Dynamic Programming

Andrew Hankinson

MUMT 611 - February 1, 2006

Agenda

- An introduction
- Some sample problems
- How it may be used

An Introduction

```
6400. 215. "HDA-208"
array("Quantum" . "
9100. 269. "HDA-16
array("Fujitsu",
13600, 275, "HD
array("Seagate"
10200. 245. "
```


6400. 215.
array("Qu
9100. 26
array(
1360, 275,
array("Seag
200. 24

Scheduled	Flug-Nummer Flight-Number	Aus From
13:10	LH 9538	Saarbrücken
13:55	DI 7068	Köln Bonn
15:30	C9 1521	Heringsdorf
15:30	LH 9511	Heringsdorf
16:35	SN 2585	Brüssel
17:50	HE 518	Dortmund
17:55	DI 7072	Köln Bonn
:10	D5 116	Dortmund
15	3L 306	Friedrichshafen
0	C9 1586	

Takes big problems...

40	+	20

breaks them into smaller problems

((2x5) x 4)	+	((2 x 5) x 2)

solves & caches smaller common results
(which are easier to compute)

(2 x 5 = 10) -> *m*

m x 4	+	*m* x 2

and solves the larger problem.

60

That was easy!

Try this...

calculate fib(5)

0 1

1 1

2 2 2

3 2 2 1

4 3 3 3 2 2

5 5 5 5 5 5 3 3 3

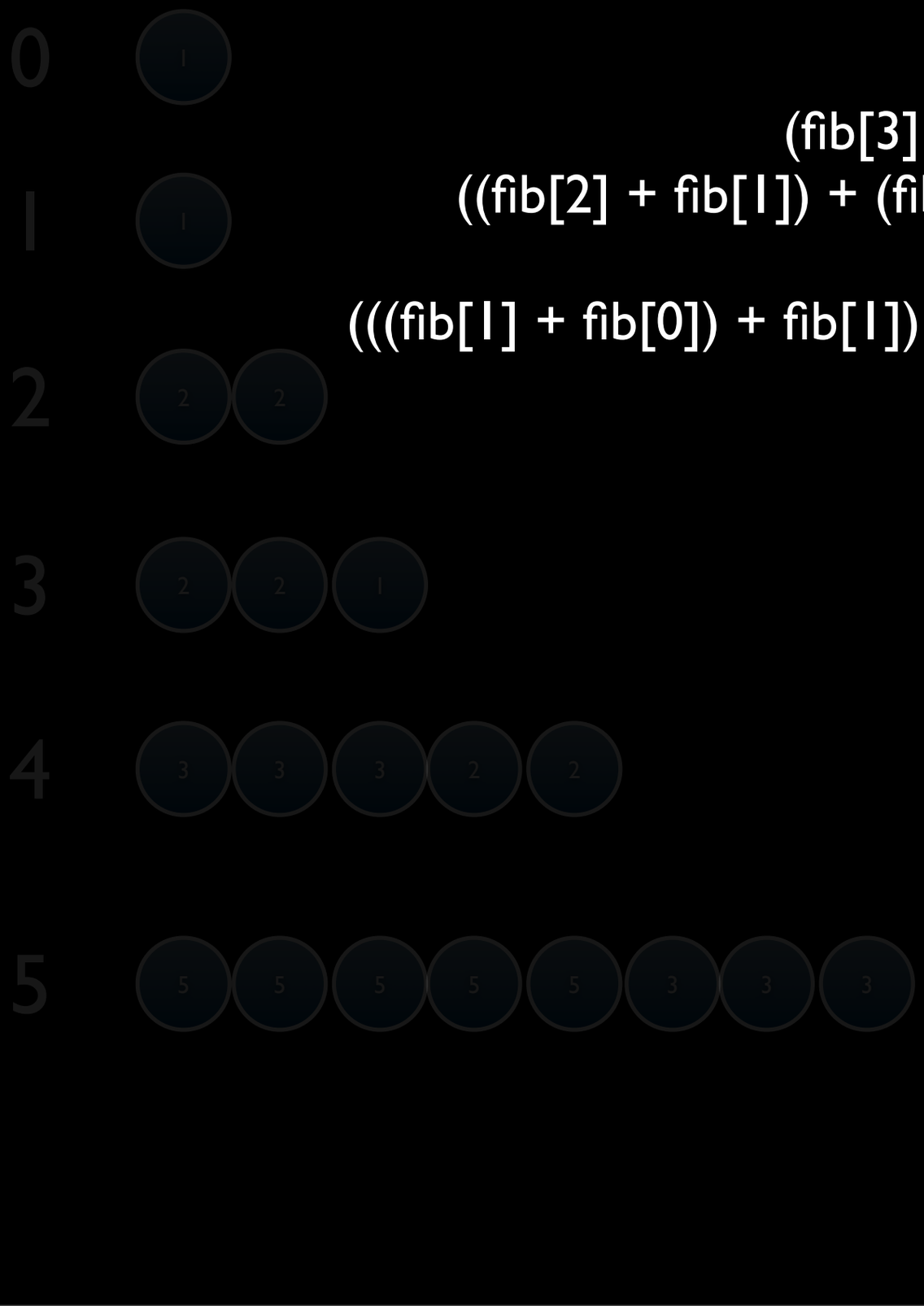
fib[5]

fib[4] + fib[3]

(fib[3] + fib[2]) + (fib[2] + fib[1])

((fib[2] + fib[1]) + (fib[1] + fib[0])) + ((fib[1] + fib[0]) + fib[1])

(((fib[1] + fib[0]) + fib[1]) + (fib[1] + fib[0])) + ((fib[1] + fib[0]) + fib[1])



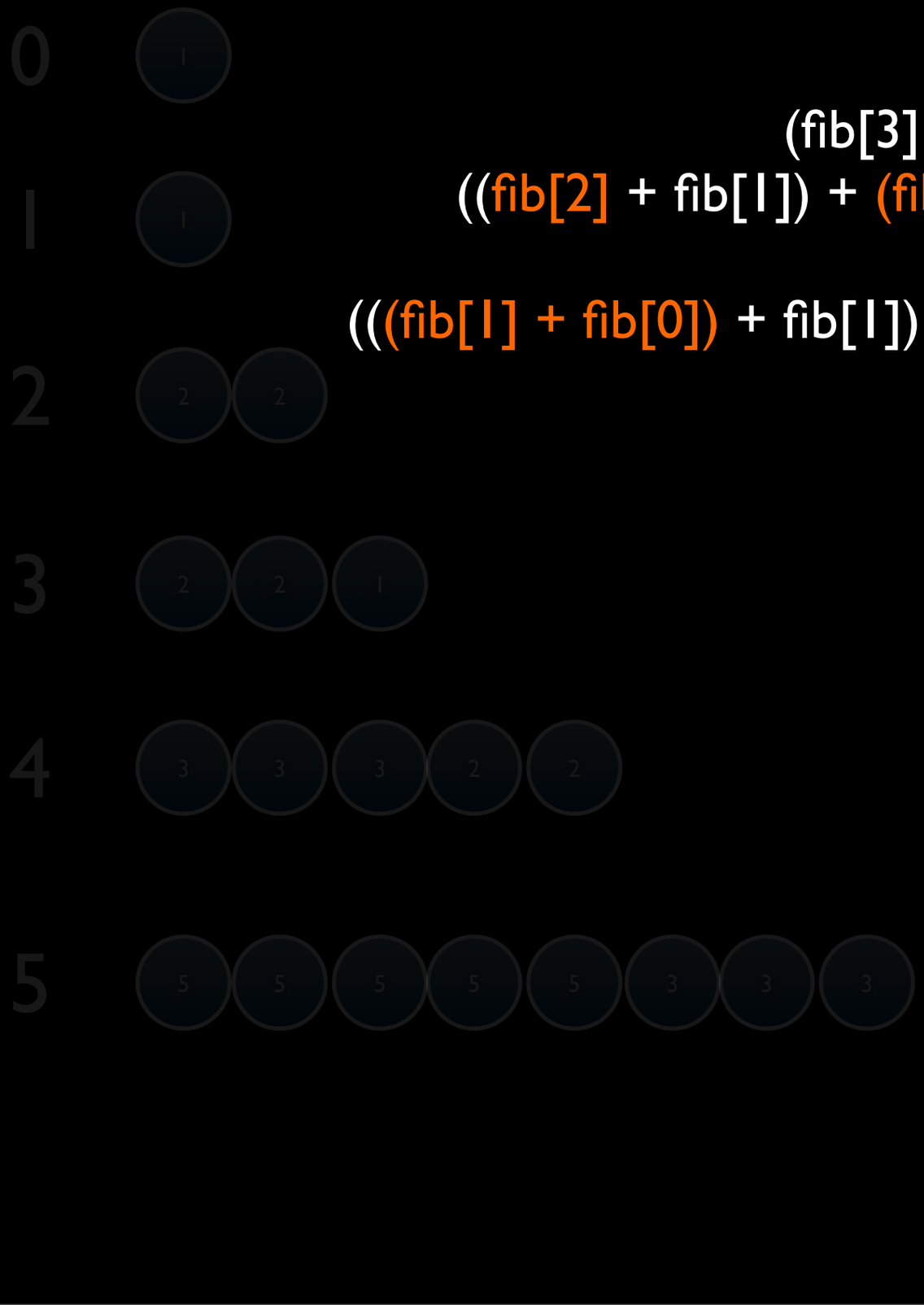
fib[5]

fib[4] + fib[3]

(fib[3] + fib[2]) + (fib[2] + fib[1])

((fib[2] + fib[1]) + (fib[1] + fib[0])) + ((fib[1] + fib[0]) + fib[1])

(((fib[1] + fib[0]) + fib[1]) + (fib[1] + fib[0])) + ((fib[1] + fib[0]) + fib[1])



fib[5]

fib[4] + fib[3]

(fib[3] + fib[2]) + (fib[2] + fib[1])

((fib[2] + fib[1]) + (fib[1] + fib[0])) + ((fib[1] + fib[0]) + fib[1])

(((fib[1] + fib[0]) + fib[1]) + ~~(fib[1] + fib[0])~~) + (~~(fib[1] + fib[0])~~ + fib[1])

m

0 1

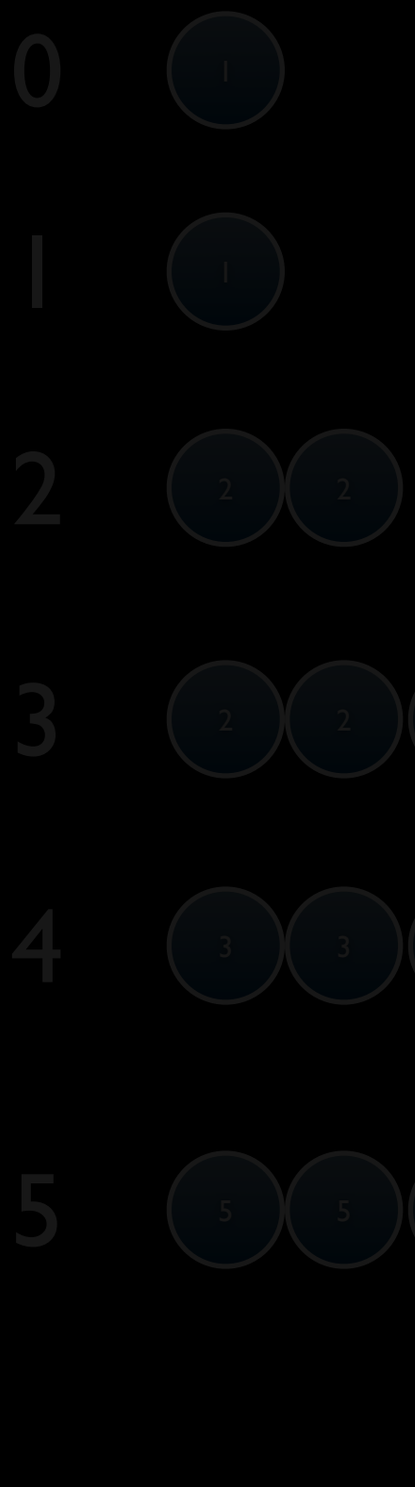
1 1

2 2 2

3 2 2 1

4 3 3 3 2 2

5 5 5 5 5 5 3 3 3



fib[5]

fib[4] + fib[3]

(fib[3] + m) + (m + fib[1])

((m + fib[1]) + m) + (m + fib[1])

Dynamic Programming

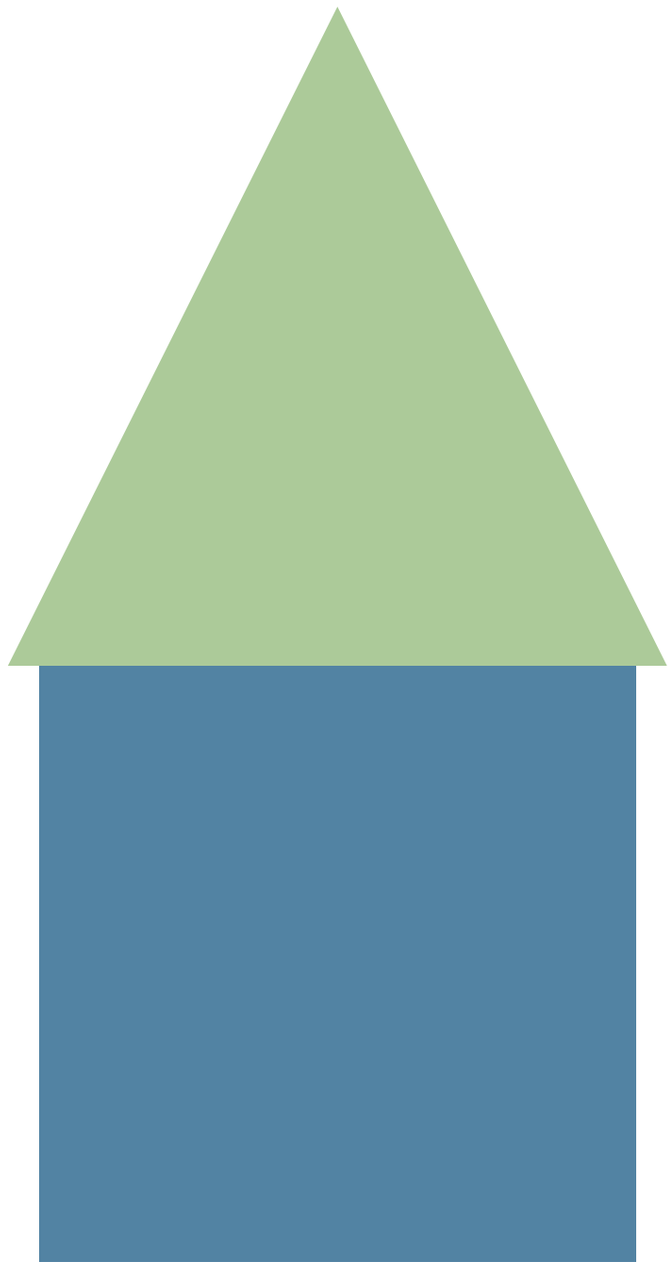
Optimal Substructure	
Best possible solution to subproblems (lowest ‘cost’)	

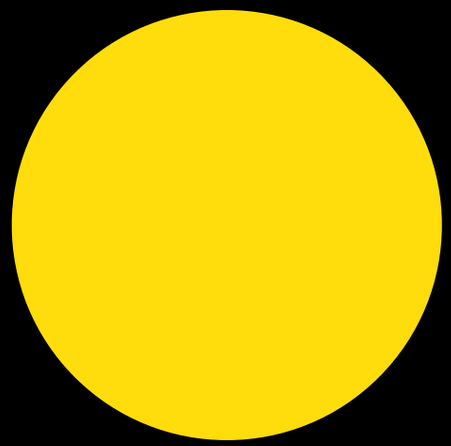
Dynamic Programming

Optimal Substructure	Overlapping Subproblems
Best possible solution to subproblems (lowest ‘cost’)	Each subproblem has components whose solutions are the same

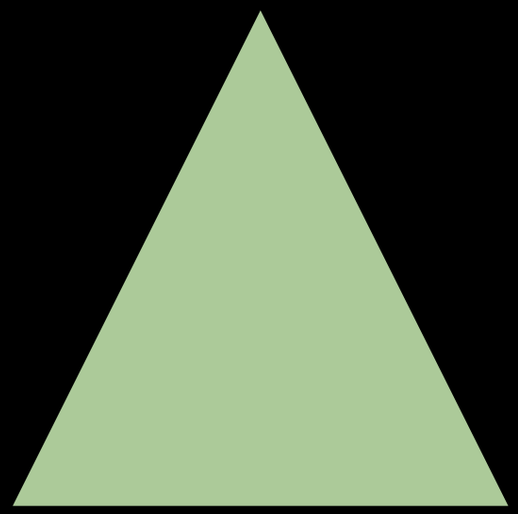
Applications

Computer Recognition

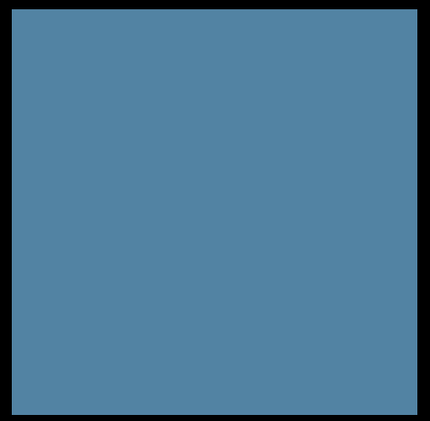




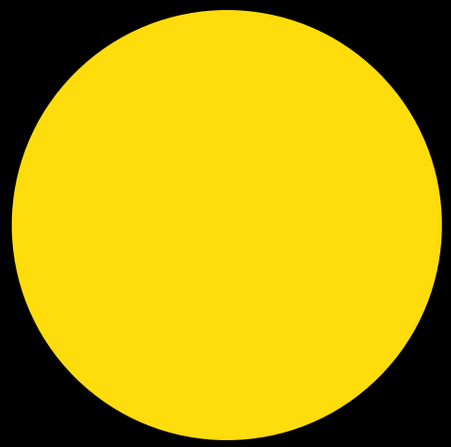
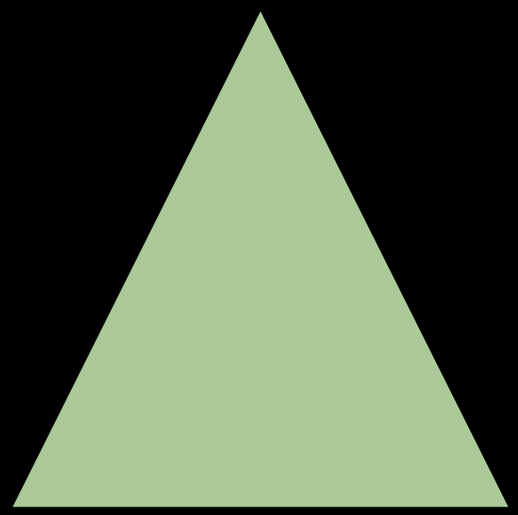
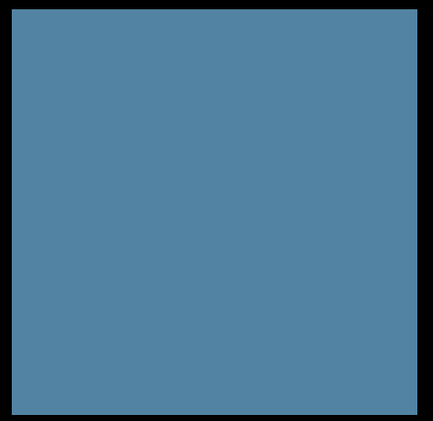
isRound
isYellow
isHigh



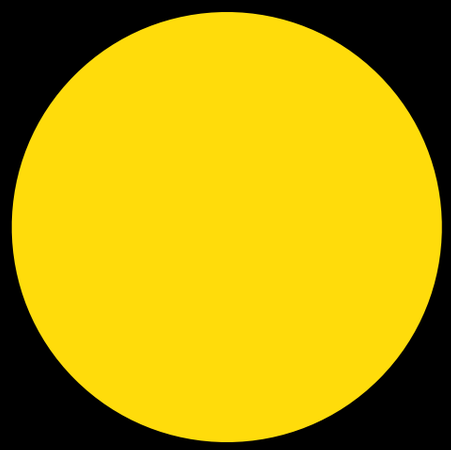
isTriangle
isGreen
isOnTopOfSquare

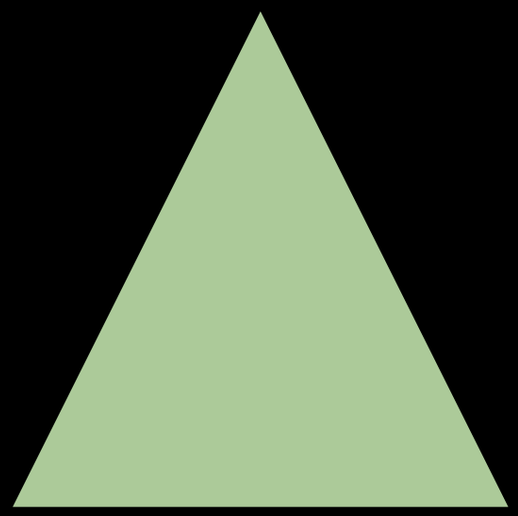


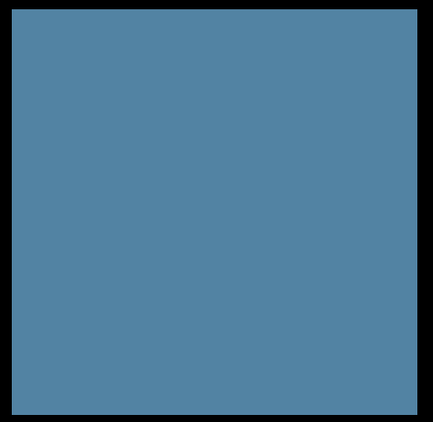
isSquare
isBlue
isBelowTriangle

		
isRound isYellow isHigh	isTriangle isGreen isOnTopOfSquare	isSquare isBlue isBelowTriangle

isRound	isYellow	isHigh	isTriangle	isGreen	isOnTopOfSquare	isSquare	isBlue	isBelowTriangle
compute Shape	compute Colour	compute Position	compute Shape	compute Colour	compute Position	compute Shape	compute Colour	compute Position







isRound isYellow isHigh			isTriangle isGreen isOnTopOfSquare			isSquare isBlue isBelowTriangle		
isRound	isYellow	isHigh	isTriangle	isGreen	isOnTopOfSquare	isSquare	isBlue	isBelowTriangle
compute Shape	compute Colour	compute Position	compute Shape	compute Colour	compute Position	compute Shape	compute Colour	compute Position

Look for similar solutions to different problems

Look for the optimal solution to every problem

Comments?