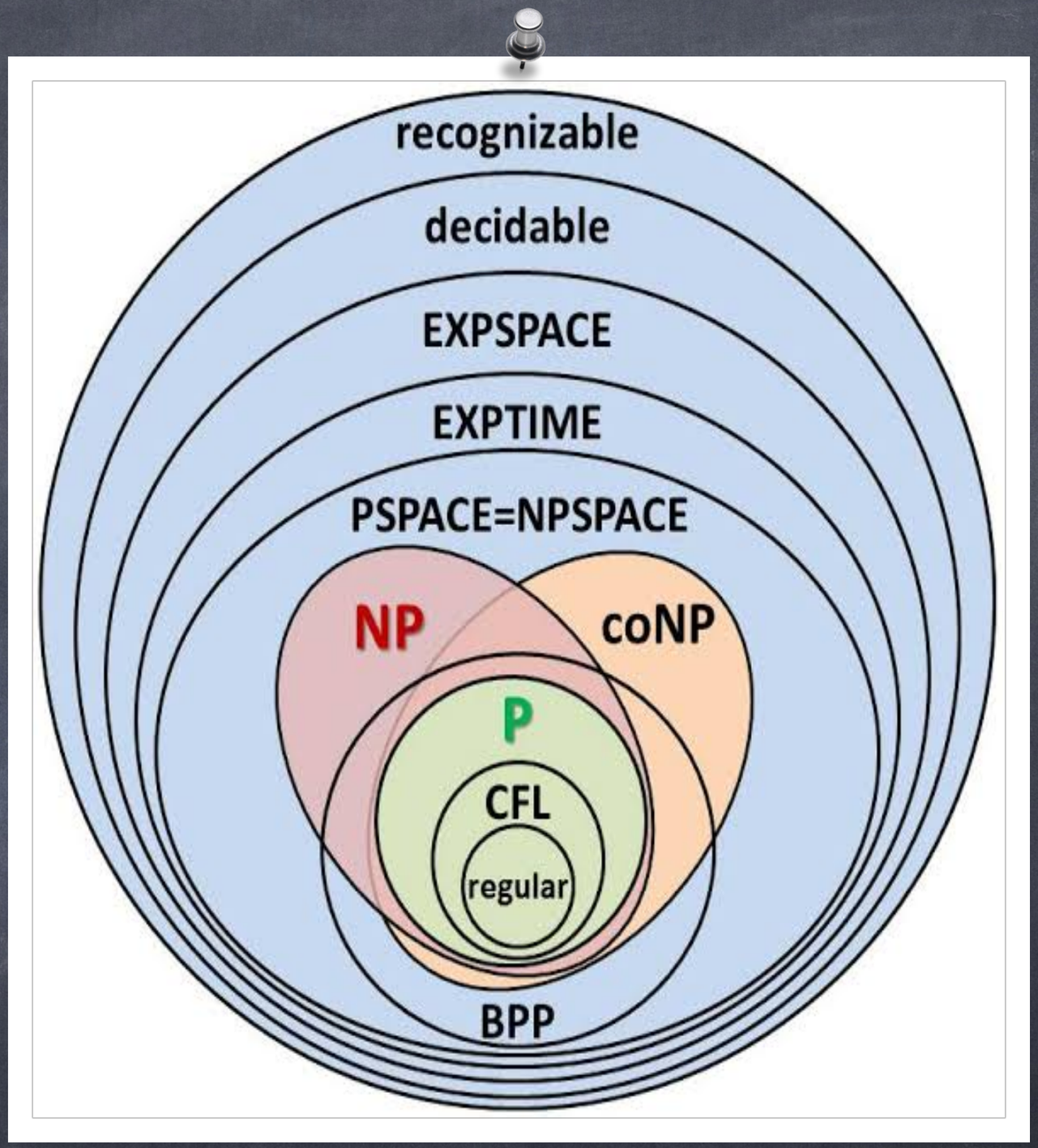
NP: Non-deterministic Polynomial time algorithms

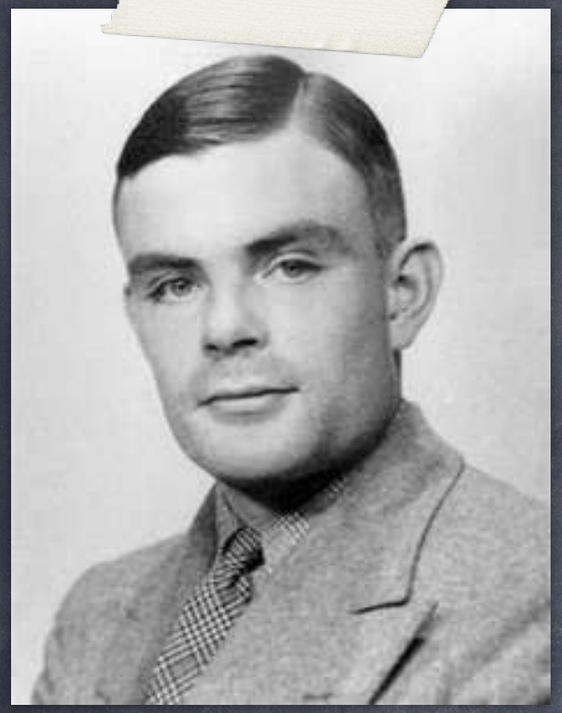
Verifying time: $O(n^k)$ with $O(n^k)$ bits of advice for some const. k


recognizable
decidable
EXPSPACE
EXPTIME
PSPACE=NPSPACE
NP
coNP
P
CFL
regular
BPP

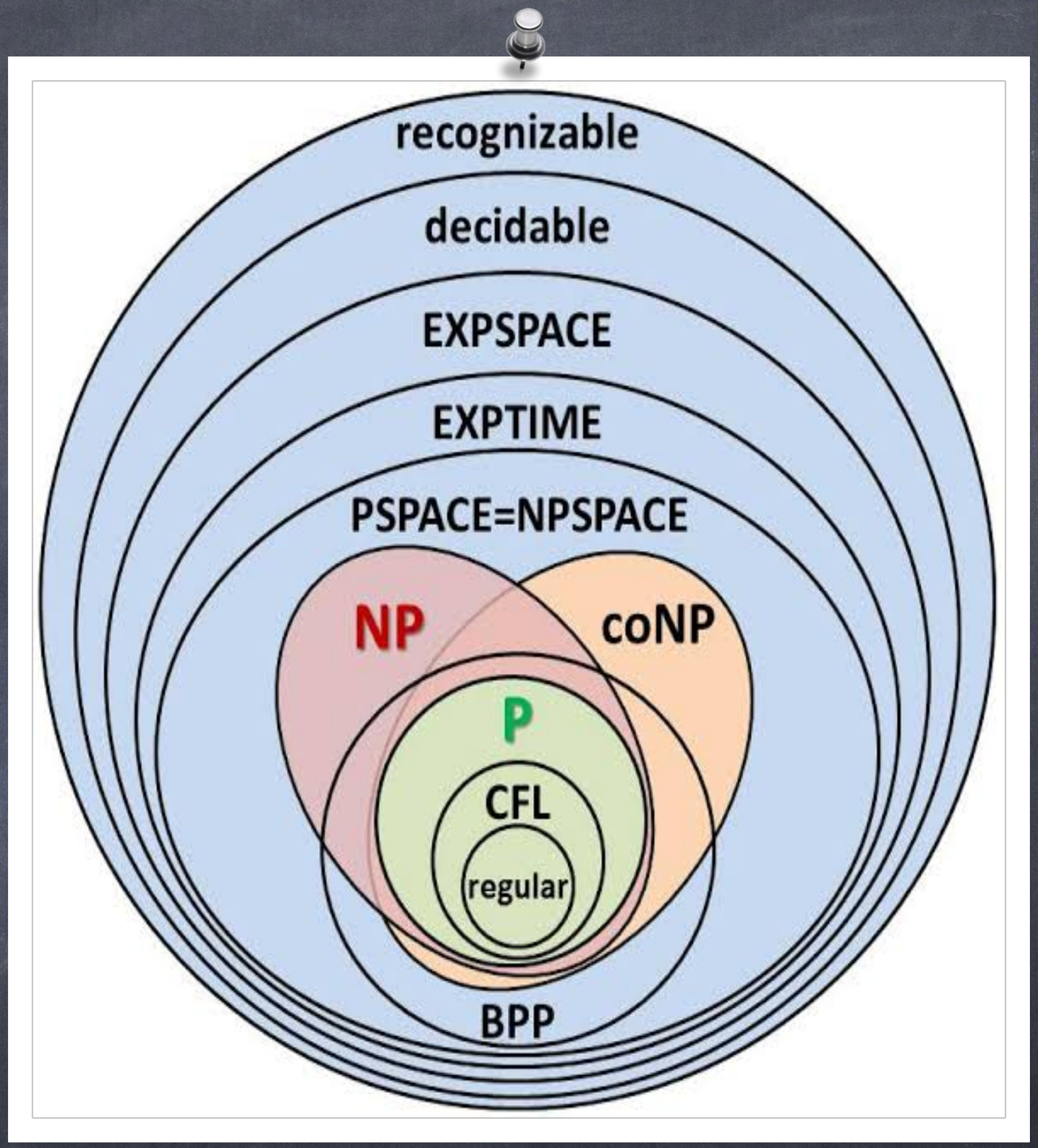


Alan Turing

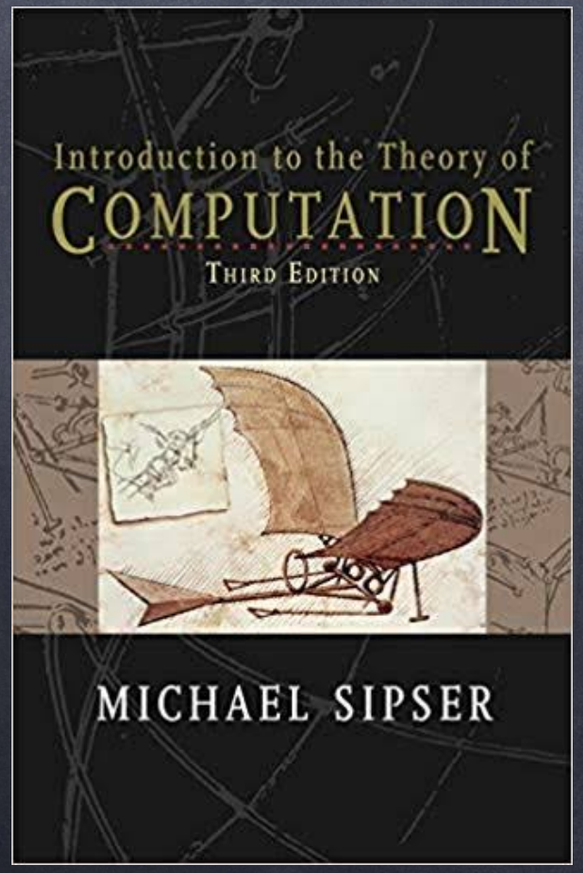

recognizable
decidable
EXPSPACE
EXPTIME
PSPACE=NPSPACE
NP
coNP
P
CFL
regular
BPP



Alan Turing



Amazing first book



A bit about what I do...

Algebraic Complexity Theory

A bit about what I do...

Algebraic Complexity Theory

Algebraic Problems

Easy?

Hard?

A bit about what I do...

Algebraic Complexity Theory

Algebraic Problems?

A bit about what I do...

Algebraic Complexity Theory

Algebraic Problems

1. Primality Test

A bit about what I do...

Algebraic Complexity Theory

Algebraic Problems

1. Primality Test

A bit about what I do...

Algebraic Complexity Theory

Algebraic Problems

1. Primality Test
2. Matrix Multiplication

A bit about what I do...

Algebraic Complexity Theory

Algebraic Problems

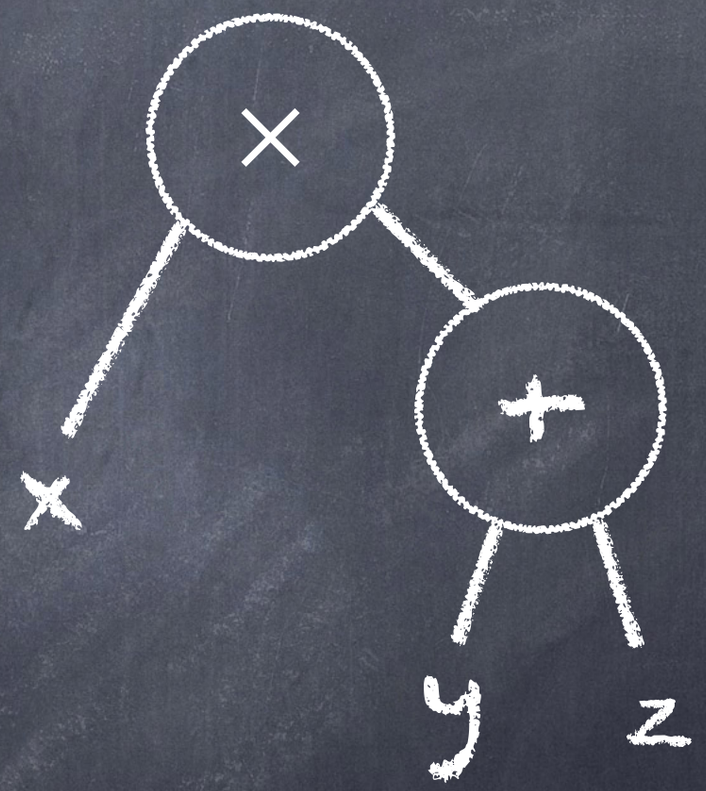
1. Primality Test
2. Matrix Multiplication
3. Integer Factorisation

A bit about what I do...

Algebraic Complexity Theory

Algebraic Problems

1. Primality Test
2. Matrix Multiplication
3. Integer Factorisation
4. Algebraic Circuit Complexity

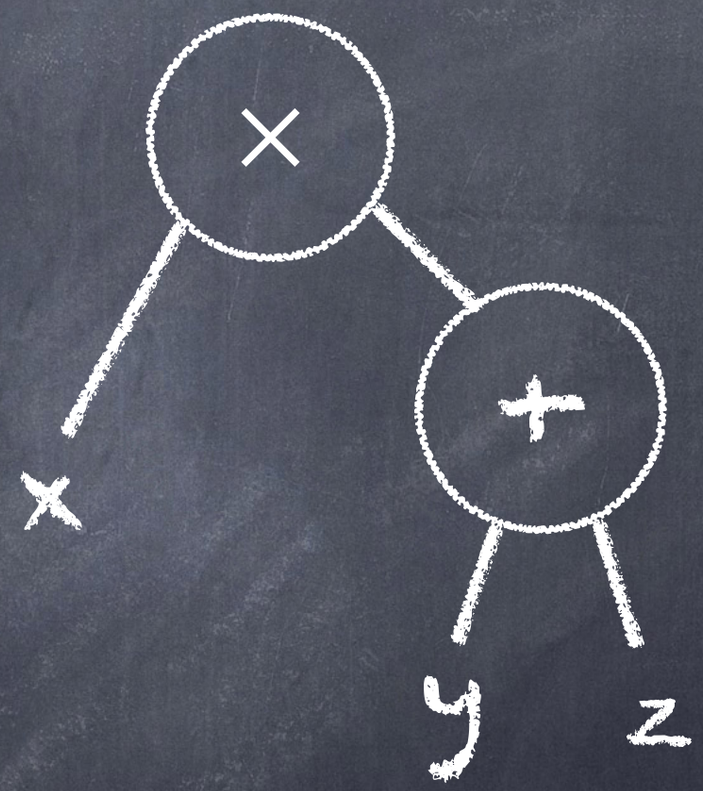


A bit about what I do...

Algebraic Complexity Theory

Algebraic Problems

1. Primality Test
2. Matrix Multiplication
3. Integer Factorisation
4. Algebraic Circuit Complexity



and many more....