Dynamic Programming

Solving Optimization Problems with Dynamic Programming

SoftUni Foundation

SoftUni Team

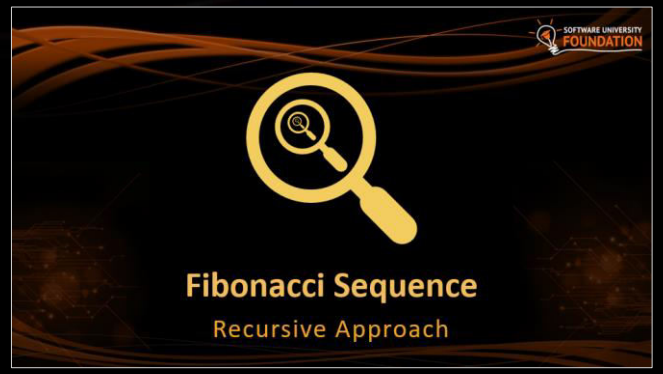
Technical Trainers

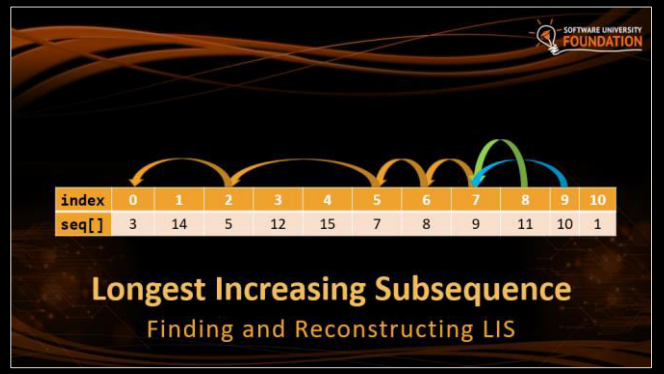
Software University

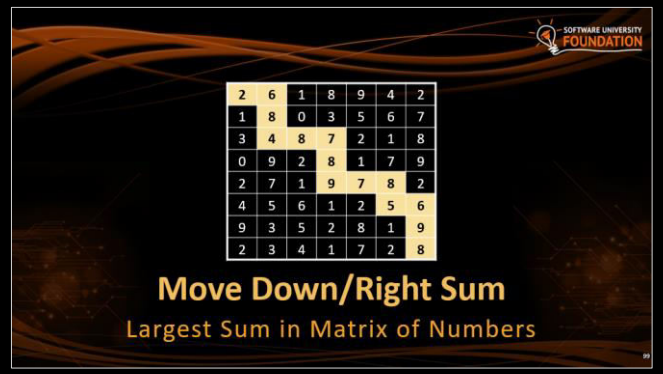
http://softuni.bg

Table of Contents

SoftUni Foundation







Have a Question?

SoftUni Foundation

sli.do

#DsAlgo

Dynamic Programming

SoftUni Foundation

- **Dynamic programming** (**DP**)
  - Solve **optimization** problems
  - Usually by breaking down into **overlapping sub-problems**
- In contrast → **Divide and Conquer**
  - Sub-problems don't overlap

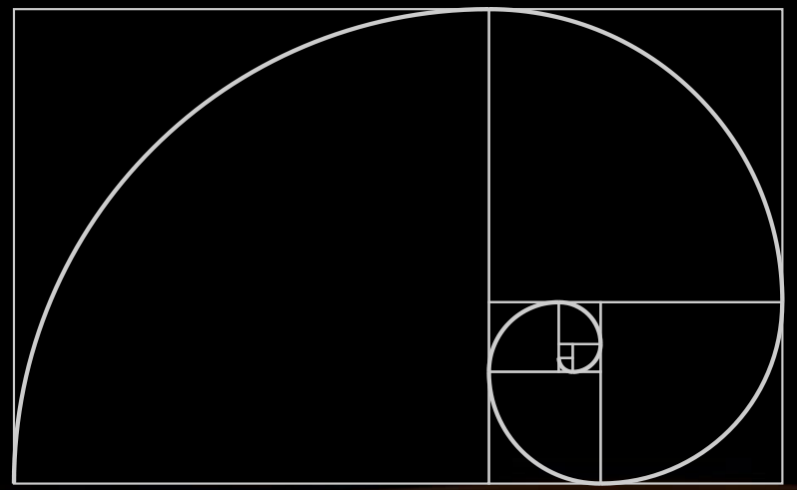
SoftUni Foundation

Fibonacci Sequence

Recursive Approach

Example: Fibonacci Sequence

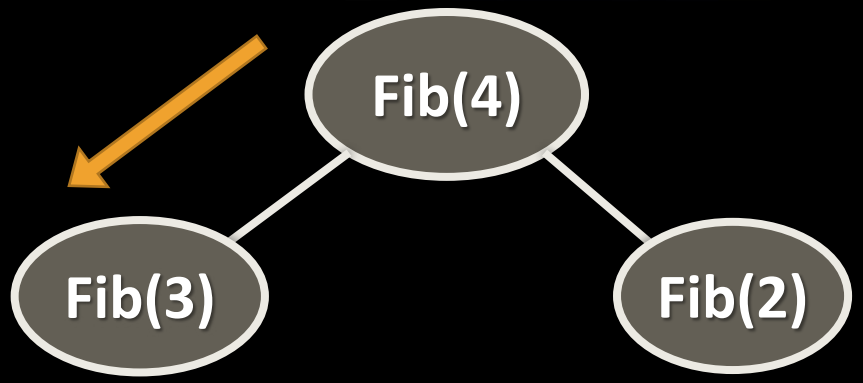
- **The Fibonacci sequence** holds the following integers:
  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …
  - The **first two** numbers are **0** and **1**
  - Each subsequent number is the sum of the previous two numbers
- Recursive mathematical formula:
  - $F_0 = 0, F_1 = 1$
  - $F_n = F_{n-1} + F_{n-2}$



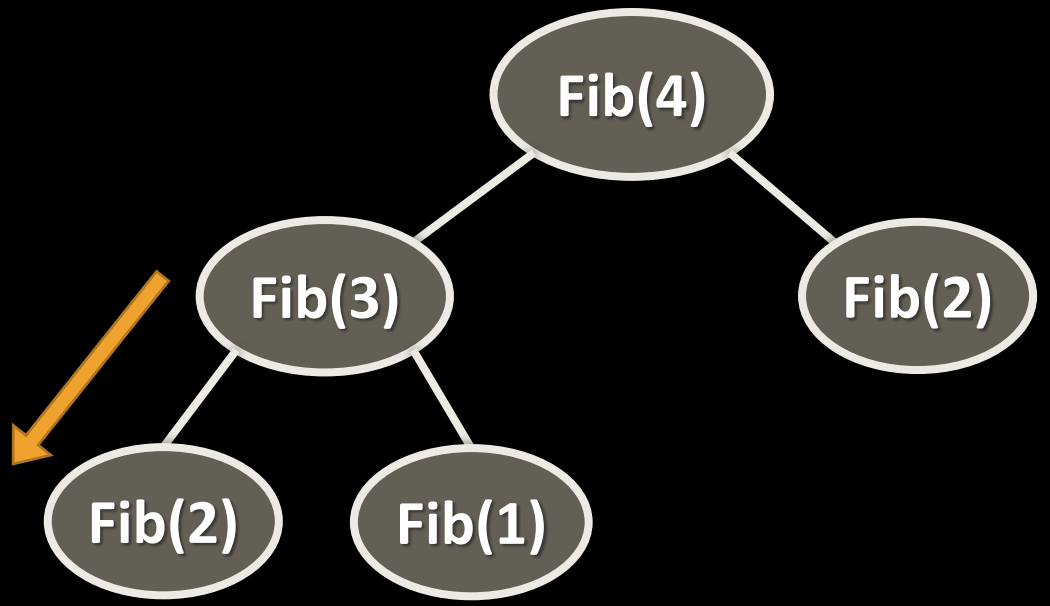
Fibonacci

- Fib(4)

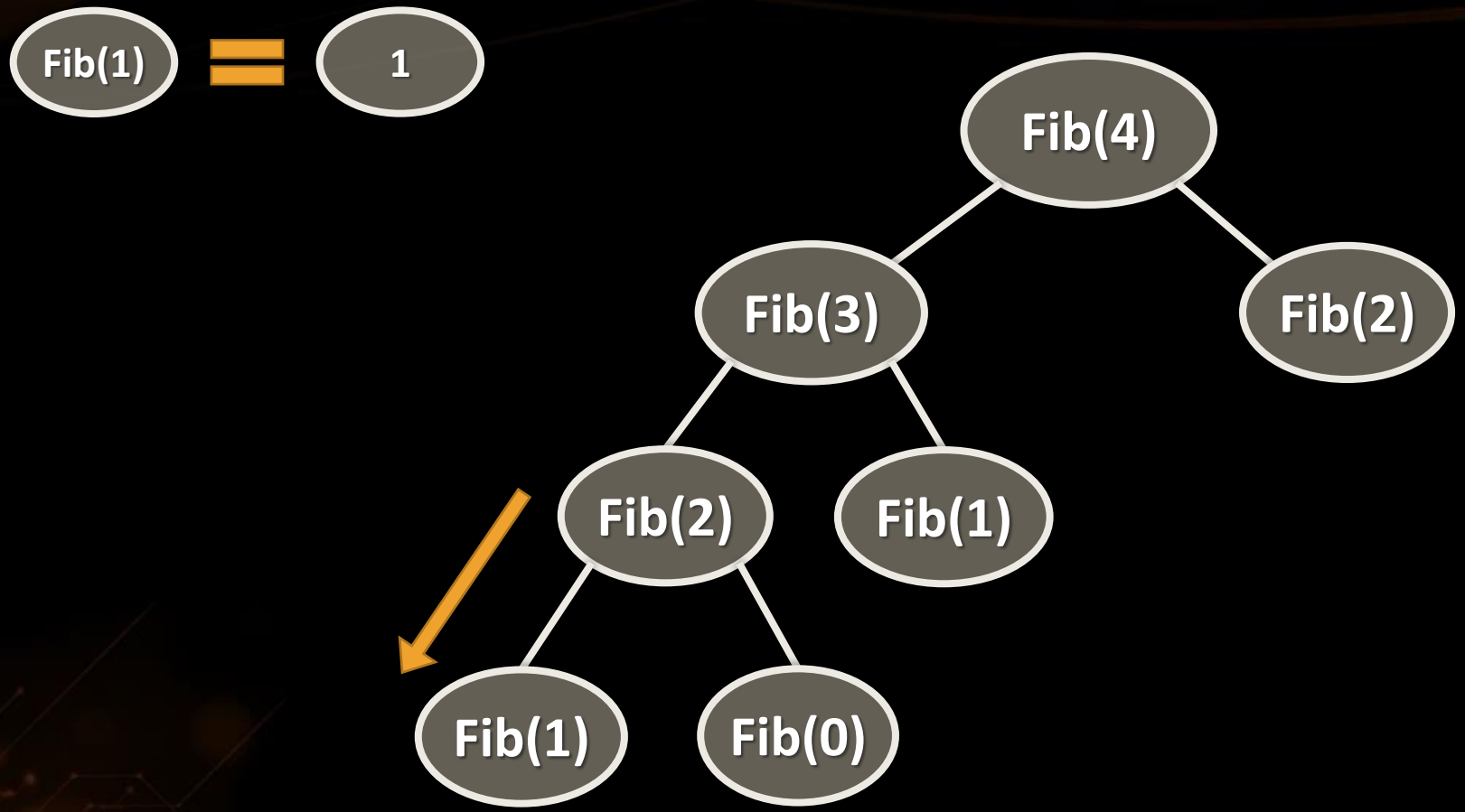
Fibonacci



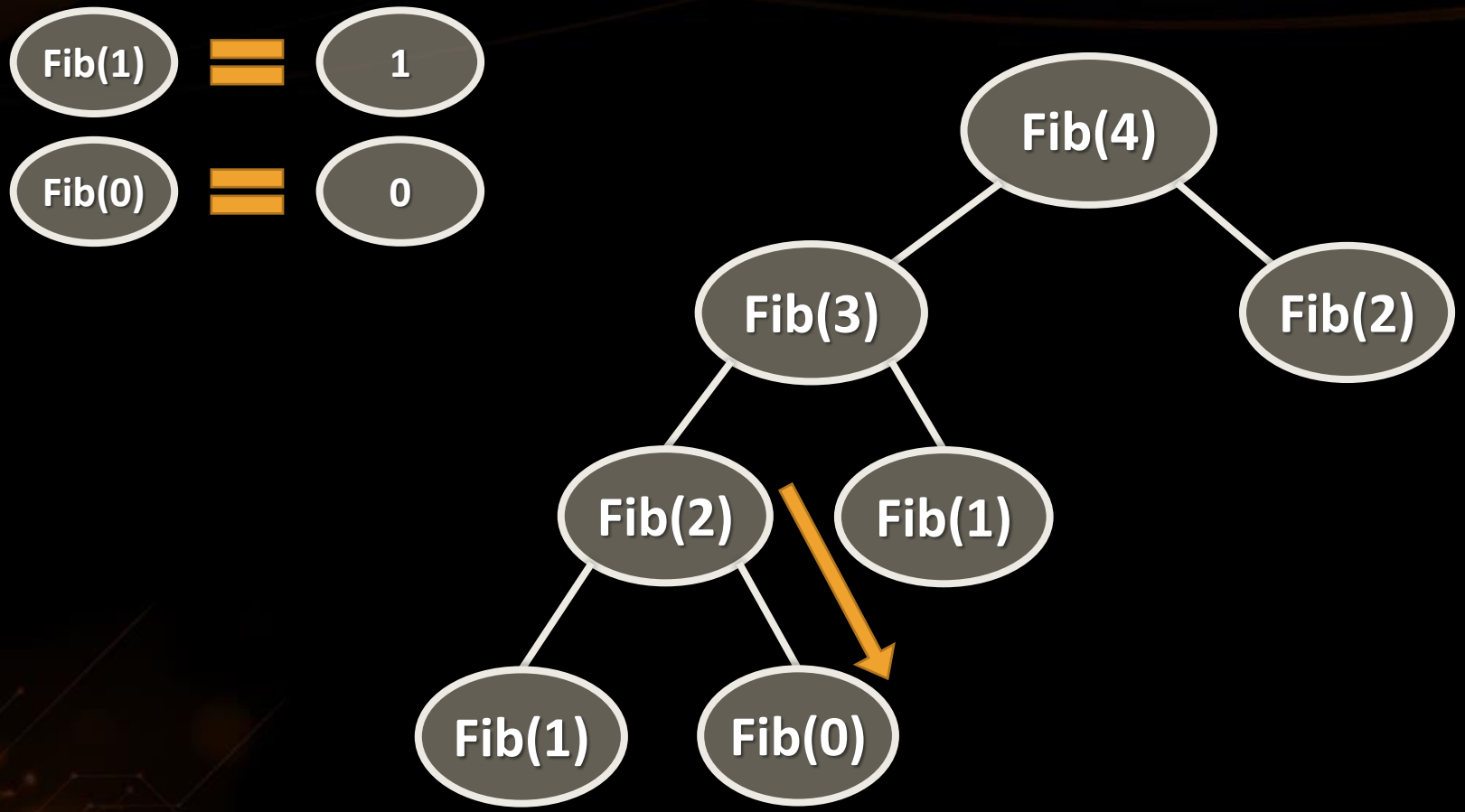
Fibonacci



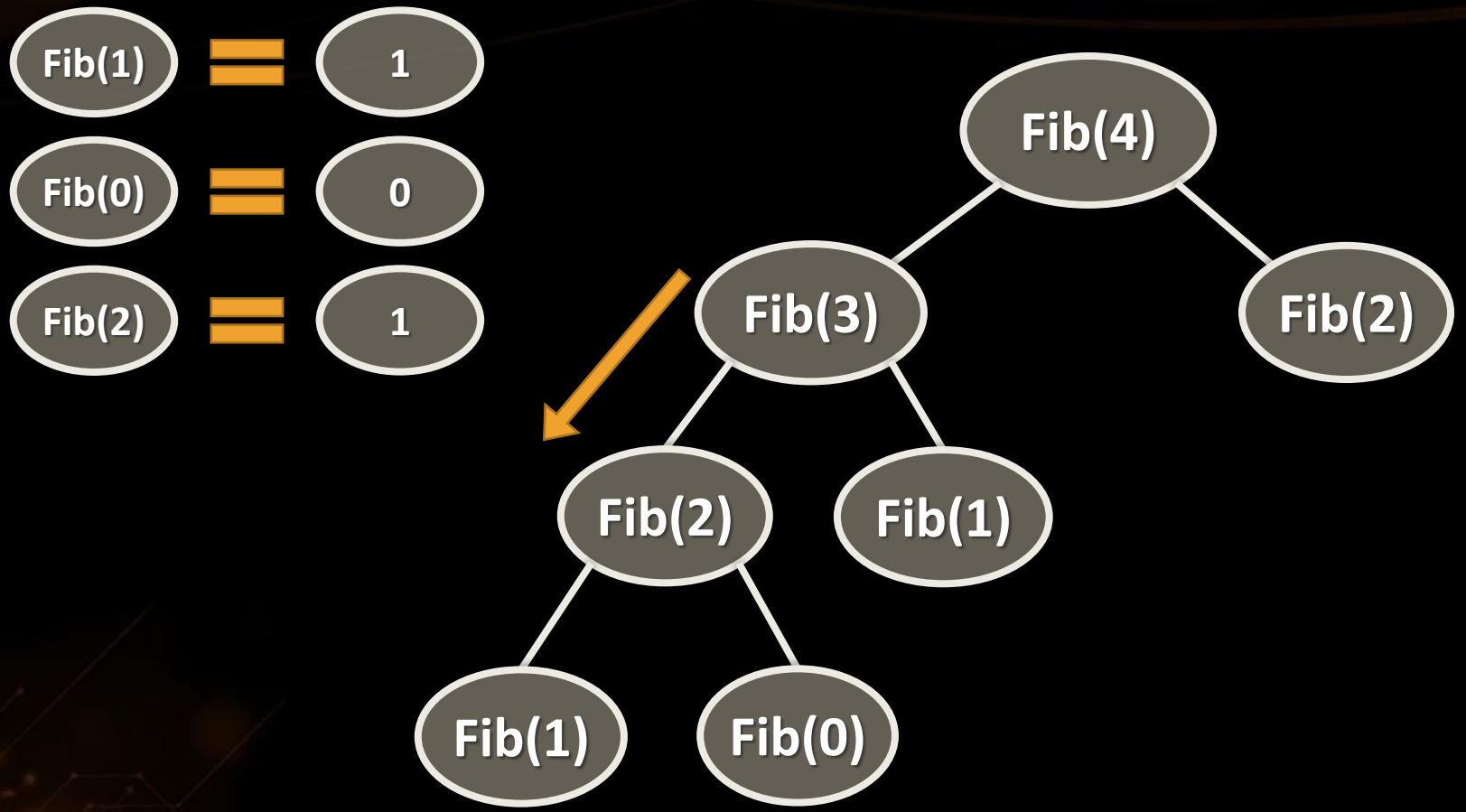
Fibonacci



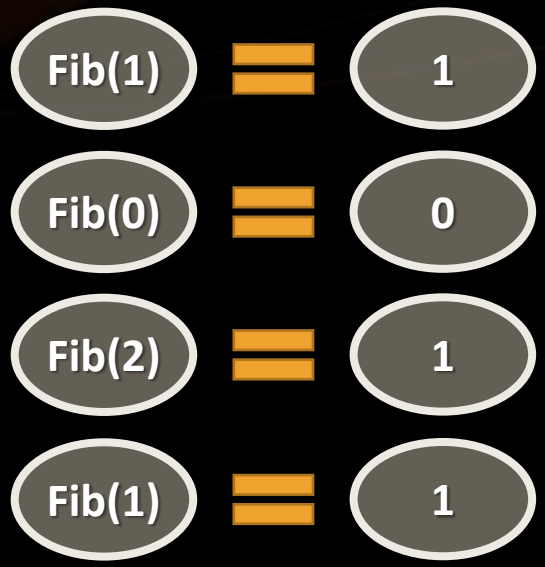
Fibonacci

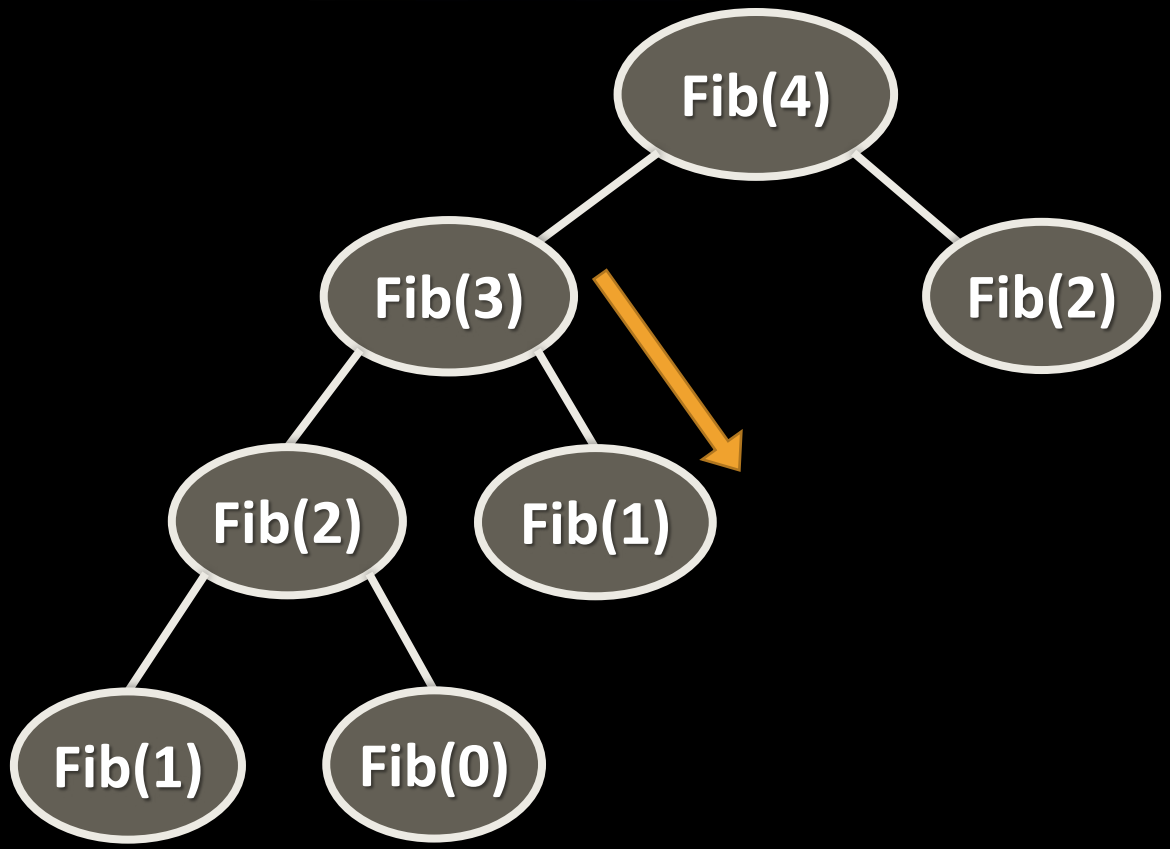


Fibonacci

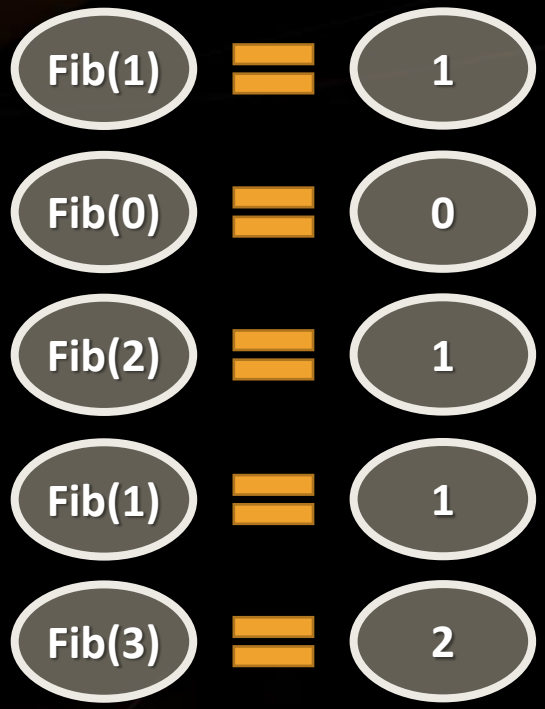


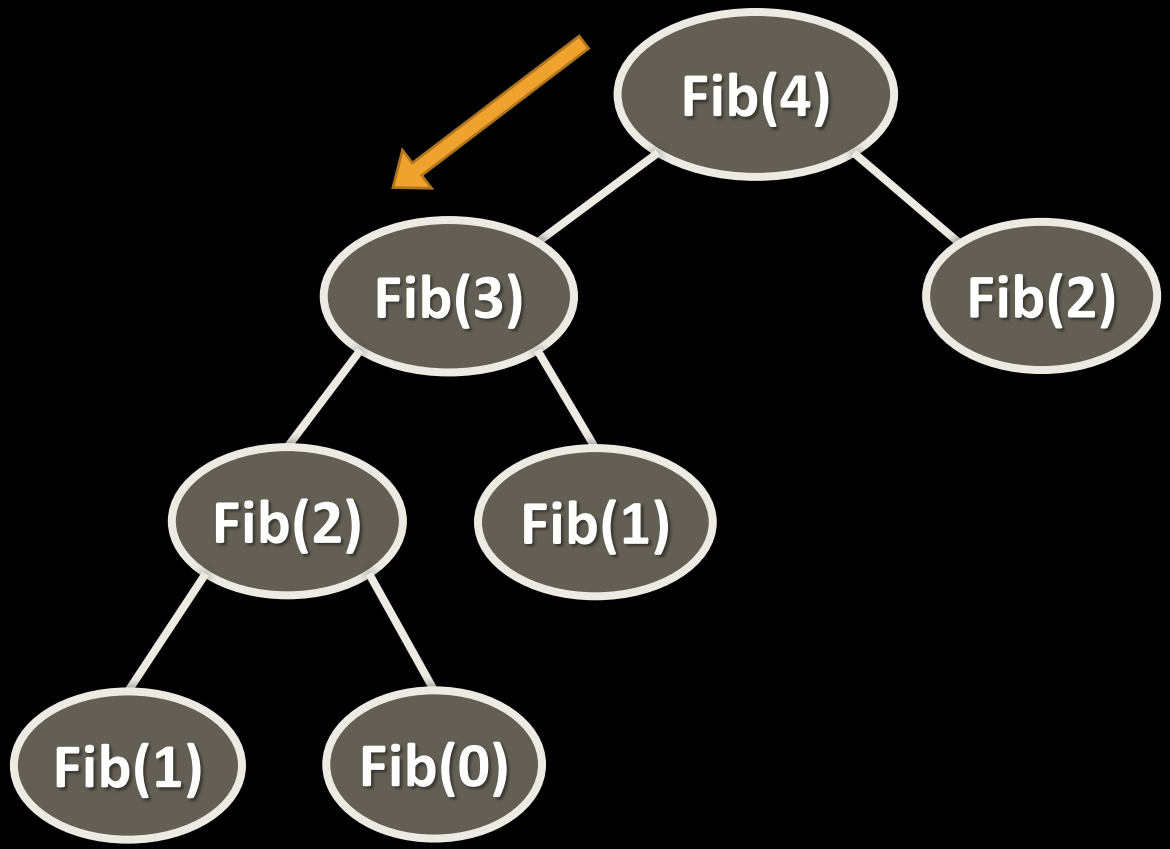
Fibonacci



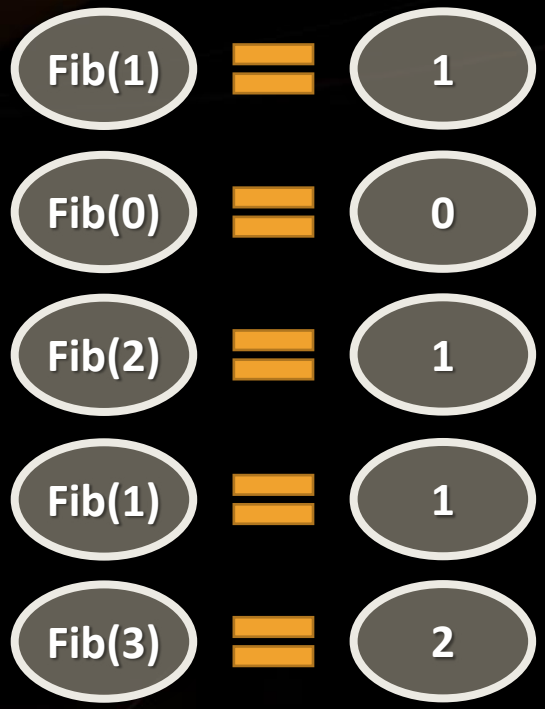


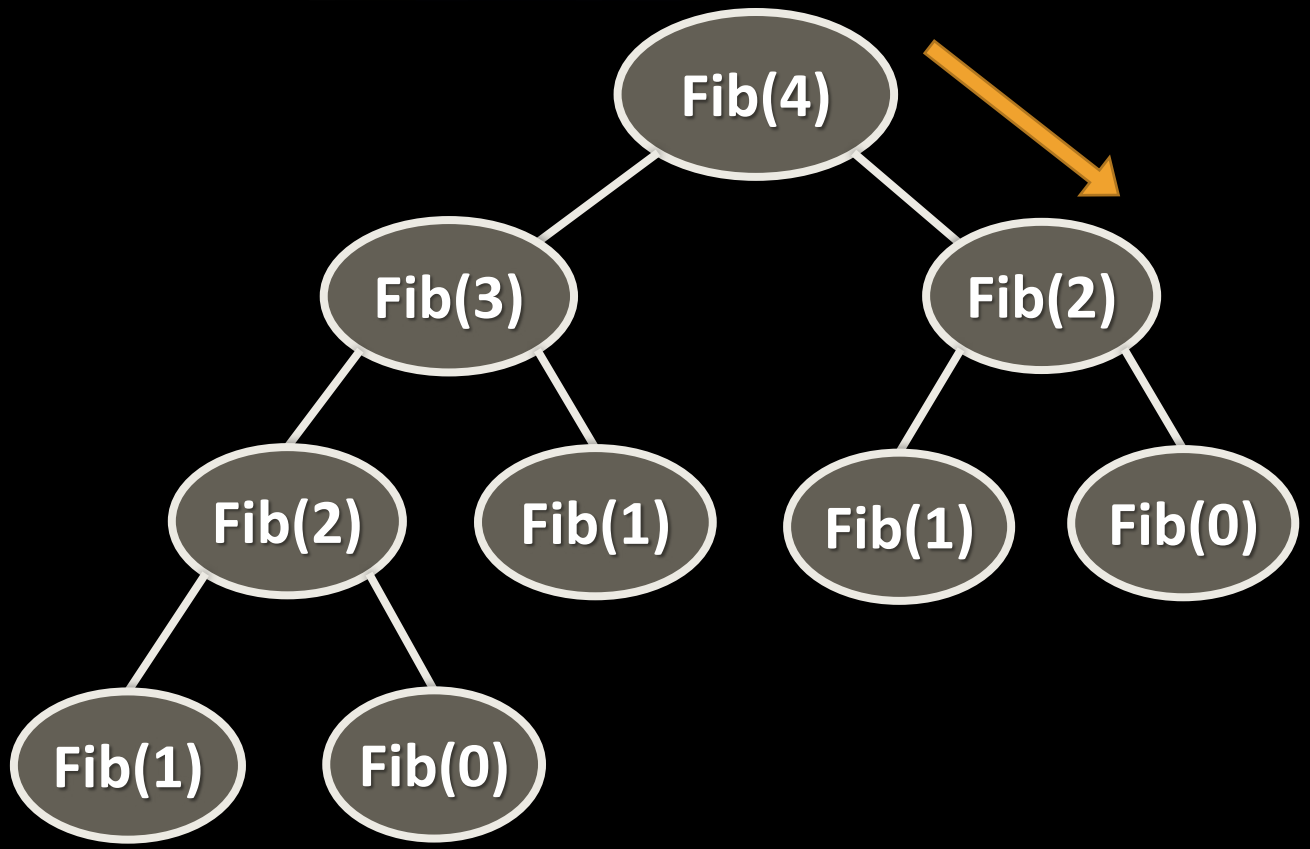
Fibonacci


Fib(1) = 1
Fib(0) = 0
Fib(2) = 1
Fib(1) = 1
Fib(3) = 2


Fib(4)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)

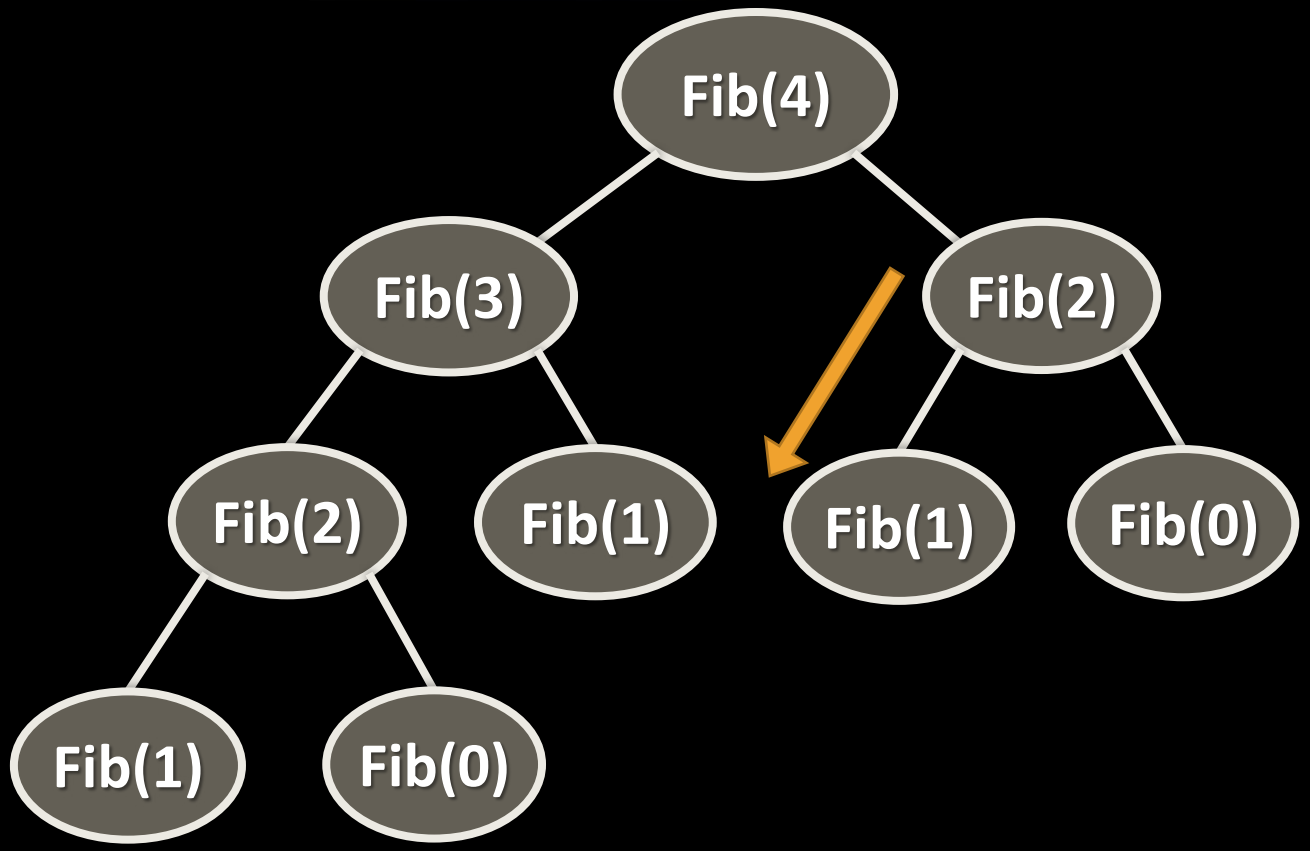
Fibonacci


Fib(1) = 1
Fib(0) = 0
Fib(2) = 1
Fib(1) = 1
Fib(3) = 2

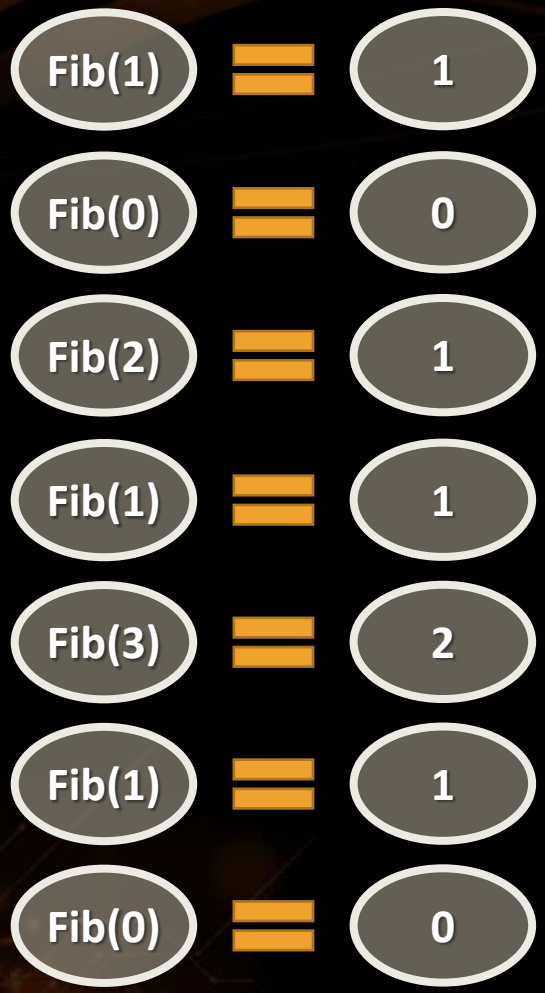

Fib(4)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

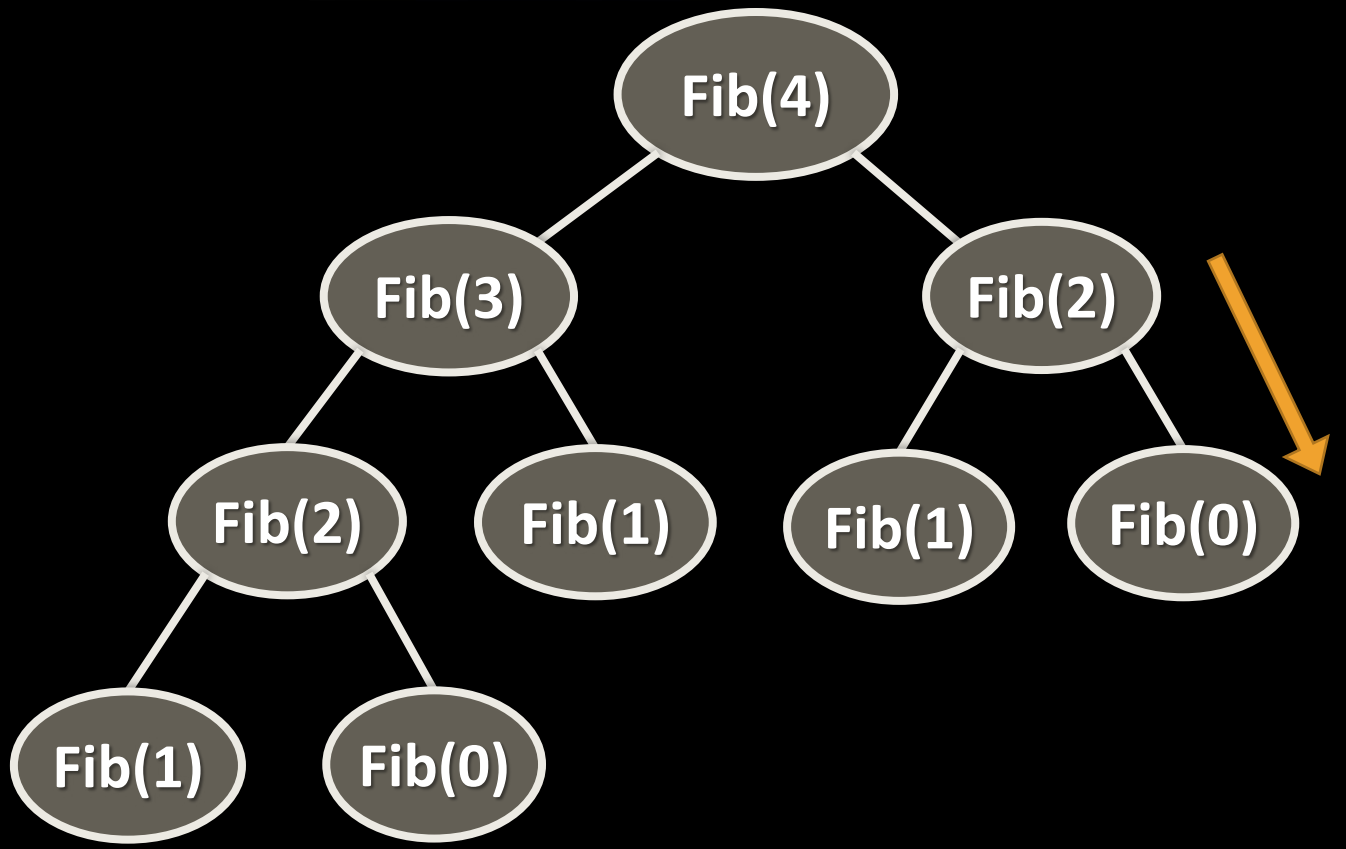
Fibonacci


Fib(1) = 1
Fib(0) = 0
Fib(2) = 1
Fib(1) = 1
Fib(3) = 2
Fib(1) = 1


Fib(4)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

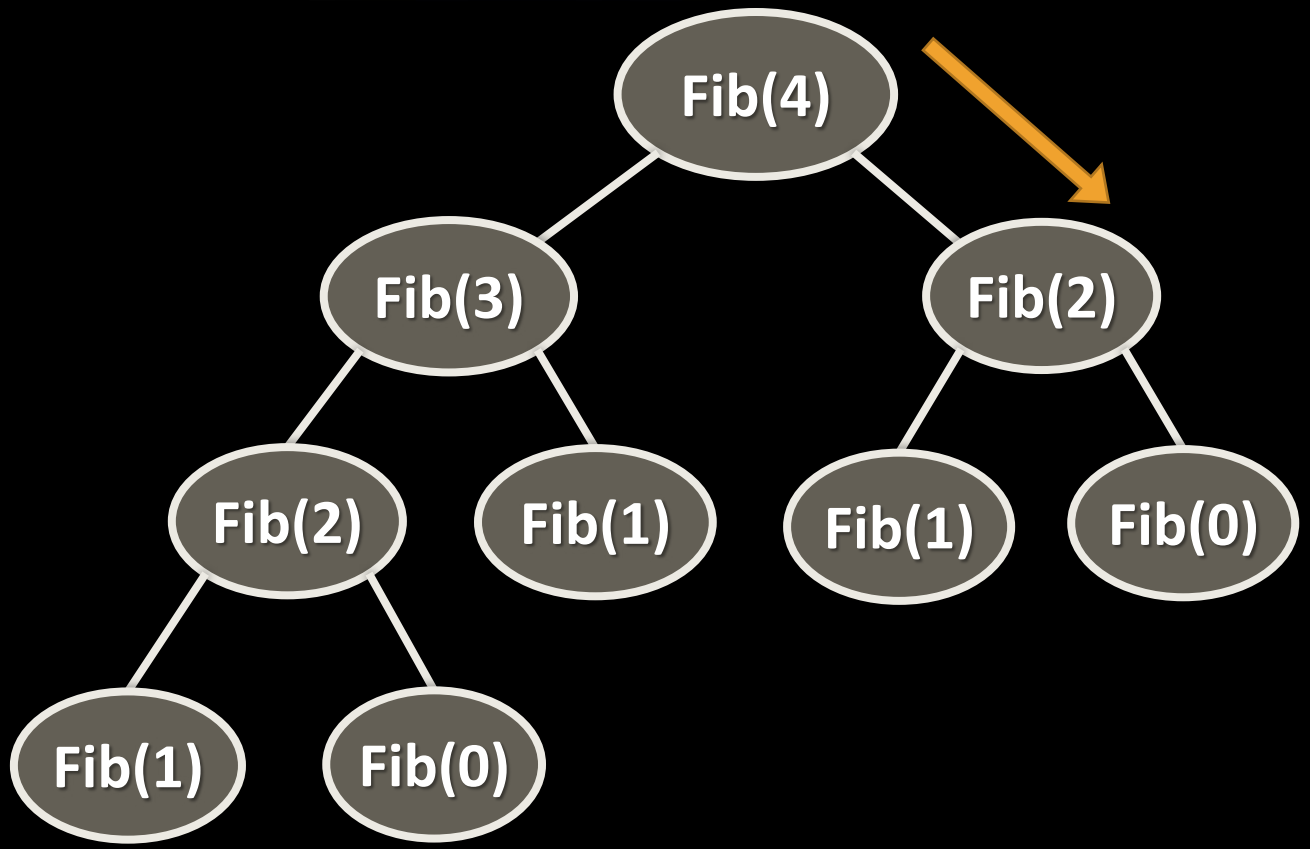
Fibonacci


Fib(1) = 1
Fib(0) = 0
Fib(2) = 1
Fib(1) = 1
Fib(3) = 2
Fib(1) = 1
Fib(0) = 0

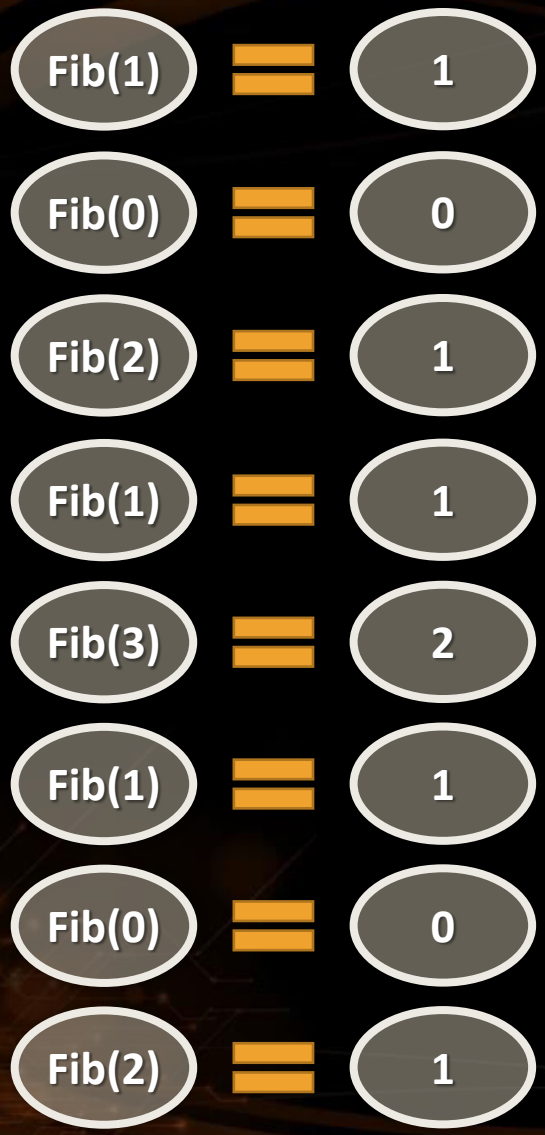

Fib(4)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

Fibonacci


Fib(1) = 1
Fib(0) = 0
Fib(2) = 1
Fib(1) = 1
Fib(3) = 2
Fib(1) = 1
Fib(0) = 0
Fib(2) = 1


Fib(4)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

Fibonacci


Fib(1) = 1
Fib(0) = 0
Fib(2) = 1
Fib(1) = 1
Fib(3) = 2
Fib(1) = 1
Fib(0) = 0
Fib(2) = 1


Fib(4) = 3

Fib(4)
Fib(3)
Fib(2)
Fib(2)
Fib(1)
Fib(1)
Fib(0)
Fib(1)
Fib(0)

Memoization

SoftUni Foundation

- DP → sub-problems **overlap**
- In order to **avoid solving** problems **multiple times**, memorize
  - **Memoization** → **save/cache** sub-problem solutions **for later use**
- Typically using an **array**, **matrix** or a **hash table**

SoftUni Foundation

Fibonacci Sequence with DP

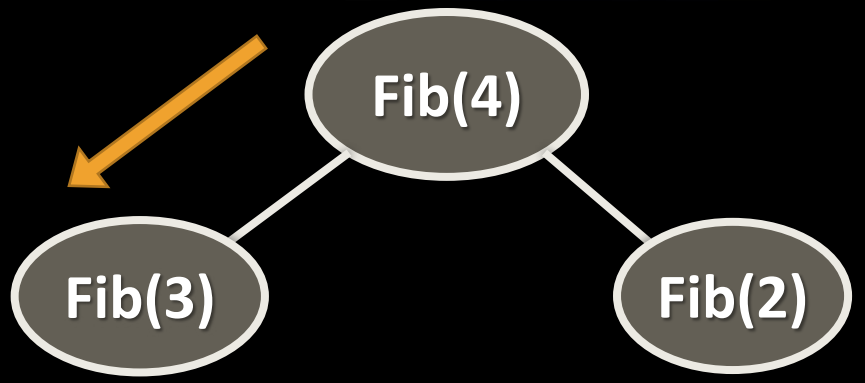
Recursive Top-Down Approach

Fibonacci

- Fib(4)

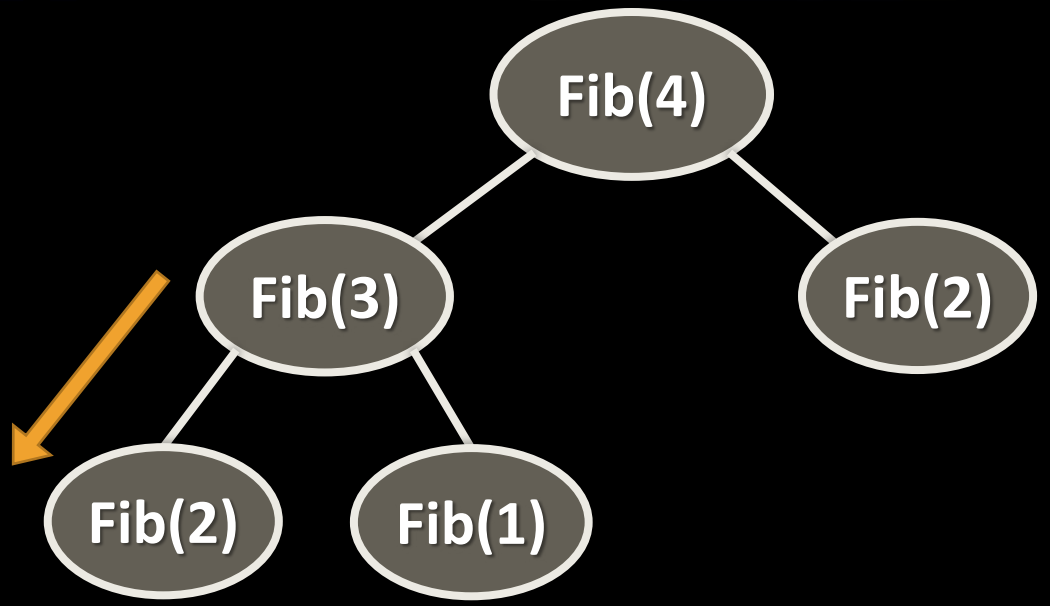
0	1	2	3	4

Fibonacci



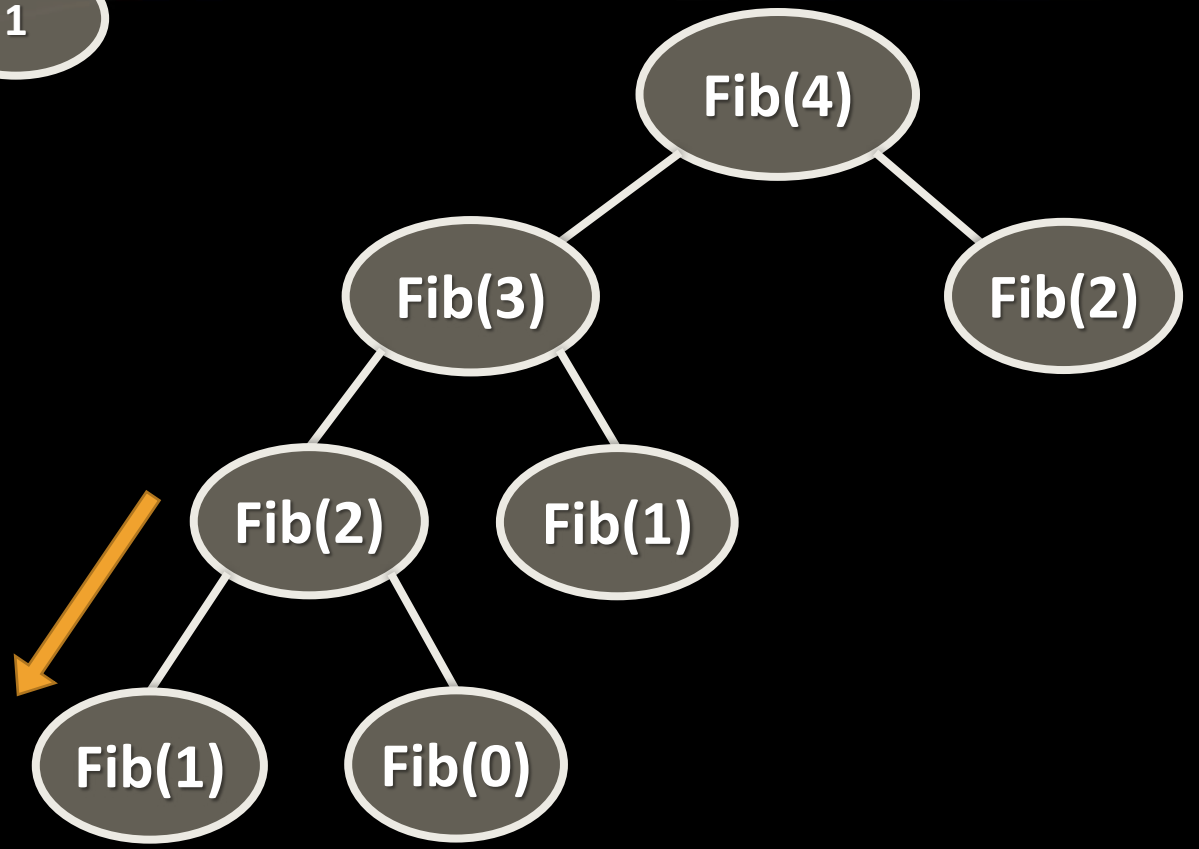
0	1	2	3	4

Fibonacci



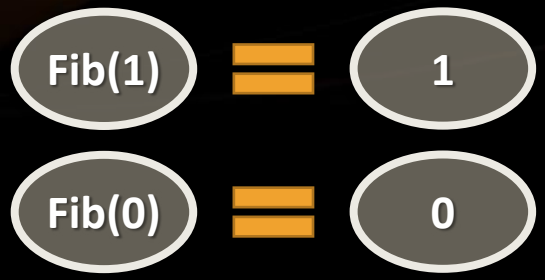
0	1	2	3	4

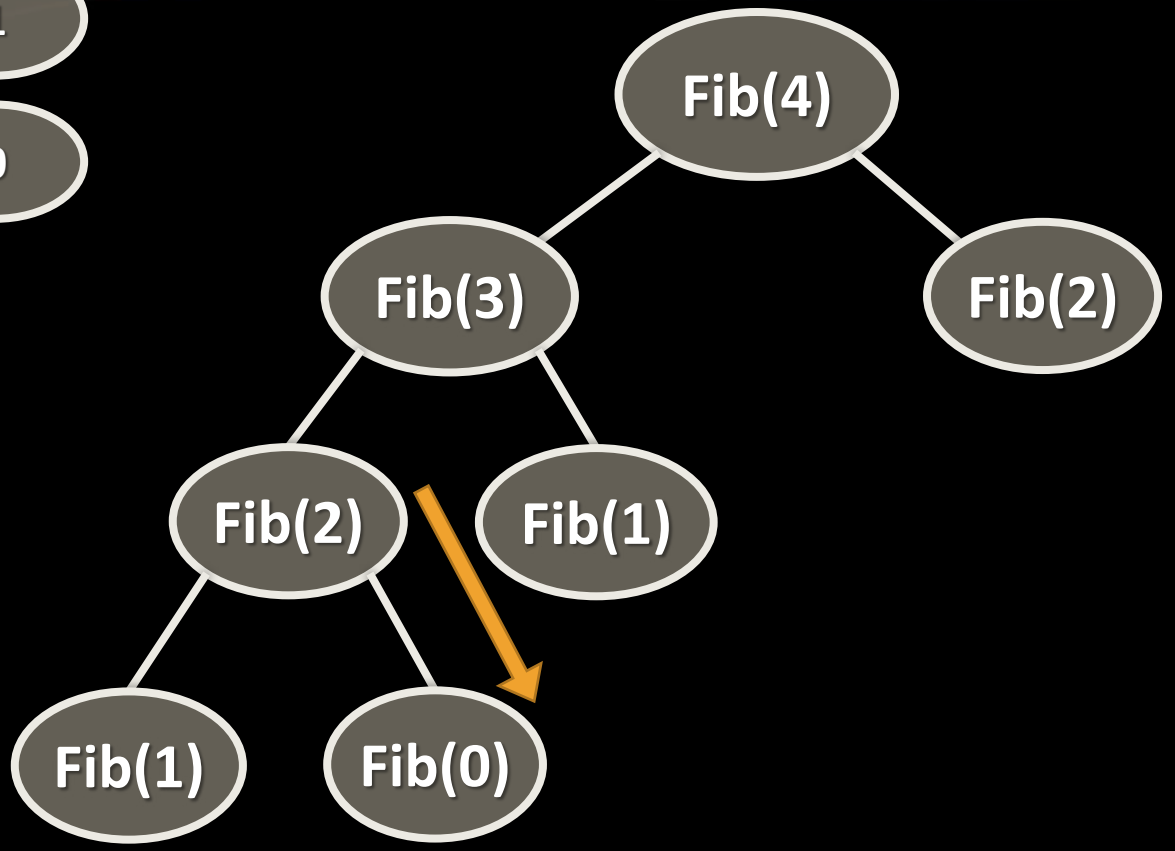
Fibonacci



0	1	2	3	4
	1			

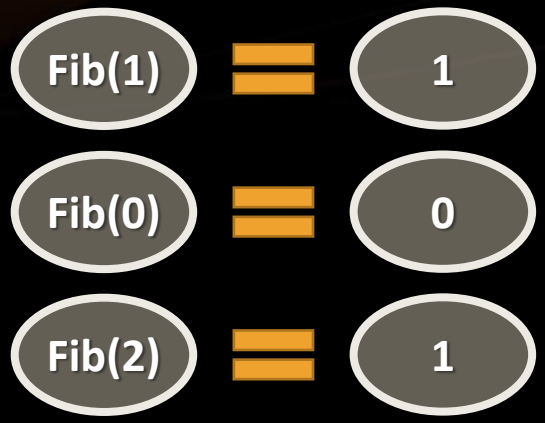
Fibonacci

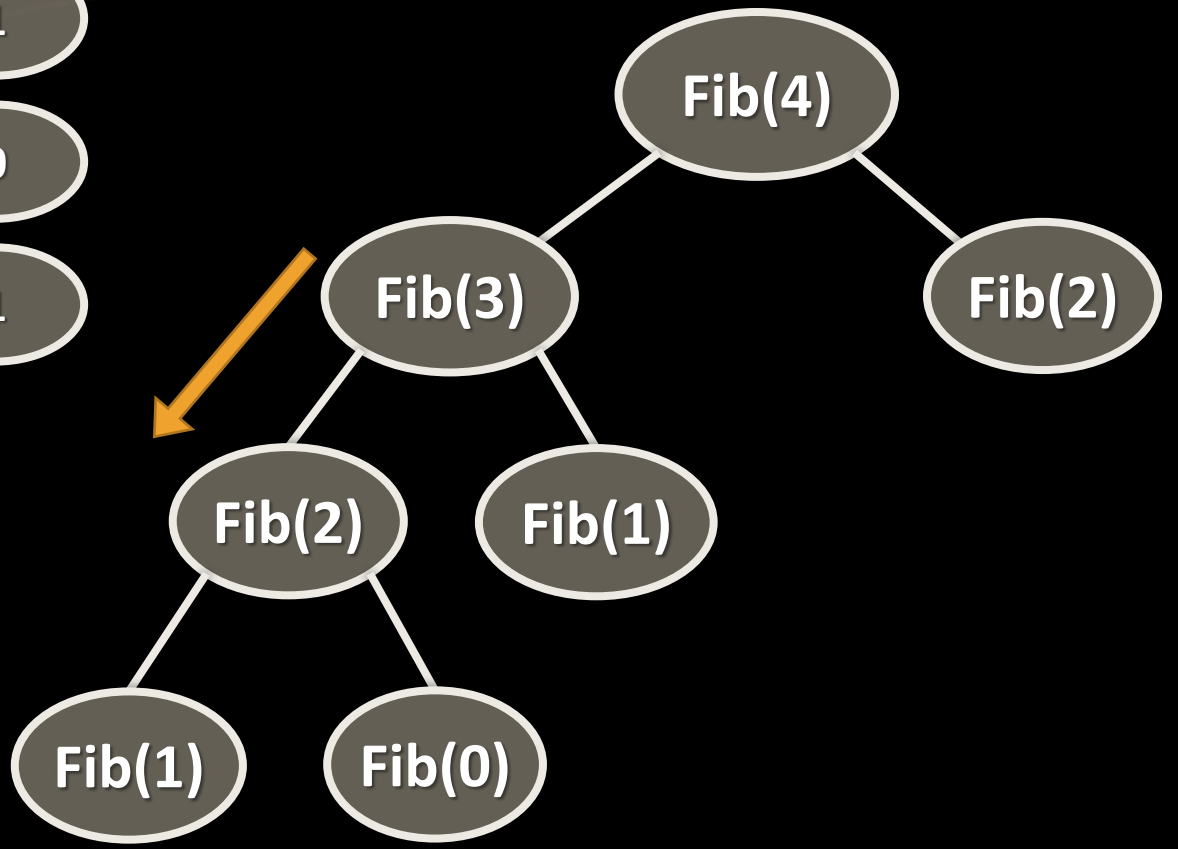




0	1	2	3	4
0	1			

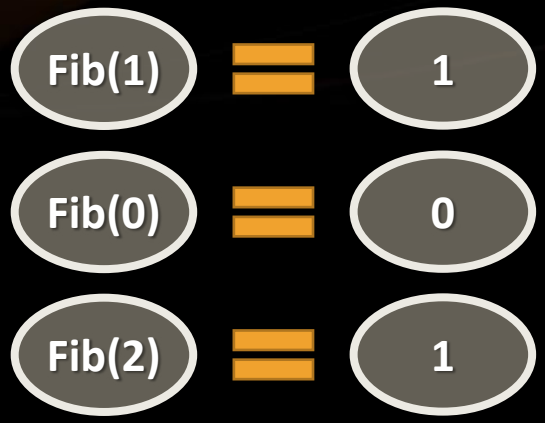
Fibonacci

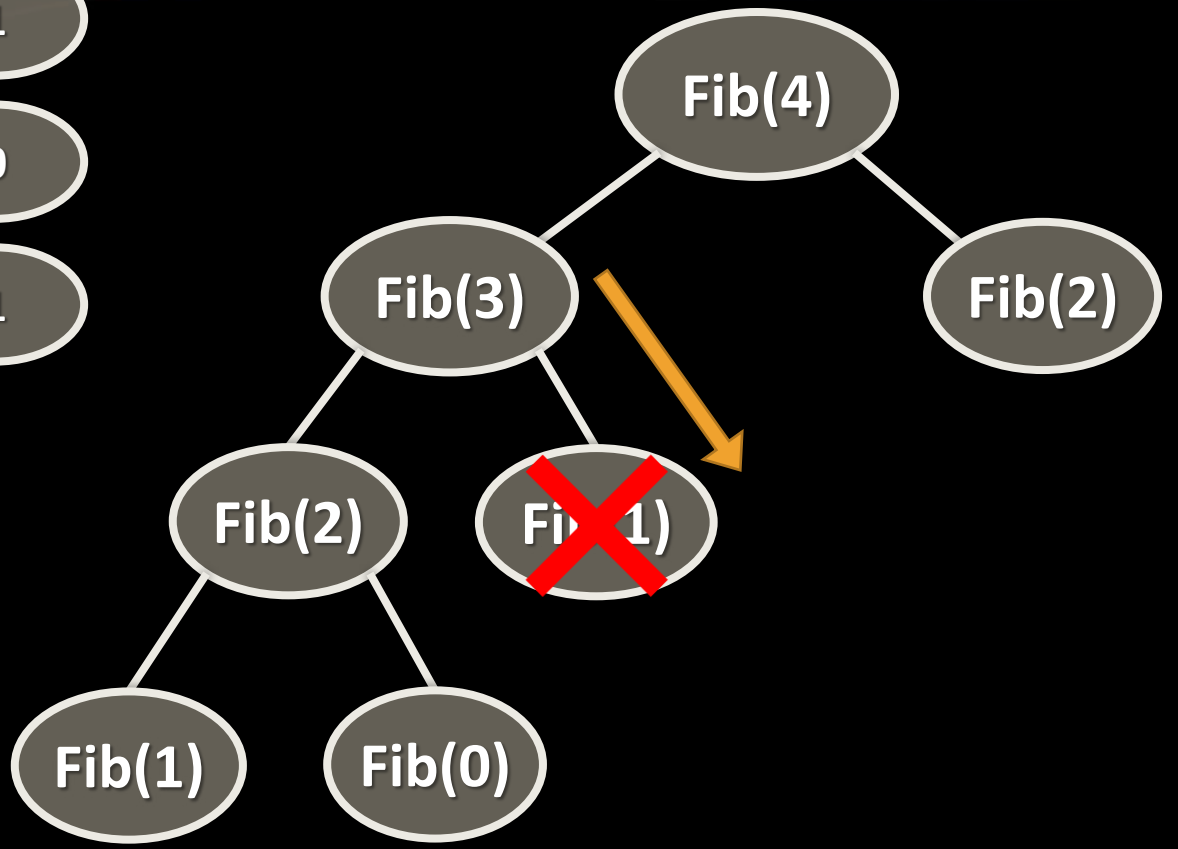




0	1	2	3	4
0	1	1		

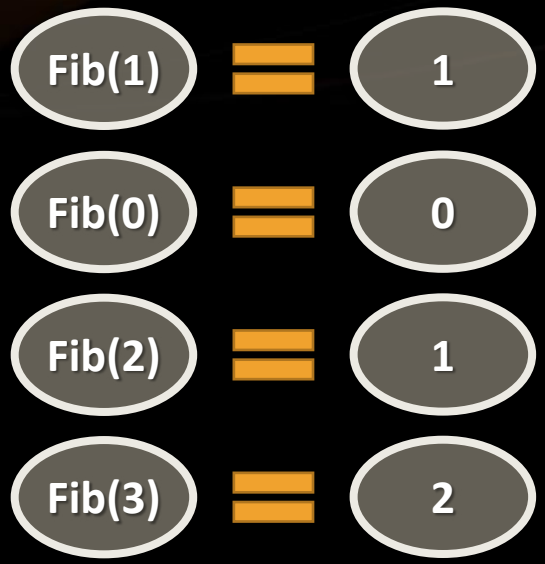
Fibonacci

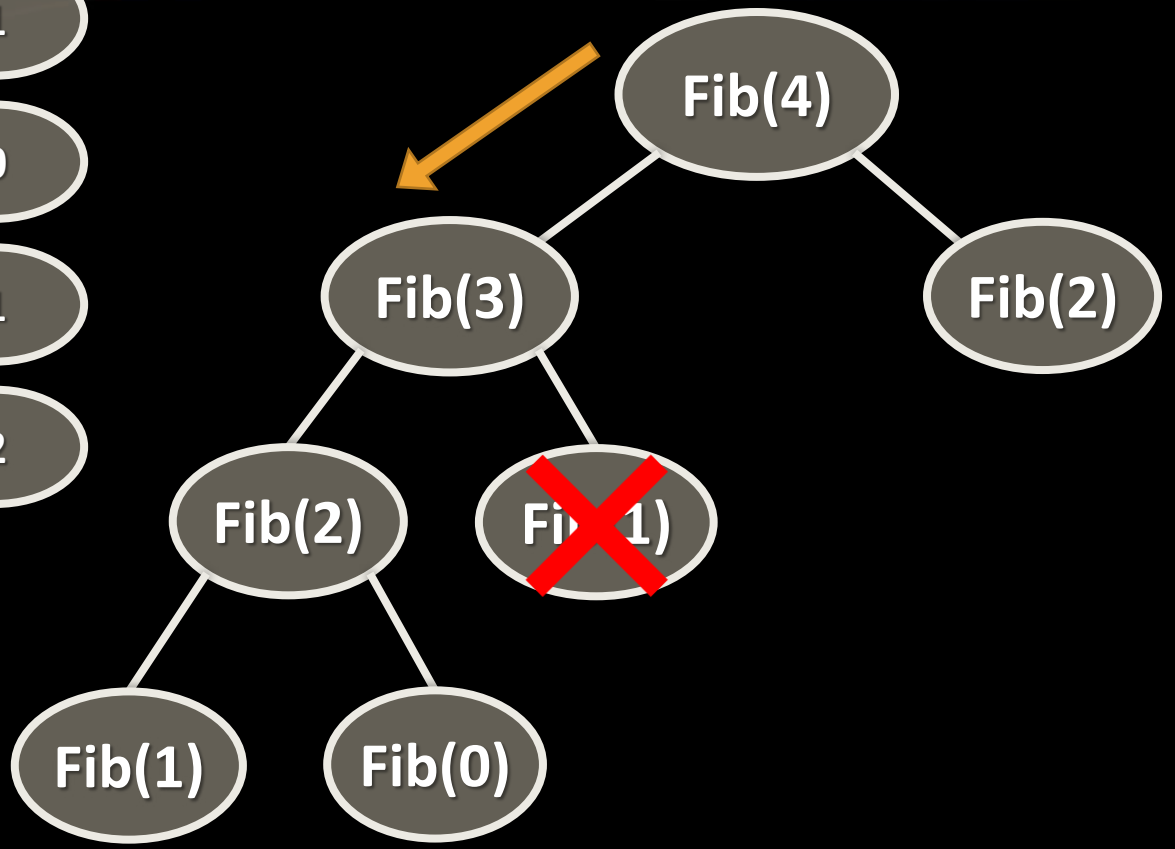




0	1	2	3	4
0	1	1		

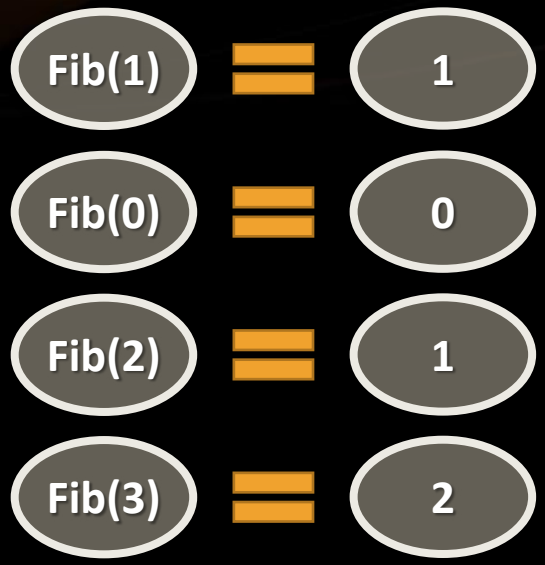
Fibonacci

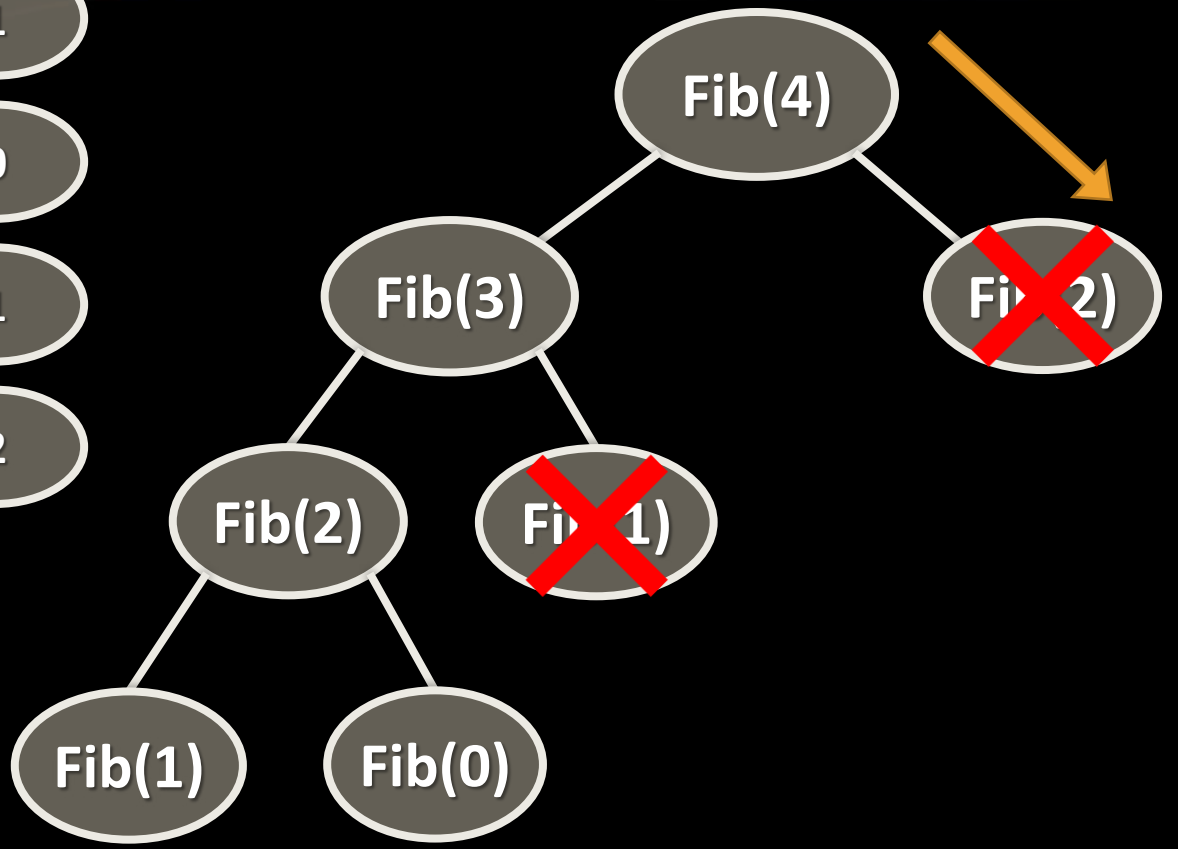




0	1	2	3	4
0	1	1	2	

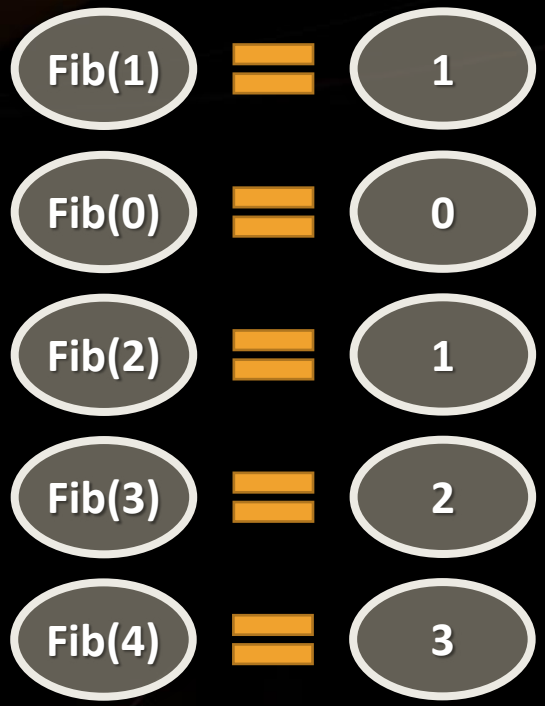
Fibonacci

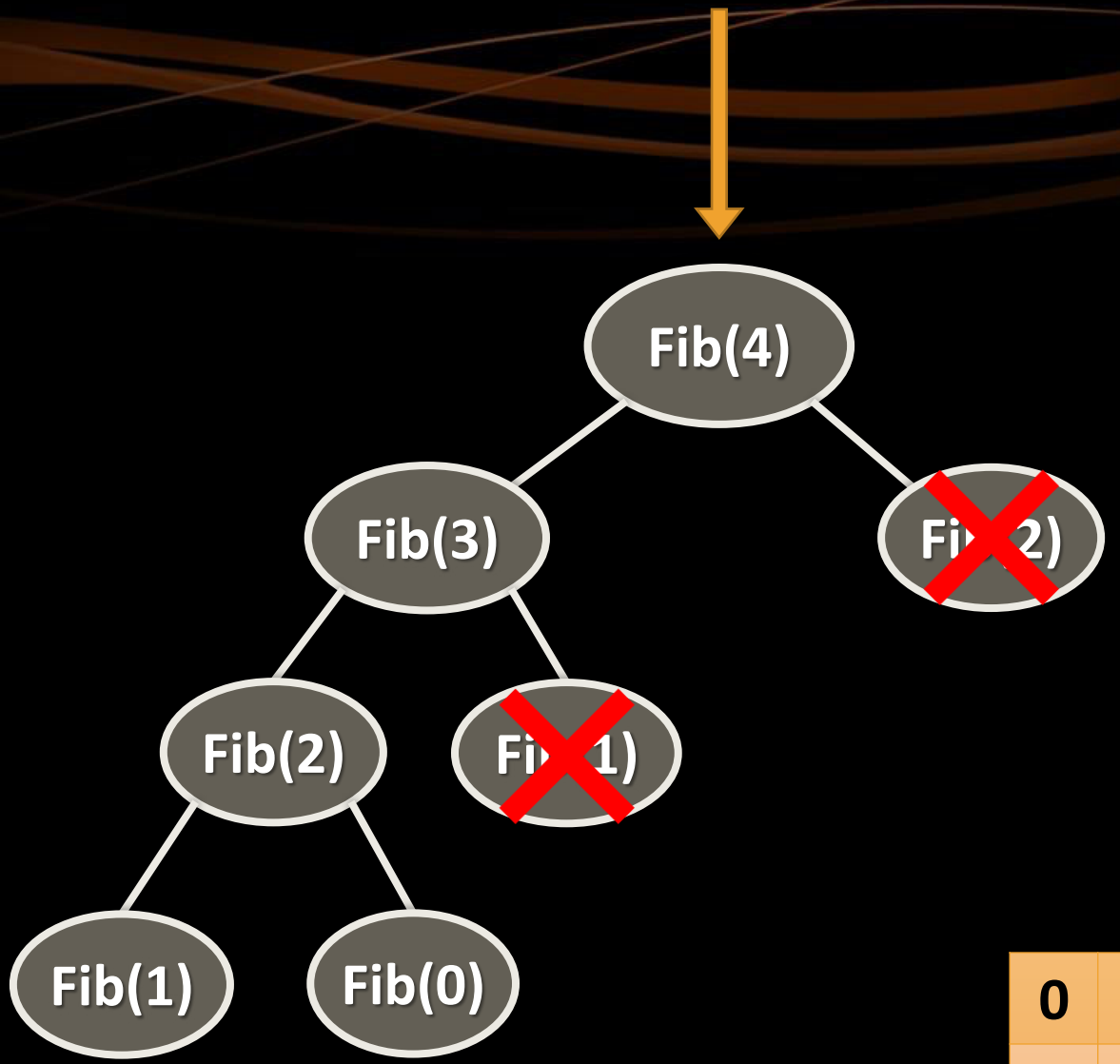




0	1	2	3	4
0	1	1	2	

Fibonacci





0	1	2	3	4
0	1	1	2	3

Top Down vs Bottom Up

SoftUni Foundation

- **Top down** approach
  - Solve **recursively** by **breaking down** the problem further and further
- **Bottom up** approach
  - Solve **iteratively** by solving smaller problems and **constructing the whole solution from the bottom up**

SoftUni Foundation

Fibonacci Sequence

Iterative Bottom-Up Approach

Fibonacci

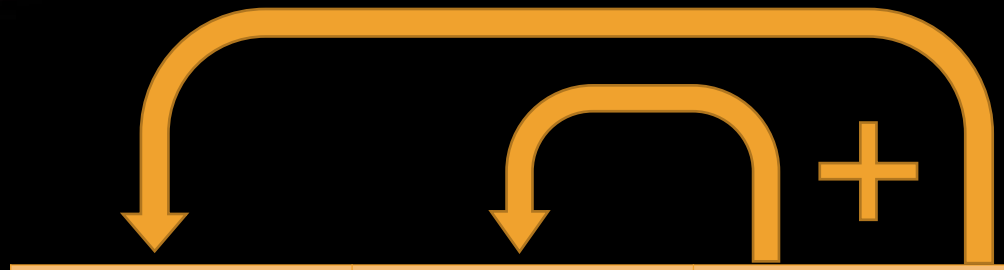
SoftUni Foundation

- Fib(4)

0	1	2	3	4
0	1			

Fibonacci

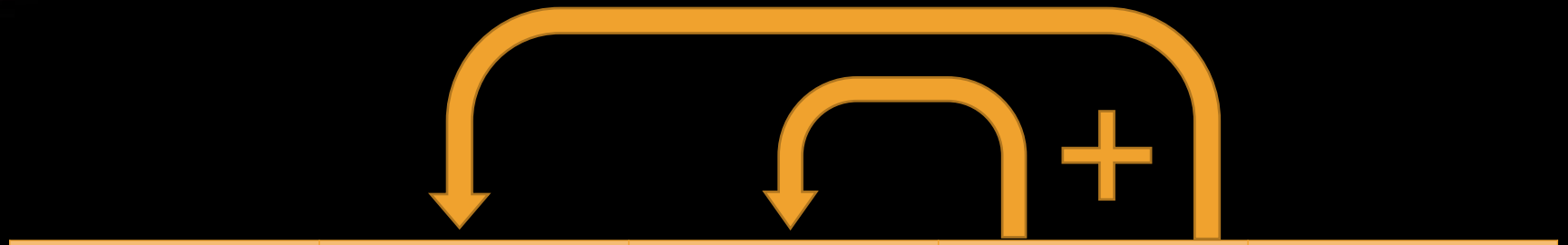
- Fib(4)



0	1	2	3	4
0	1	1		

Fibonacci

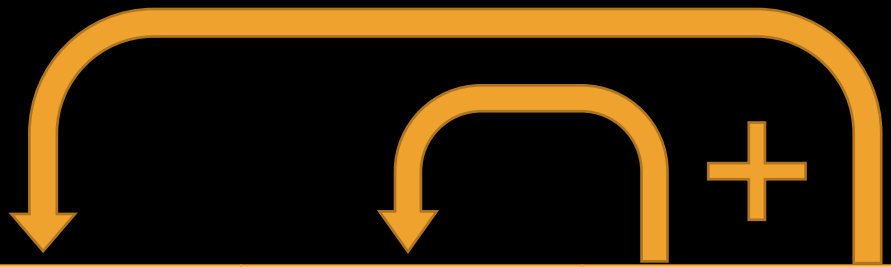
- Fib(4)



0	1	2	3	4
0	1	1	2	

Fibonacci

- Fib(4)



0	1	2	3	4
0	1	1	2	3

Compare Fibonacci Solutions

SoftUni Foundation

- Recursive Fibonacci (divide-and-conquer, no memoization)
  - Complexity: ~ **O(1.6^n)**
- **Top-down** dynamic programming (recursive with memorization)
  - Complexity: ~ **O(n)**
- **Bottom-up** dynamic programming (iterative)
  - Complexity: ~ **O(n)**
- If we want to find the 36^{th} Fibonacci number:
  - Recursive solution takes ~ **48 315 633** steps
  - Dynamic programming solution takes ~ **36** steps

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

Longest Increasing Subsequence

Finding and Reconstructing LIS

Longest Increasing Subsequence (LIS)

SoftUni Foundation

- Goal: find the largest subsequence of increasing numbers within a given sequence
- This subsequence is not necessarily contiguous, or unique
- Example:
  - {**3**, **5**, 8, **6**, **7**} → {3, 5, 6, 7}

SoftUni Foundation

LIS Optimal Substructure

Recursive Top-Down Approach

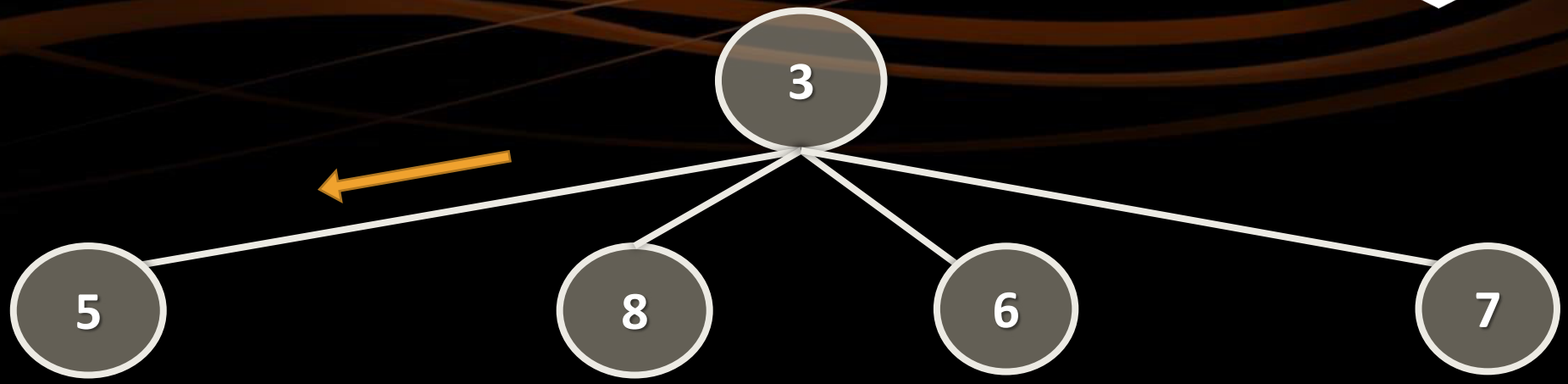
Optimal Sub-Structure

SoftUni Foundation

- **Break up** the problem into **sub-problems**
- Characterize the structure of an **optimal solution**
- **Compute** the optimal solutions, typically in a bottom-up fashion
- Construct an **optimal solution** from computed information

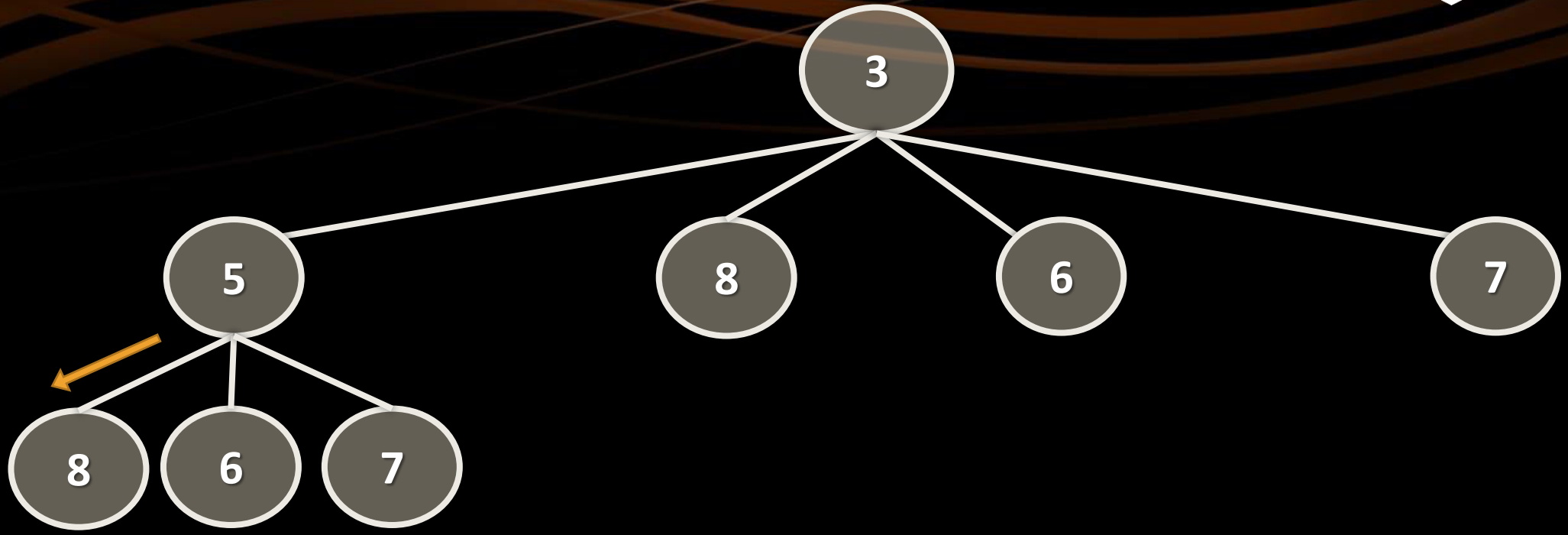
Optimal sub-structure

LIS – Optimal SS



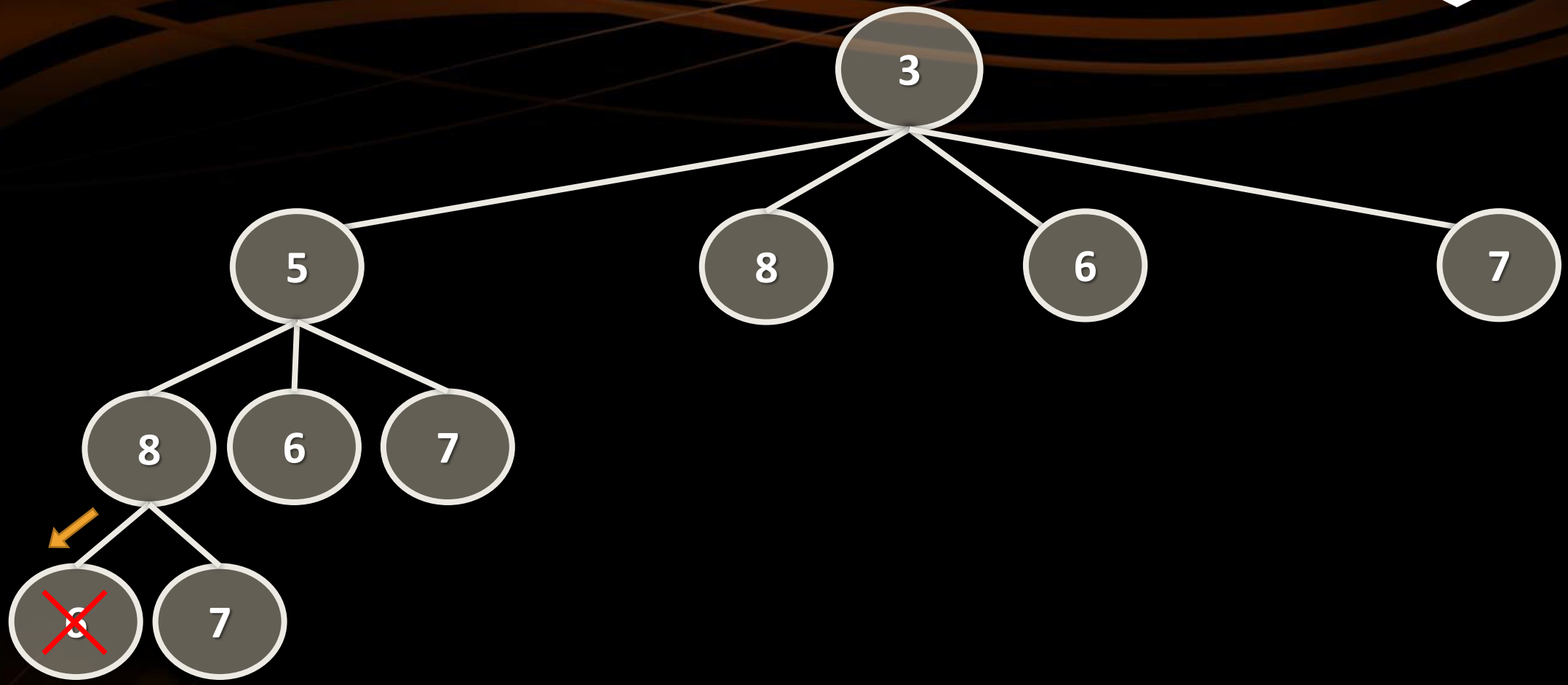
{3, 5, 8, 6, 7}

LIS – Optimal SS



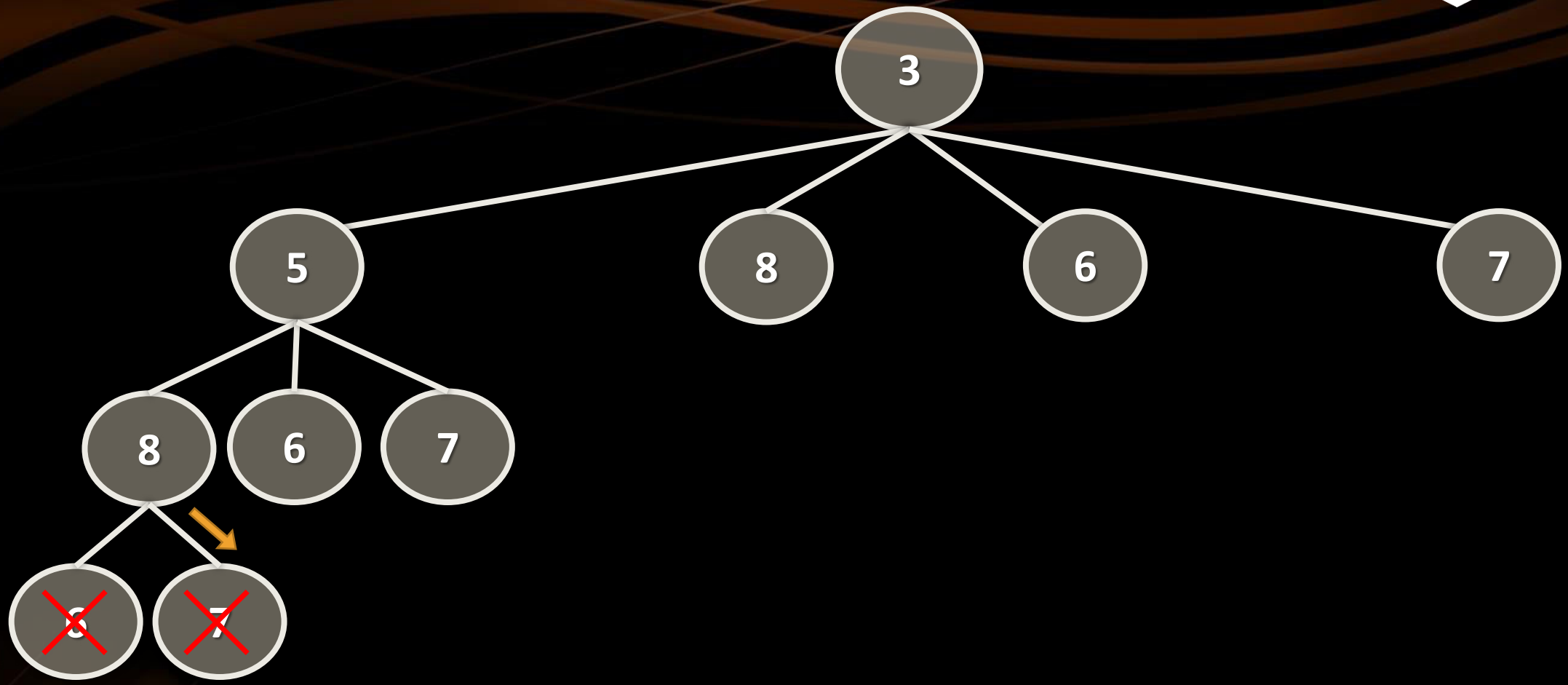
{3, 5, 8, 6, 7}

LIS – Optimal SS



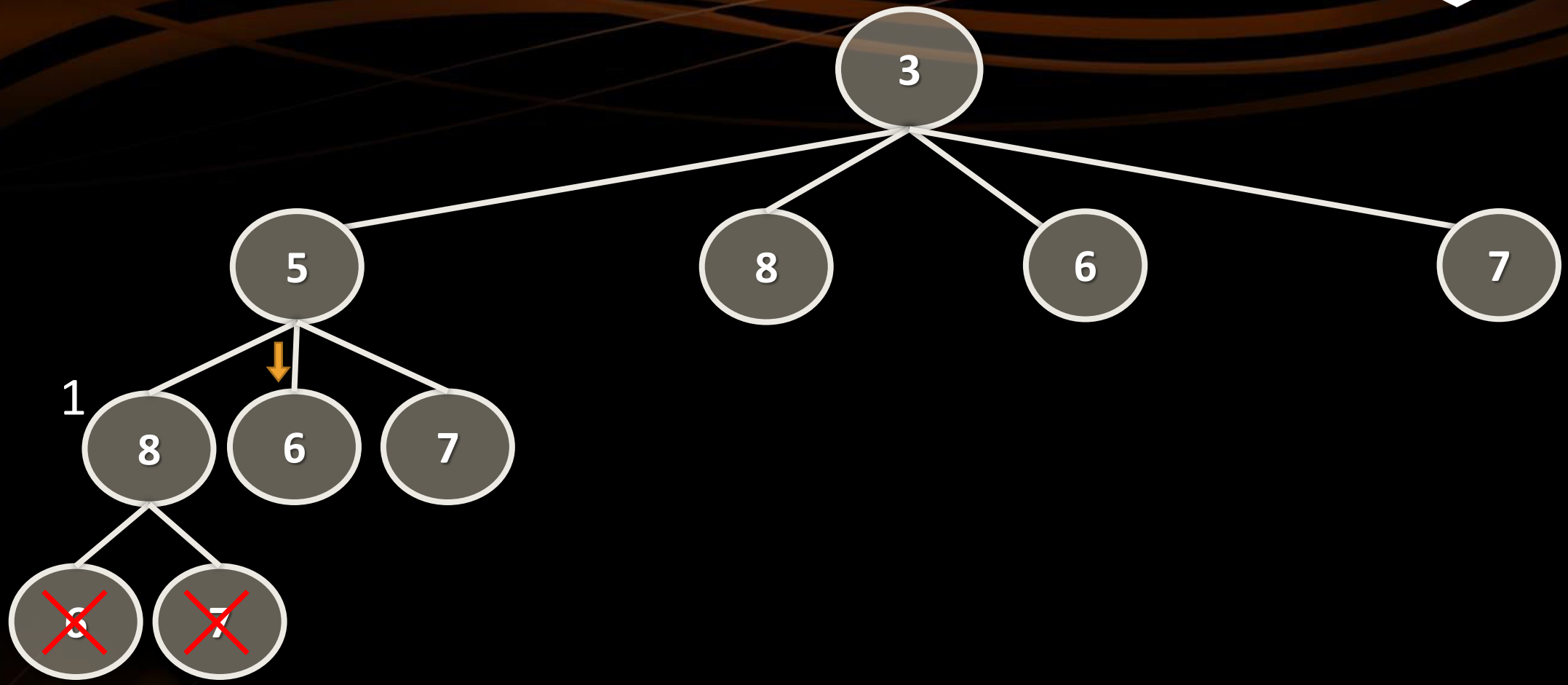
{3, 5, 8, 6, 7}

LIS – Optimal SS



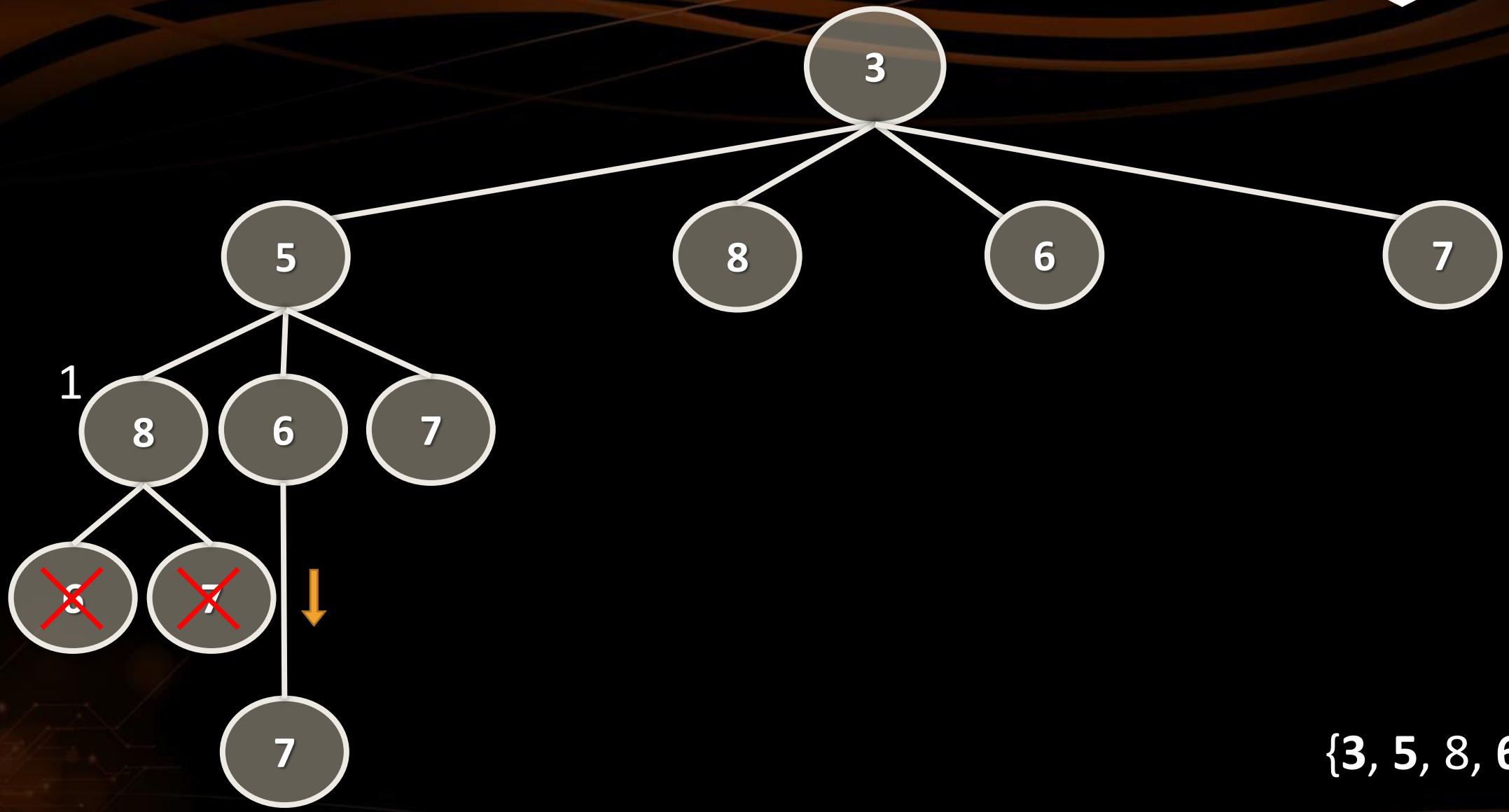
{3, 5, 8, 6, 7}

LIS – Optimal SS



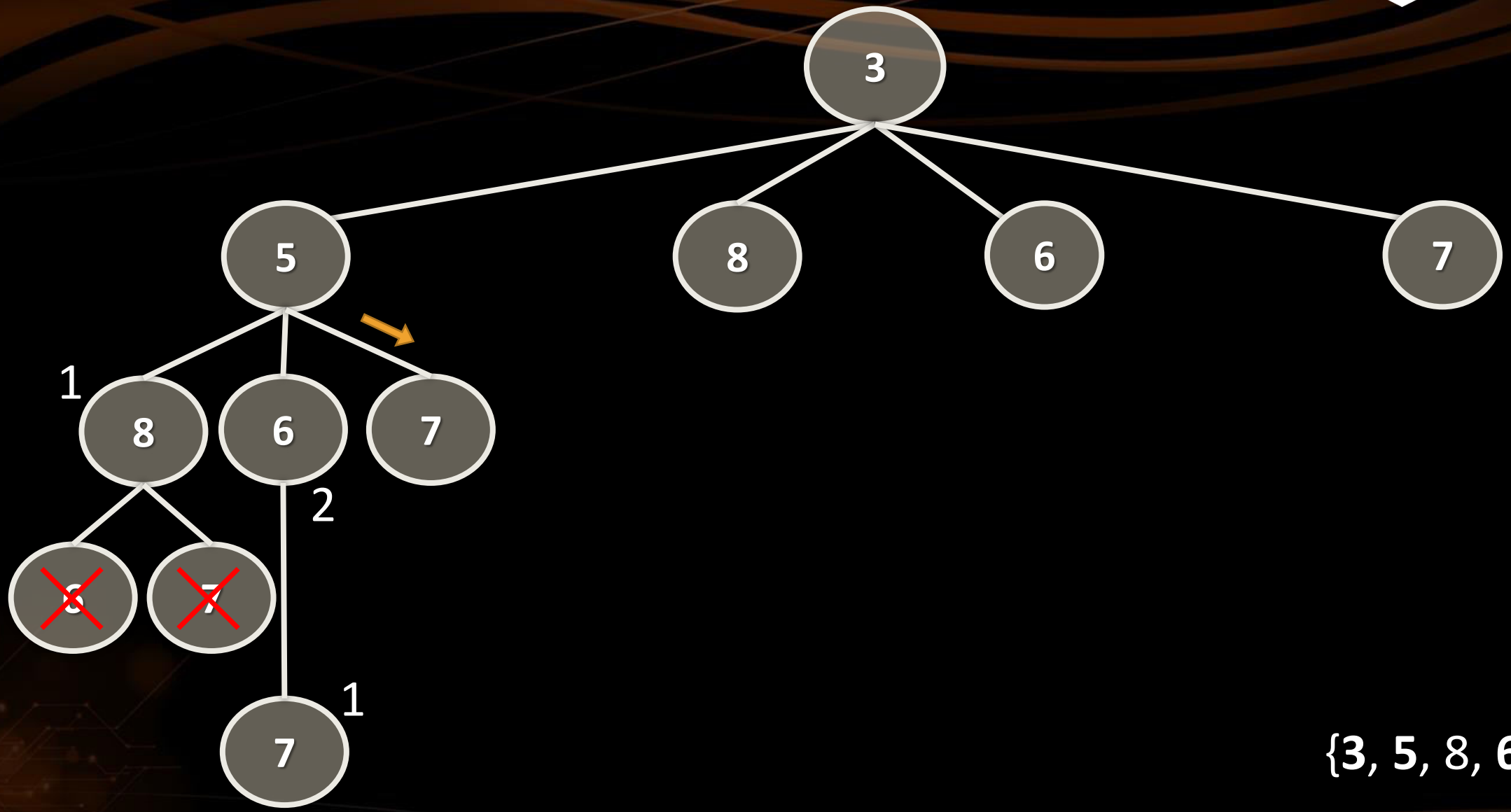
{3, 5, 8, 6, 7}

LIS – Optimal SS



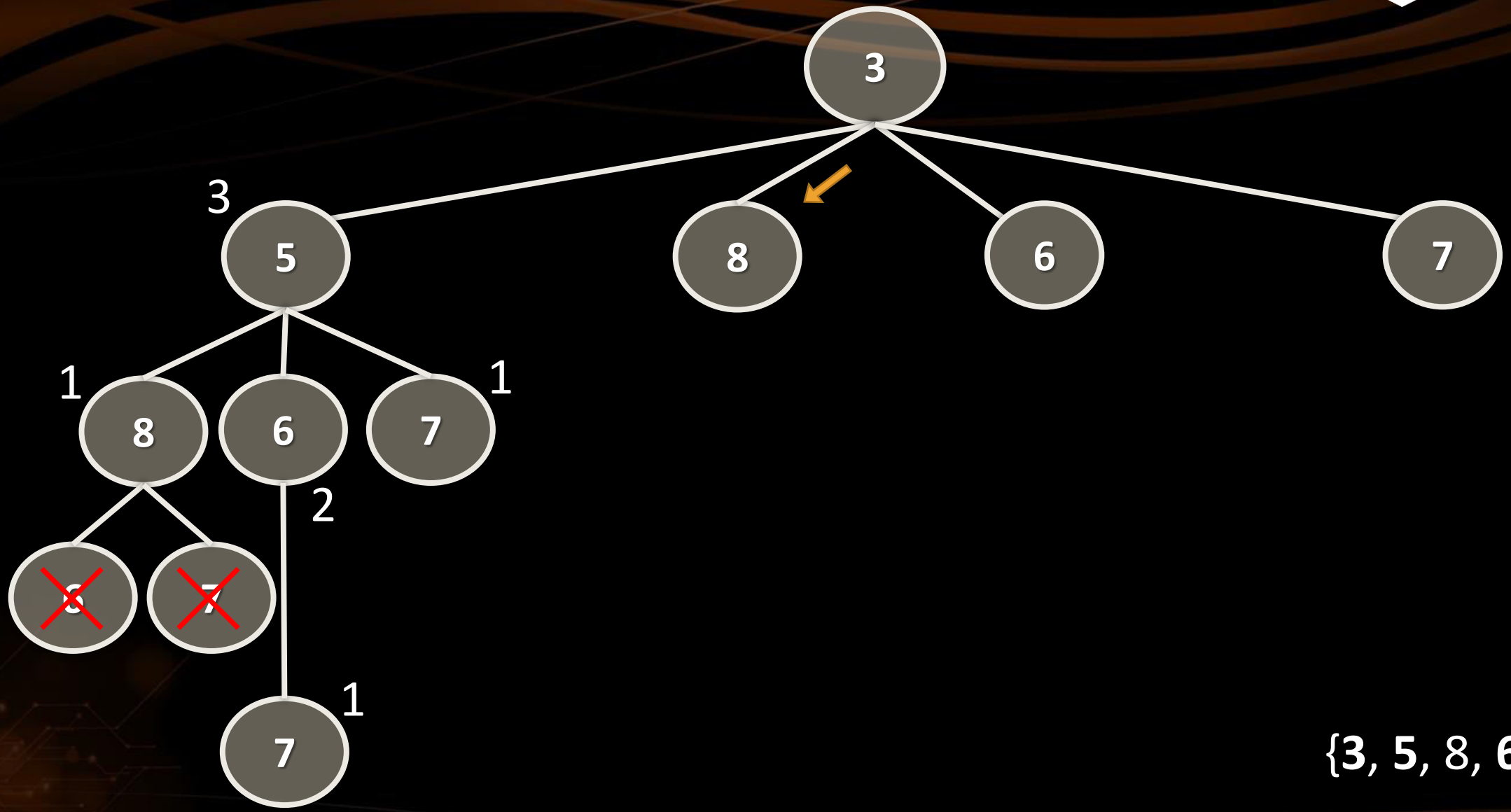
{3, 5, 8, 6, 7}

LIS – Optimal SS



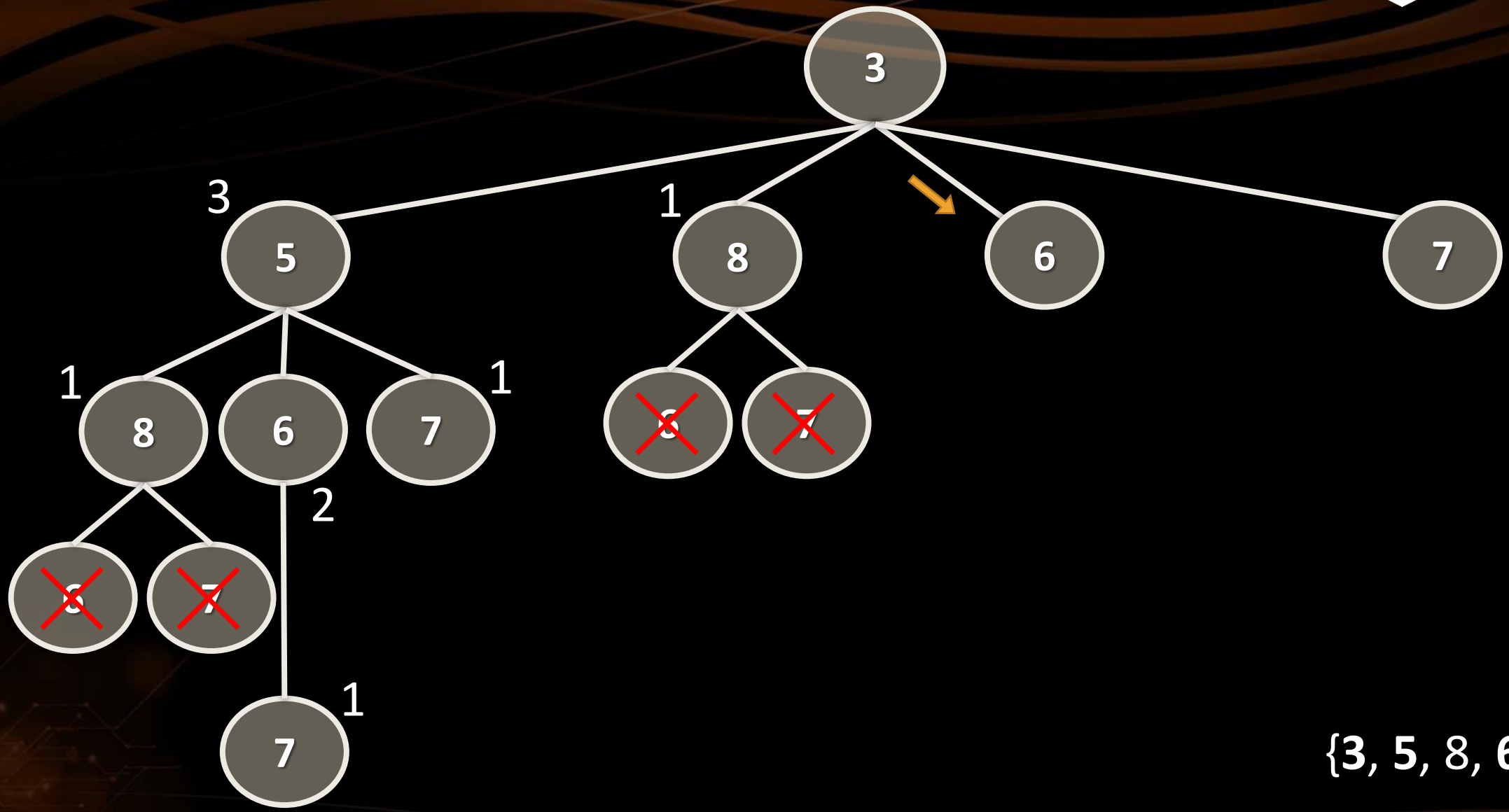
{3, 5, 8, 6, 7}

LIS – Optimal SS



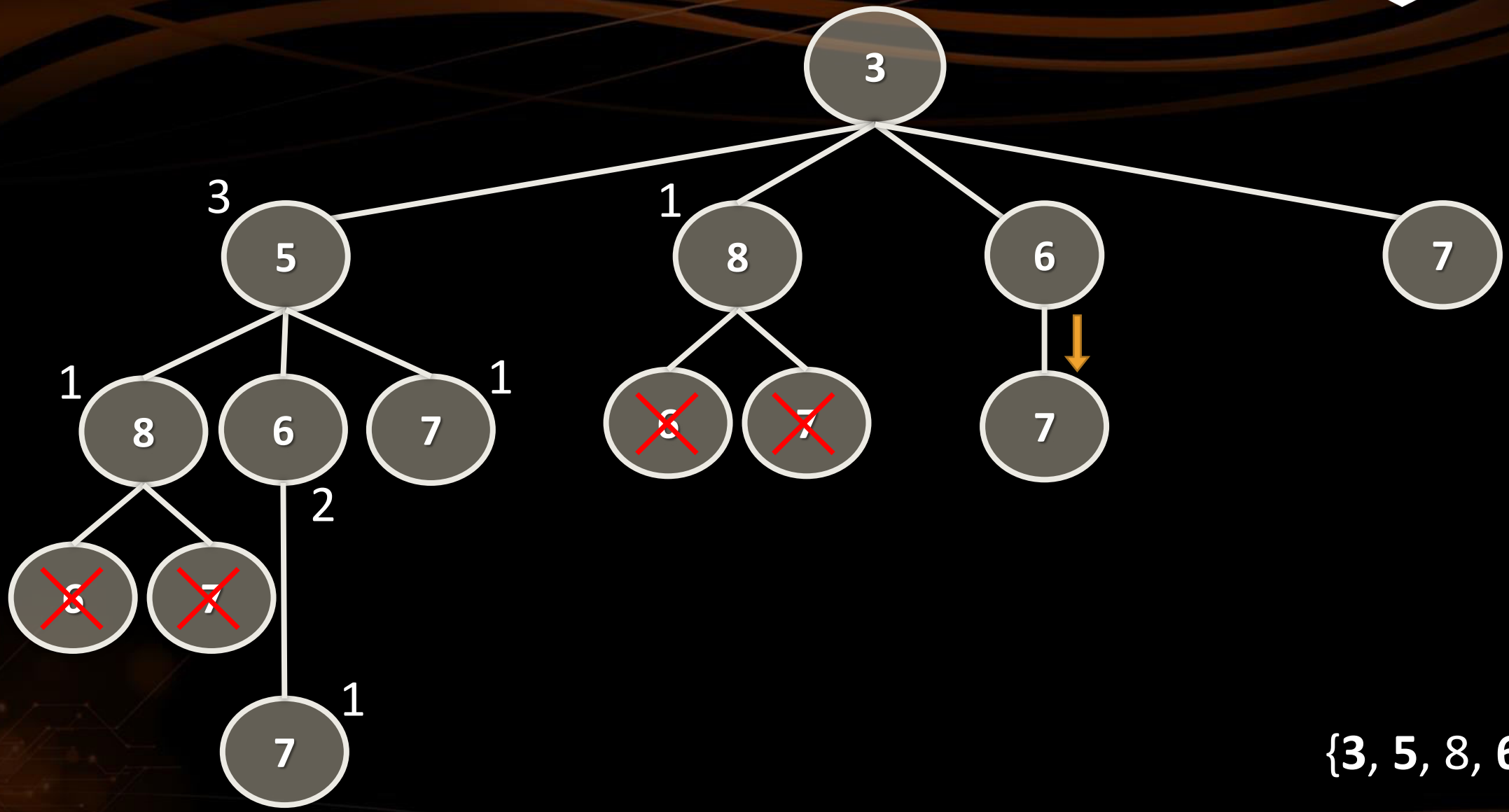
{3, 5, 8, 6, 7}

LIS – Optimal SS



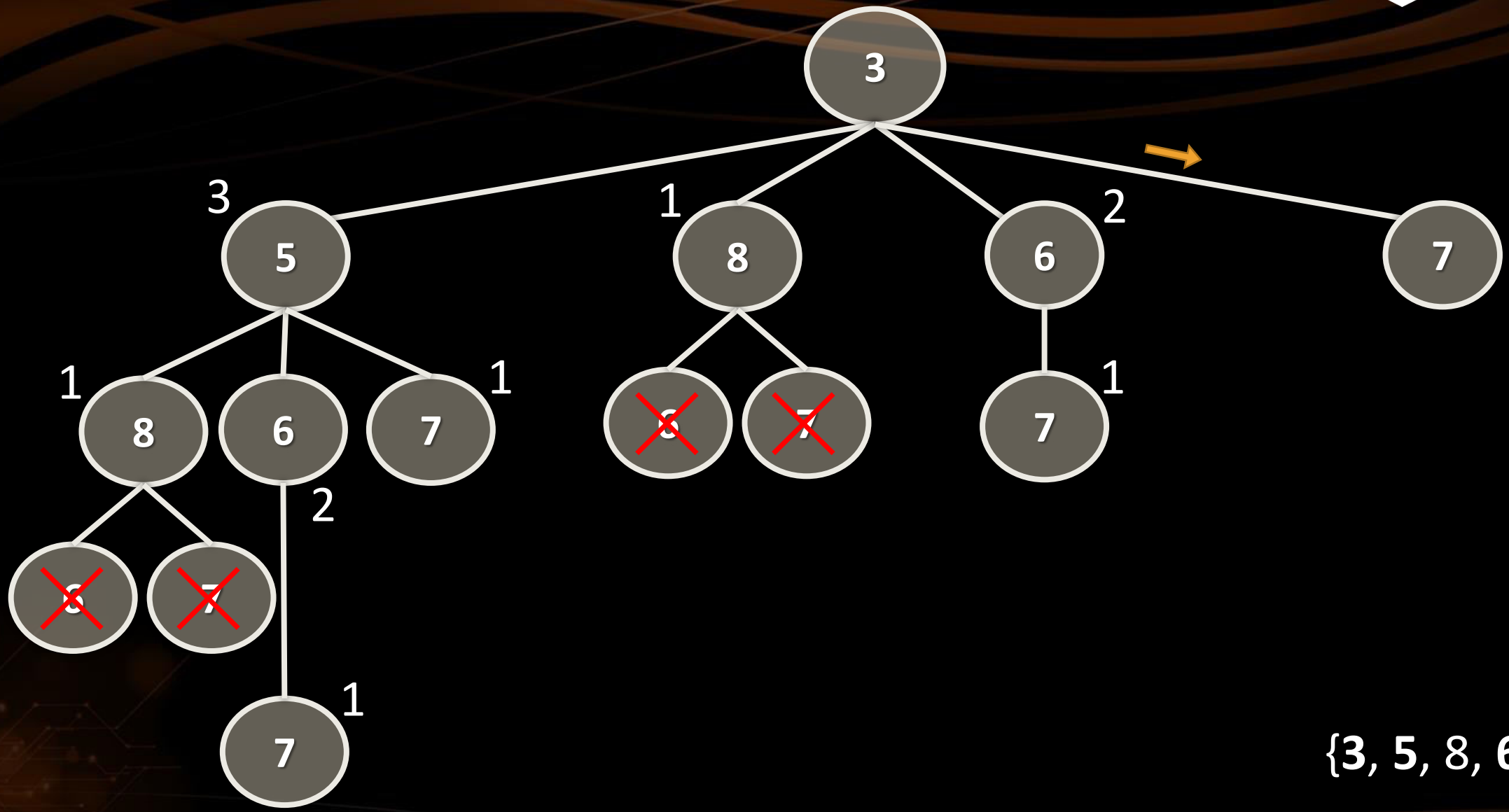
{3, 5, 8, 6, 7}

LIS – Optimal SS



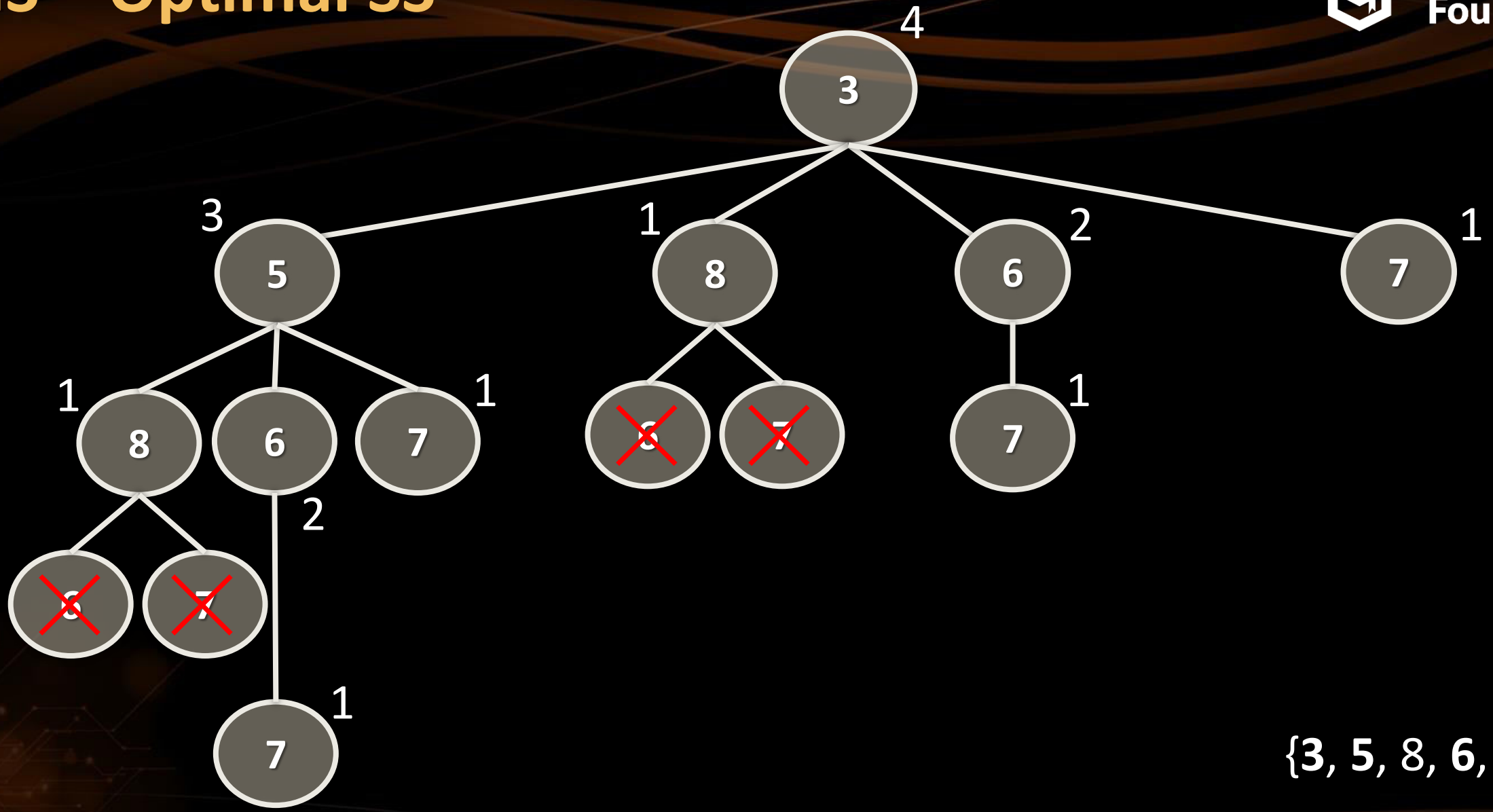
{3, 5, 8, 6, 7}

LIS – Optimal SS



{3, 5, 8, 6, 7}

LIS – Optimal SS



{3, 5, 8, 6, 7}

Memoization

SoftUni Foundation

- DP → sub-problems **overlap**
- In order to **avoid solving** problems **multiple times**, memorize
  - **Memoization** → **save/cache** sub-problem solutions **for later use**

- Example:
  - **Save** the length of the LIS **starting/ending** with each number

	0	1	2	3	4
Number	3	5	8	6	7
Length of LIS	4	3	1	2	1

SoftUni Foundation

Longest Increasing Subsequence

Iterative Bottom-Up Approach

Longest Increasing Subsequence

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]											

Longest Increasing Subsequence

SoftUni Foundation

LIS	3

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1										

Longest Increasing Subsequence

SoftUni Foundation

LIS	3, 14

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2									

Longest Increasing Subsequence

SoftUni Foundation

LIS	3, 5

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2								

Longest Increasing Subsequence

SoftUni Foundation

LIS	3, 5, 12

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3							

Longest Increasing Subsequence

SoftUni Foundation

LIS	3, 5, 12, 15

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4						

Longest Increasing Subsequence

SoftUni Foundation

LIS	3, 5, 7

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3					

Longest Increasing Subsequence

SoftUni Foundation

LIS	3, 5, 7, 8

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3	4				

Longest Increasing Subsequence

SoftUni Foundation

LIS	3, 5, 7, 8, 9

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3	4	5			

Longest Increasing Subsequence

SoftUni Foundation

LIS	3, 5, 7, 8, 9, 11

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3	4	5	6		

Longest Increasing Subsequence

SoftUni Foundation

LIS	3, 5, 7, 8, 9, 10

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3	4	5	6	6	

Longest Increasing Subsequence

SoftUni Foundation

LIS	1

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3	4	5	6	6	1

Longest Increasing Subsequence

SoftUni Foundation

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3	4	5	6 ✓	6 ✓	1

Longest Increasing Subsequence

SoftUni Foundation

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3	4	5	6	6	1

LIS	{**3**}	{3,**14**}	{3,**5**}	{3,5,**12**}	{3,5,12,**15**}	{3,5,**7**}	{3,5,7,**8**}	{3,5,7,8,**9**}	{3,5,7,8,9,**11**}	{3,5,7,8,9,**10**}	{**1**}

Reconstructing LIS

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]											

index	0	1	2	3	4	5	6	7	8	9	10
prev[]											

Reconstructing LIS

LIS	3

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1										

index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1										

Reconstructing LIS

SoftUni Foundation

LIS	3, 14

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2									

index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0									

Reconstructing LIS

SoftUni Foundation

LIS	3, 5

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2								

index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0								

Reconstructing LIS

SoftUni Foundation

LIS	3, 5, 12

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3							

index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2							

Reconstructing LIS

SoftUni Foundation

LIS	3, 5, 12, 15

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4						

index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3						

Reconstructing LIS

SoftUni Foundation

LIS	3, 5, 7

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3					

index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2					

Reconstructing LIS

SoftUni Foundation

LIS	3, 5, 7, 8

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3	4				

index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5				

Reconstructing LIS

SoftUni Foundation

LIS	3, 5, 7, 8, 9

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3	4	5			

index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6			

Reconstructing LIS

SoftUni Foundation

LIS	3, 5, 7, 8, 9, 11

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3	4	5	6		

index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7		

Reconstructing LIS

SoftUni Foundation

LIS	3, 5, 7, 8, 9, 10

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3	4	5	6	6	

index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7	7	

Reconstructing LIS

SoftUni Foundation

LIS	1

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3	4	5	6	6	1

index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7	7	-1

Reconstructing LIS - Right-Most Solution

SoftUni Foundation

LIS	10

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3	4	5	6	6	1

index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7	7	-1

Calculating LIS – Source Code

SoftUni Foundation

```
int[] seq = { 3, 4, 8, 1, 2, 4, 32, 6, 2, 5, 33, 4, 38, 22 };
int[] len = new int[seq.Length];
for (int x = 0; x < seq.Length; x++)
{
  len[x] = 1;
  for (int i = 0; i <= x - 1; i++)
    if (seq[i] < seq[x] && len[i] + 1 > len[x])
      len[x] = 1 + len[i];
}
```

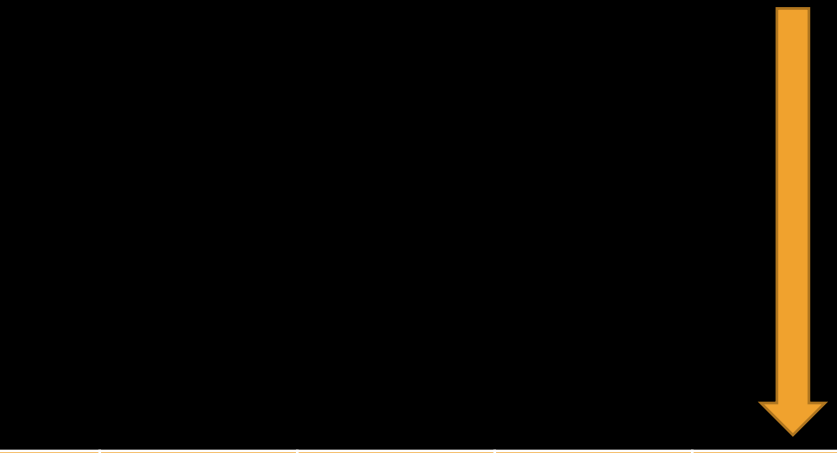
index	0	1	2	3	4	5	6	7	8	9	10
seq[]	**3**	**14**	**5**	**12**	**15**	**7**	**8**	**9**	**11**	**10**	**1**
len[]	1	2	2	3	4	3	4	5	6	6	1

Reconstructing LIS - Right-Most Solution

SoftUni Foundation

LIS	10

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1



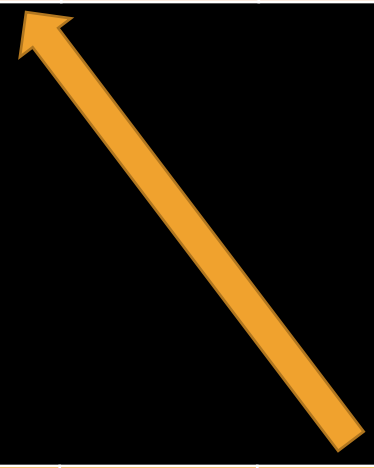
index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7	7	-1

Reconstructing LIS - Right-Most Solution

SoftUni Foundation

LIS	10, 9

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1



index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7	7	-1

Reconstructing LIS - Right-Most Solution

SoftUni Foundation

LIS	10, 9

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

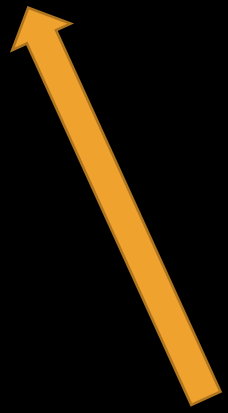
index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7	7	-1

Reconstructing LIS - Right-Most Solution

SoftUni Foundation

LIS	10, 9, 8

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1



index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7	7	-1

Reconstructing LIS - Right-Most Solution

SoftUni Foundation

LIS	10, 9, 8

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

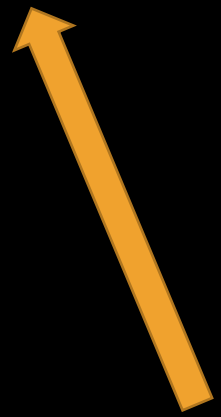
index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7	7	-1

Reconstructing LIS - Right-Most Solution

SoftUni Foundation

LIS	10, 9, 8, 7

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1



index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7	7	-1

Reconstructing LIS - Right-Most Solution

SoftUni Foundation

LIS	10, 9, 8, 7

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

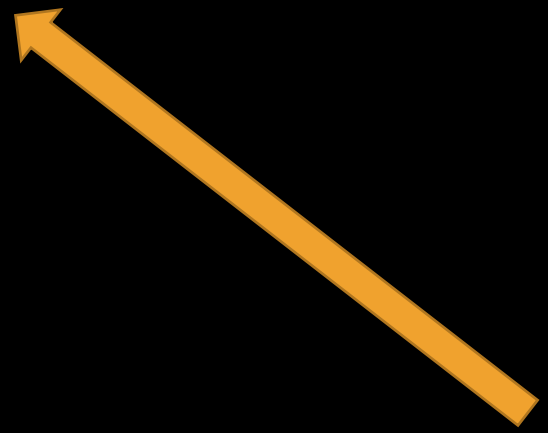
index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7	7	-1

Reconstructing LIS - Right-Most Solution

SoftUni Foundation

LIS	10, 9, 8, 7, 5

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1



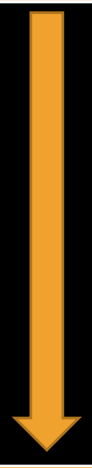
index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7	7	-1

Reconstructing LIS - Right-Most Solution

SoftUni Foundation

LIS	10, 9, 8, 7, 5

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1



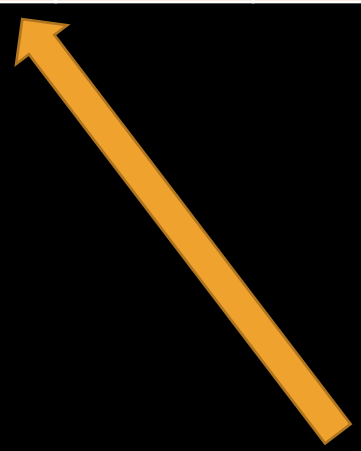
index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7	7	-1

Reconstructing LIS - Right-Most Solution

SoftUni Foundation

LIS	10, 9, 8, 7, 5, 3

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1



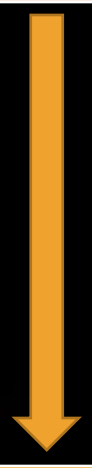
index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7	7	-1

Reconstructing LIS - Right-Most Solution

SoftUni Foundation

LIS	10, 9, 8, 7, 5, 3

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1



index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7	7	-1

Reconstructing LIS - Right-Most Solution

SoftUni Foundation

LIS	10, 9, 8, 7, 5, 3

Reverse →

LIS	3, 5, 7, 8, 9, 10

index	0	1	2	3	4	5	6	7	8	9	10
seq[]	3	14	5	12	15	7	8	9	11	10	1

index	0	1	2	3	4	5	6	7	8	9	10
len[]	1	2	2	3	4	3	4	5	6	6	1

index	0	1	2	3	4	5	6	7	8	9	10
prev[]	-1	0	0	2	3	2	5	6	7	7	-1

Calculating LIS with Previous – Source Code

SoftUni Foundation

```
int maxLen = 0;
int lastIndex = -1;
for (int x = 0; x < seq.Length; x++)
{
  len[x] = 1;
  prev[x] = -1;
  for (int i = 0; i < x; i++)
    if ((seq[i] < seq[x]) && (len[i] + 1 > len[x]))
    {
      len[x] = len[i] + 1;
      prev[x] = i;
    }
  if (len[x] > maxLen)
  {
    maxLen = len[x];
    lastIndex = x;
  }
}
```

Restoring LIS Elements – Source Code

SoftUni Foundation

```
int[] RestoreLIS(int[] seq, int[] prev, int lastIndex)
{
  var longestSeq = new List<int>();
  while (lastIndex != -1)
  {
    longestSeq.Add(seq[lastIndex]);
    lastIndex = prev[lastIndex];
  }

  longestSeq.Reverse();

  return longestSeq.ToArray();
}
```

2	**6**	1	8	9	4	2
1	**8**	0	3	5	6	7
3	**4**	**8**	**7**	2	1	8
0	9	2	**8**	1	7	9
2	7	1	**9**	**7**	**8**	2
4	5	6	1	2	**5**	**6**
9	3	5	2	8	1	**9**
2	3	4	1	7	2	**8**

Move Down/Right Sum

Largest Sum in Matrix of Numbers

"Move Down / Right Sum" Problem

SoftUni Foundation

- You are given a matrix of numbers
  - Find the **path with largest sum**
  - Start → top left
  - End → bottom right
  - Move only right/down
  - There won't be negative numbers

2	**6**	1	8	9	4	2
1	**8**	0	3	5	6	7
3	**4**	**8**	**7**	2	1	8
0	9	2	**8**	1	7	9
2	7	1	**9**	**7**	**8**	2
4	5	6	1	2	**5**	**6**
9	3	5	2	8	1	**9**
2	3	4	1	7	2	**8**

SoftUni Foundation

Move D/R Optimal Substructure

Recursive Top-Down Approach

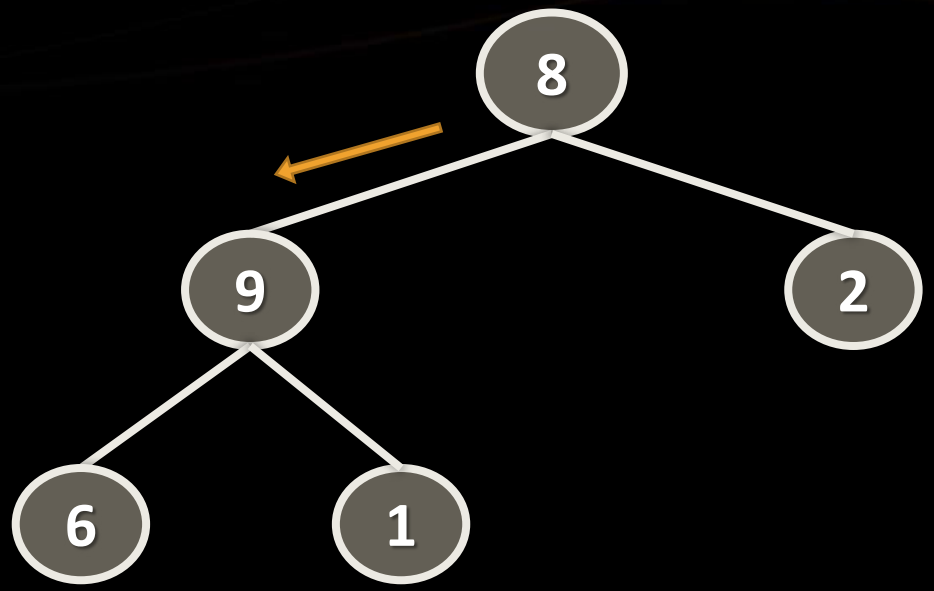
Move Down/Right

2	6	1	8	9	4	2
1	8	0	3	5	6	7
3	4	8	7	2	1	8
0	9	2	8	1	7	9
2	7	1	9	7	8	2
4	5	6	1	2	5	6
9	3	5	2	8	1	9
2	3	4	1	7	2	8

Move Down/Right

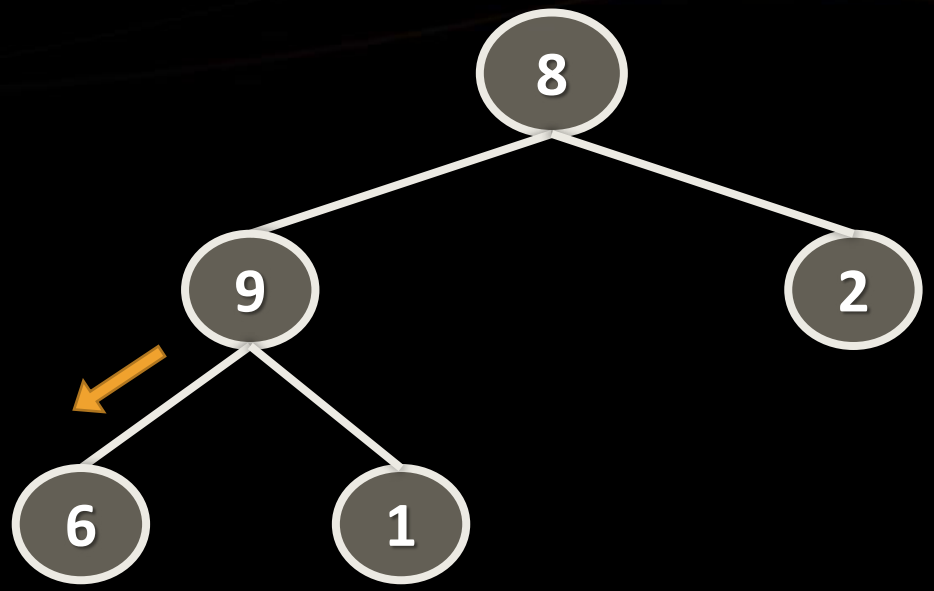
2	6	1	8	9	4	2
1	8	0	3	5	6	7
3	4	8	7	2	1	8
0	9	2	8	1	7	9
2	7	1	9	7	8	2
4	5	6	1	2	5	6
9	3	5	2	8	1	9
2	3	4	1	7	2	8

Move Down/Right



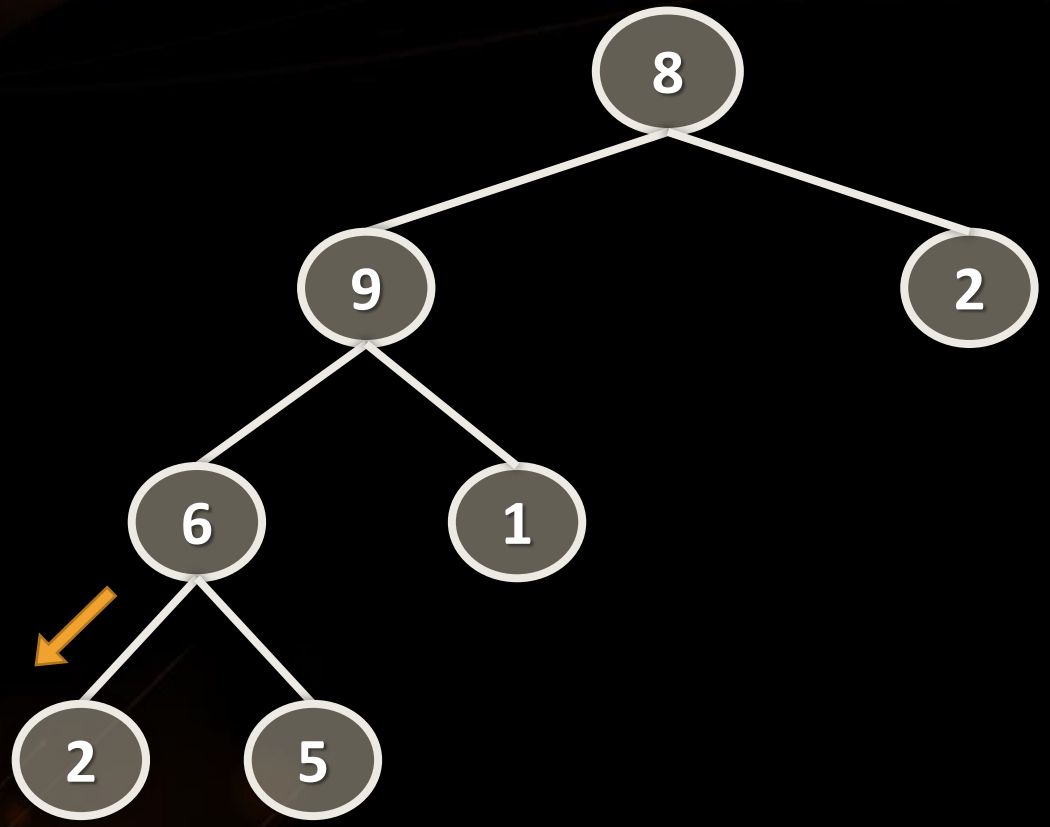
2	6	1	8	9	4	2
1	8	0	3	5	6	7
3	4	8	7	2	1	8
0	9	2	8	1	7	9
2	7	1	9	7	8	2
4	5	6	1	2	5	6
9	3	5	2	8	1	9
2	3	4	1	7	2	8

Move Down/Right



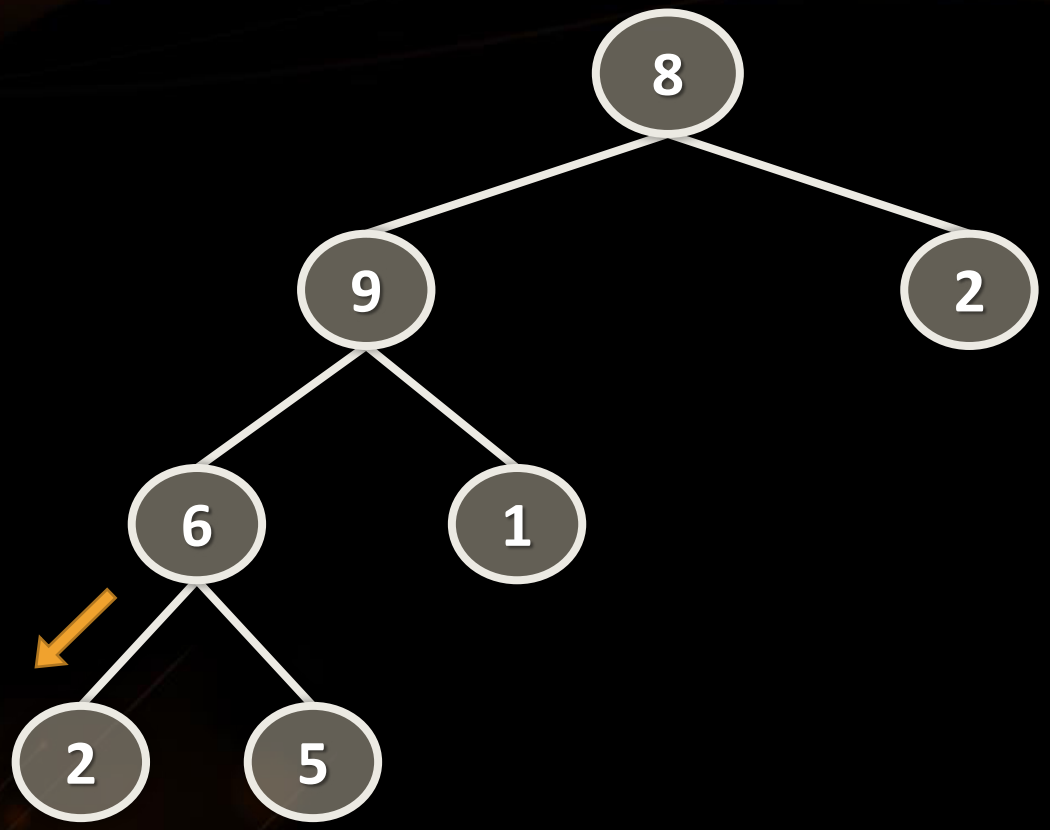
2	6	1	8	9	4	2
1	8	0	3	5	6	7
3	4	8	7	2	1	8
0	9	2	8	1	7	9
2	7	1	9	7	8	2
4	5	6	1	2	5	6
9	3	5	2	8	1	9
2	3	4	1	7	2	8

Move Down/Right



2	**6**	1	8	9	4	2
1	**8**	0	3	5	6	7
3	**4**	**8**	**7**	2	1	8
0	9	2	**8**	1	7	9
2	7	1	**9**	**7**	**8**	2
4	5	6	1	2	**5**	**6**
9	3	5	2	8	1	**9**
2	3	4	1	7	2	**8**

Move Down/Right



...

2	6	1	8	9	4	2
1	8	0	3	5	6	7
3	4	8	7	2	1	8
0	9	2	8	1	7	9
2	7	1	9	7	8	2
4	5	6	1	2	5	6
9	3	5	2	8	1	9
2	3	4	1	7	2	8

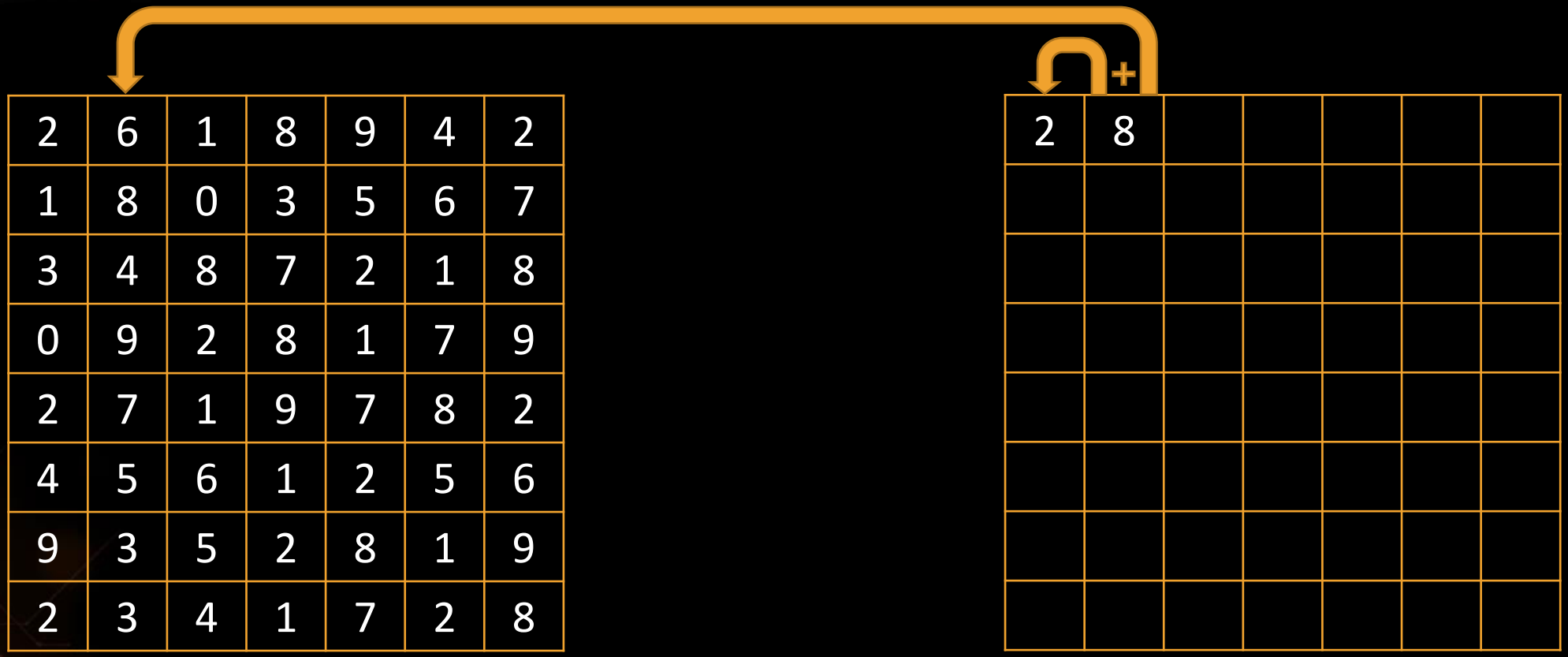
Building the DP Matrix

SoftUni Foundation

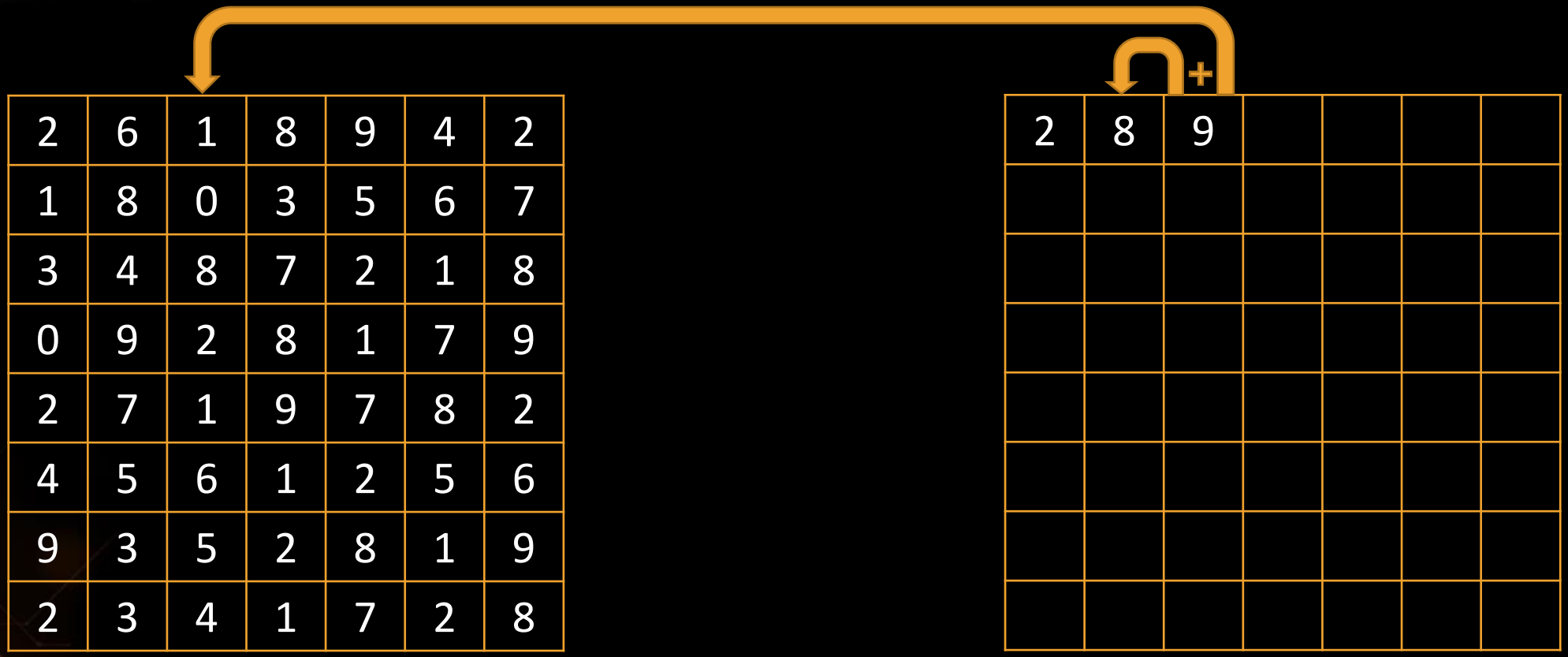
2	6	1	8	9	4	2
1	8	0	3	5	6	7
3	4	8	7	2	1	8
0	9	2	8	1	7	9
2	7	1	9	7	8	2
4	5	6	1	2	5	6
9	3	5	2	8	1	9
2	3	4	1	7	2	8

2						

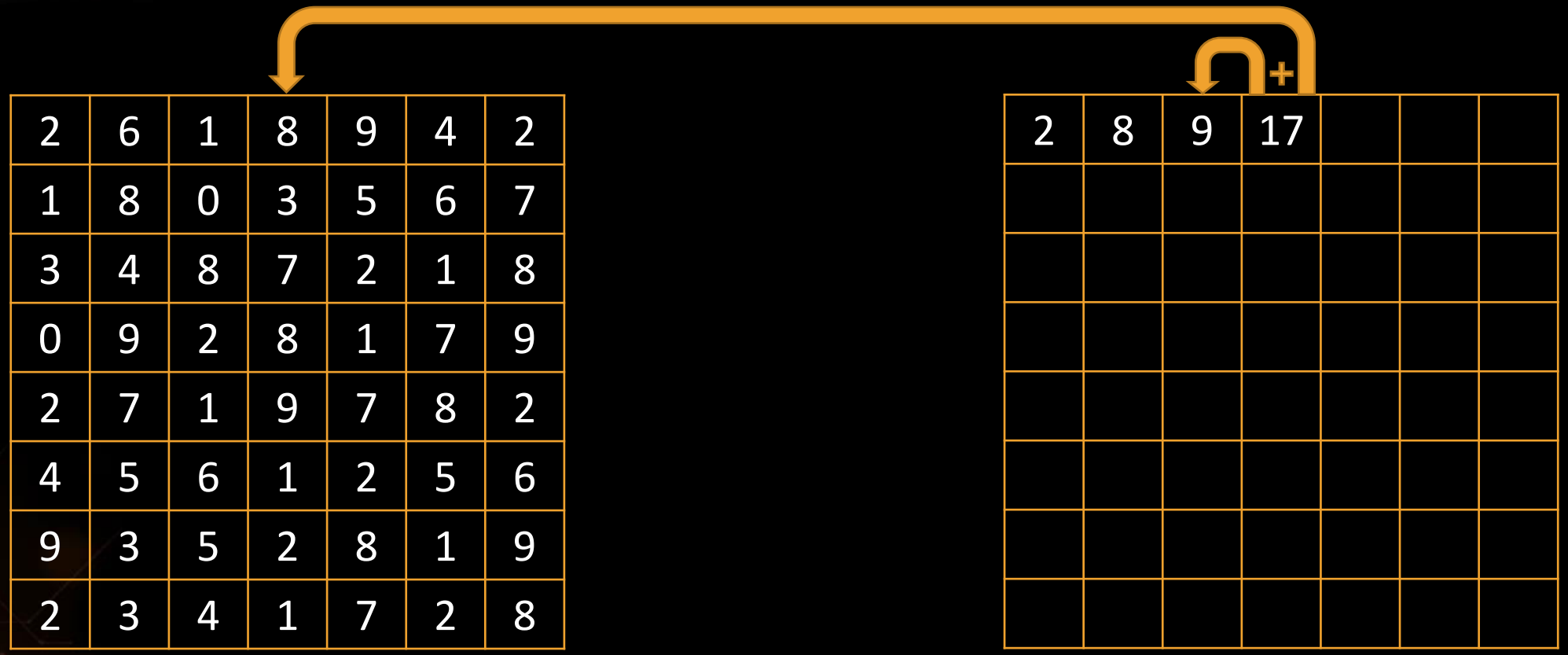
Building the DP Matrix



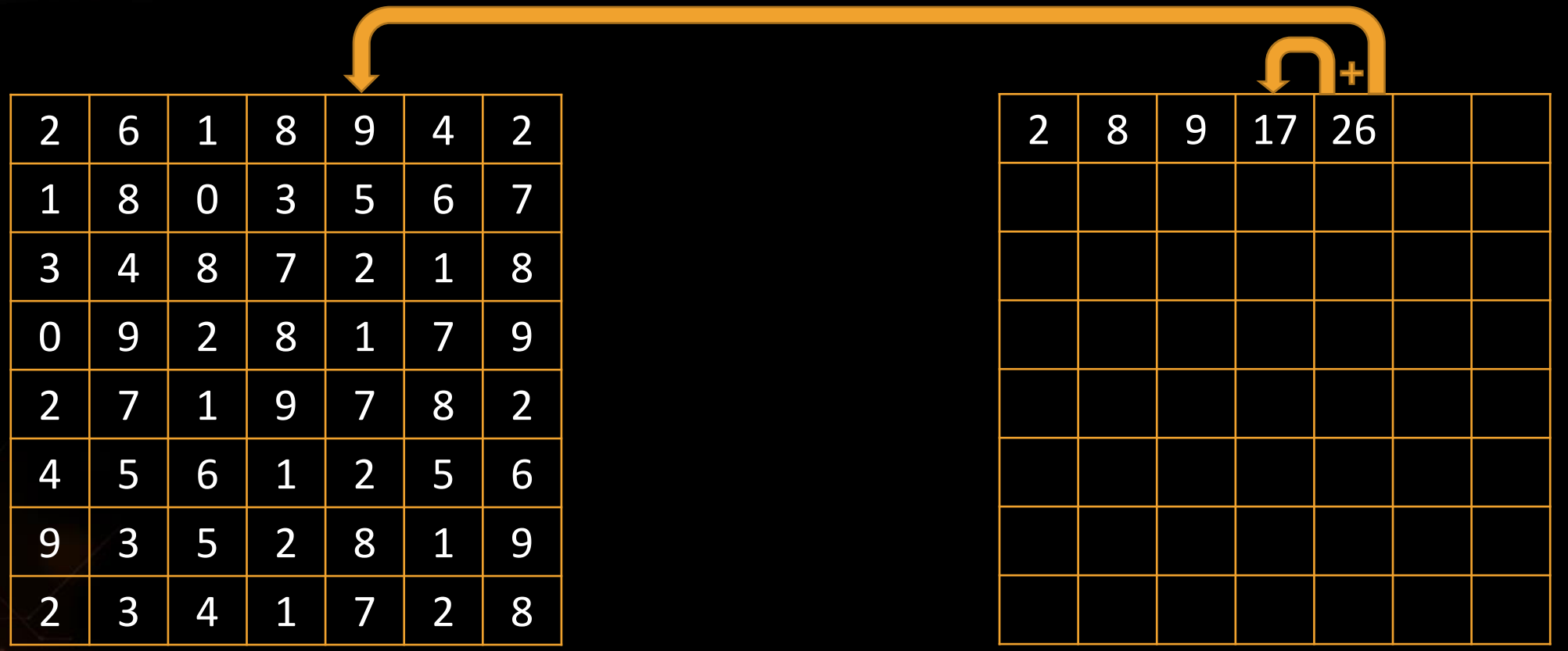
Building the DP Matrix



Building the DP Matrix



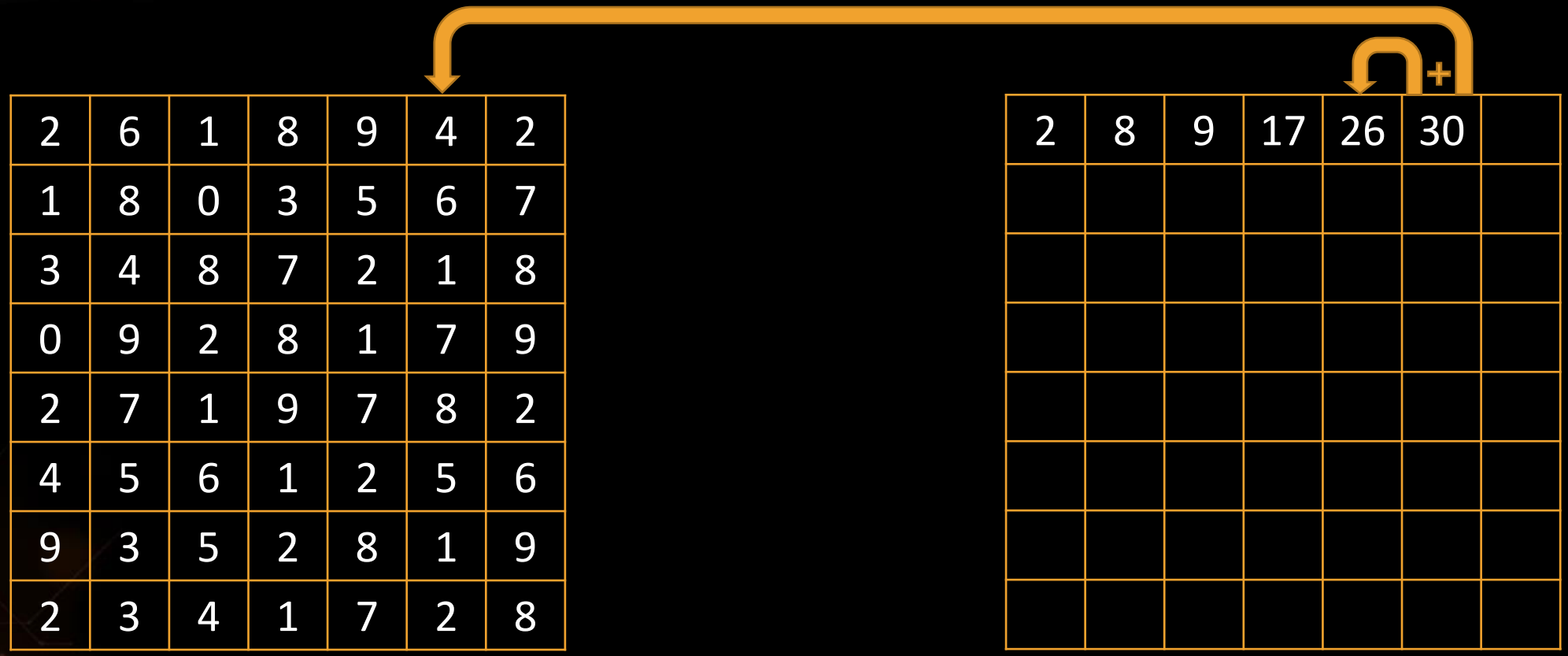
Building the DP Matrix



2	6	1	8	9	4	2
1	8	0	3	5	6	7
3	4	8	7	2	1	8
0	9	2	8	1	7	9
2	7	1	9	7	8	2
4	5	6	1	2	5	6
9	3	5	2	8	1	9
2	3	4	1	7	2	8

2	8	9	17	26		

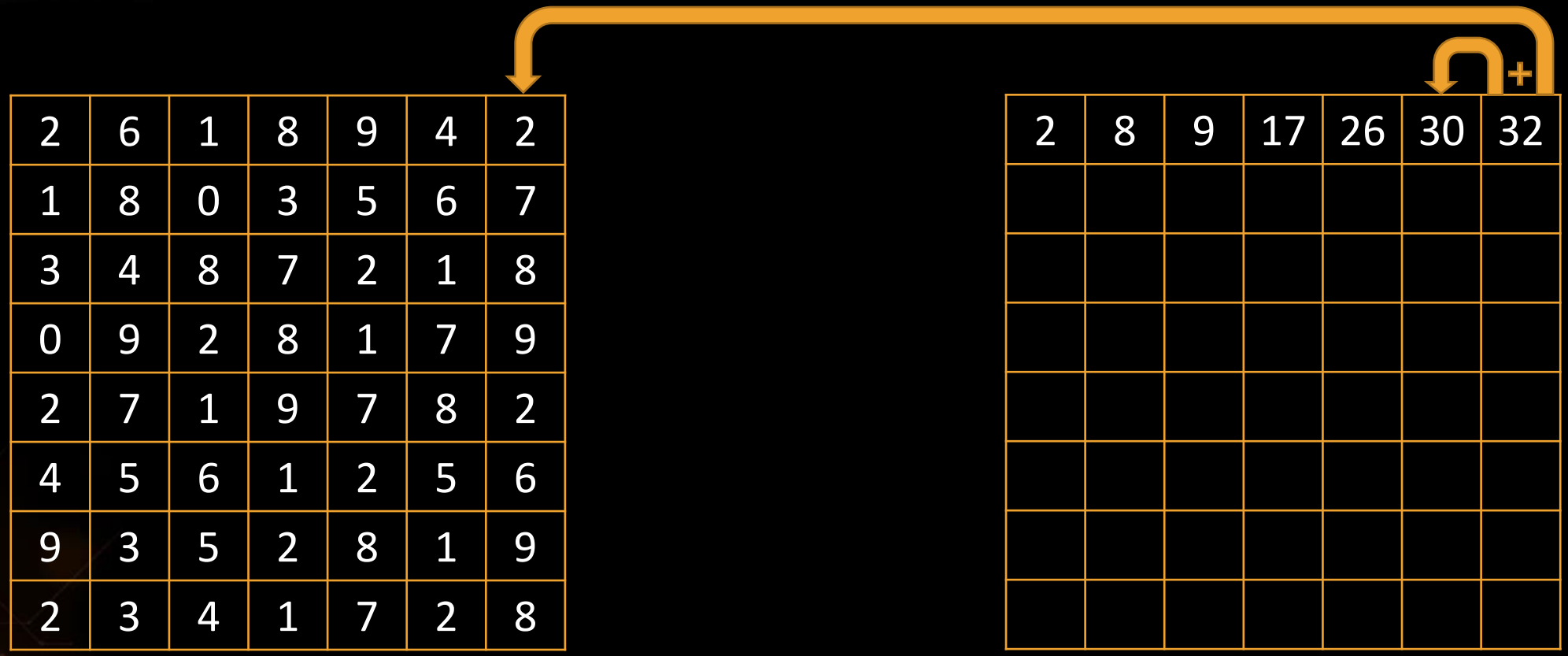
Building the DP Matrix



2	6	1	8	9	4	2
1	8	0	3	5	6	7
3	4	8	7	2	1	8
0	9	2	8	1	7	9
2	7	1	9	7	8	2
4	5	6	1	2	5	6
9	3	5	2	8	1	9
2	3	4	1	7	2	8

2	8	9	17	26	30	

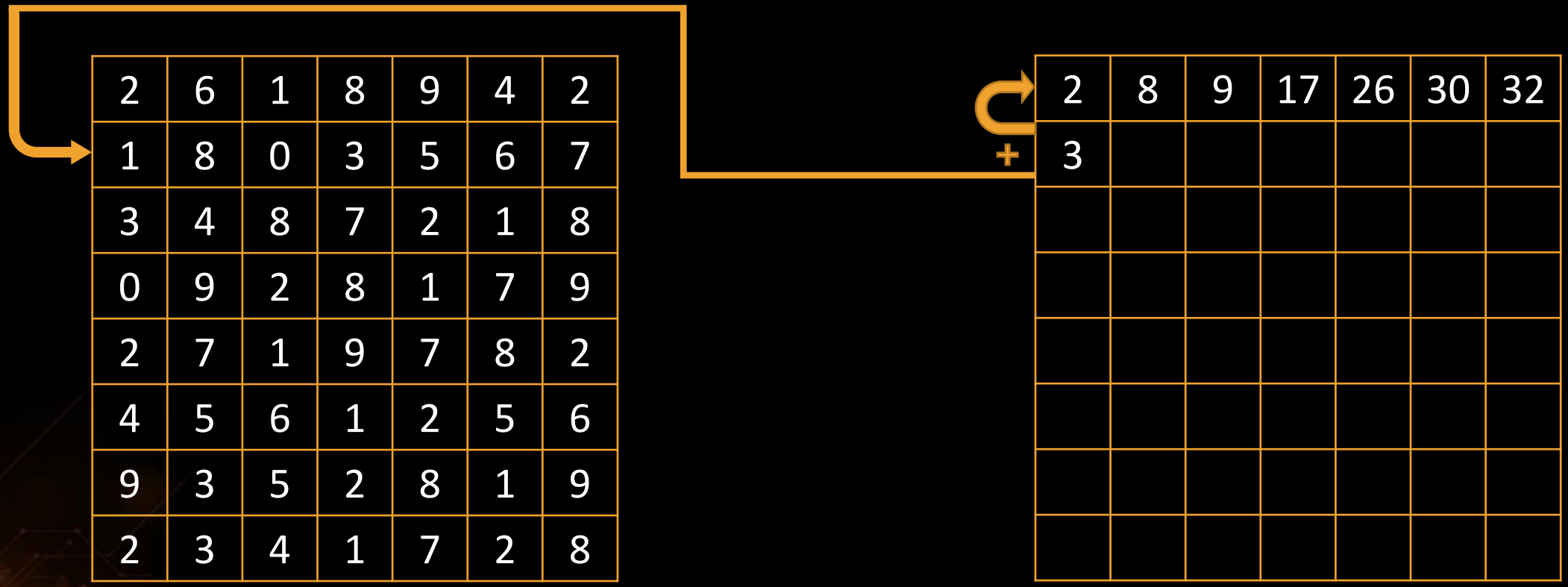
Building the DP Matrix



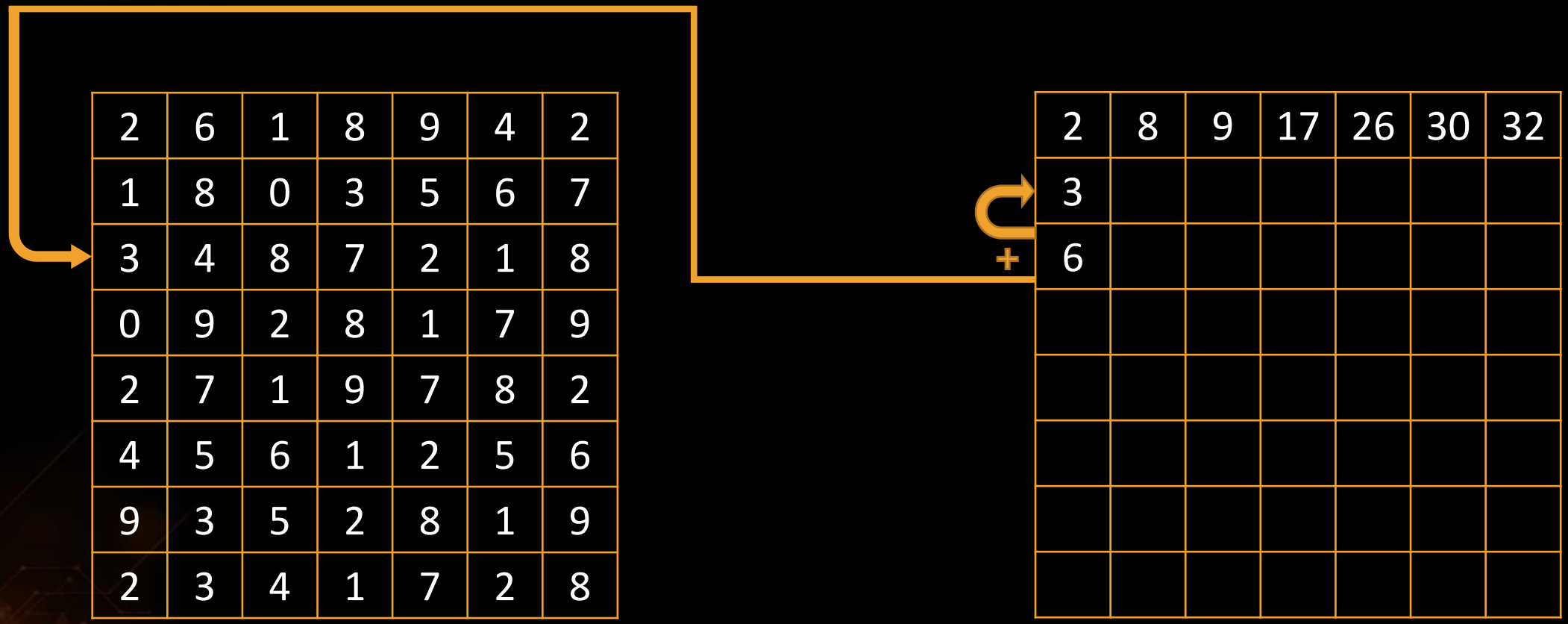
2	6	1	8	9	4	2
1	8	0	3	5	6	7
3	4	8	7	2	1	8
0	9	2	8	1	7	9
2	7	1	9	7	8	2
4	5	6	1	2	5	6
9	3	5	2	8	1	9
2	3	4	1	7	2	8

2	8	9	17	26	30	32

Building the DP Matrix



Building the DP Matrix



Building the DP Matrix

2	6	1	8	9	4	2
1	8	0	3	5	6	7
3	4	8	7	2	1	8
0	9	2	8	1	7	9
2	7	1	9	7	8	2
4	5	6	1	2	5	6
9	3	5	2	8	1	9
2	3	4	1	7	2	8

2	8	9	17	26	30	32
3						
6						
6						

Building the DP Matrix

2	6	1	8	9	4	2
1	8	0	3	5	6	7
3	4	8	7	2	1	8
0	9	2	8	1	7	9
2	7	1	9	7	8	2
4	5	6	1	2	5	6
9	3	5	2	8	1	9
2	3	4	1	7	2	8

2	8	9	17	26	30	32
3						
6						
6						
8						

Building the DP Matrix

2	6	1	8	9	4	2
1	8	0	3	5	6	7
3	4	8	7	2	1	8
0	9	2	8	1	7	9
2	7	1	9	7	8	2
4	5	6	1	2	5	6
9	3	5	2	8	1	9
2	3	4	1	7	2	8

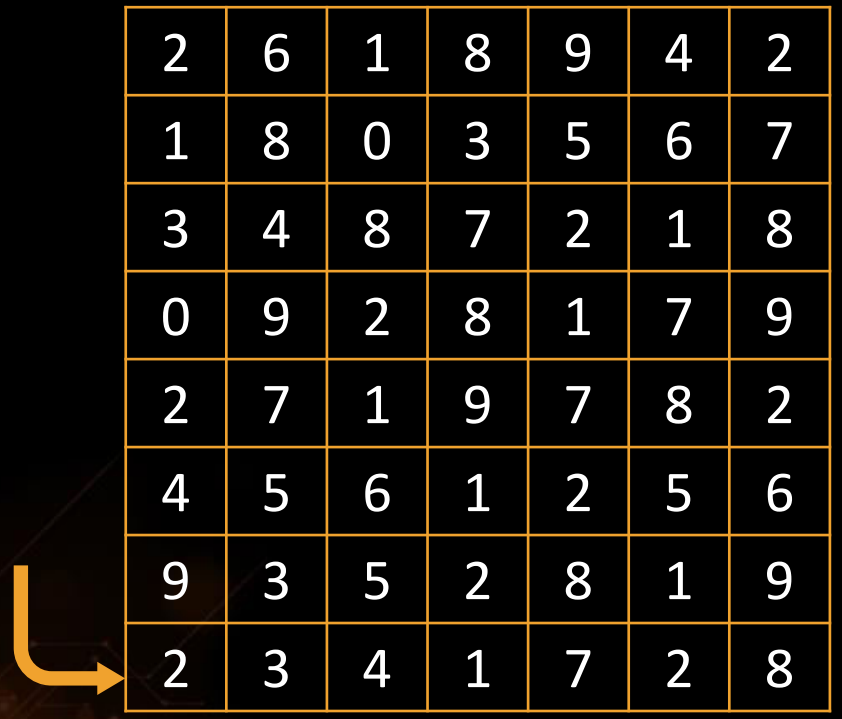
2	8	9	17	26	30	32
3						
6						
6						
8						
12						

Building the DP Matrix

2	6	1	8	9	4	2
1	8	0	3	5	6	7
3	4	8	7	2	1	8
0	9	2	8	1	7	9
2	7	1	9	7	8	2
4	5	6	1	2	5	6
9	3	5	2	8	1	9
2	3	4	1	7	2	8

2	8	9	17	26	30	32
3						
6						
6						
8						
12						
21						

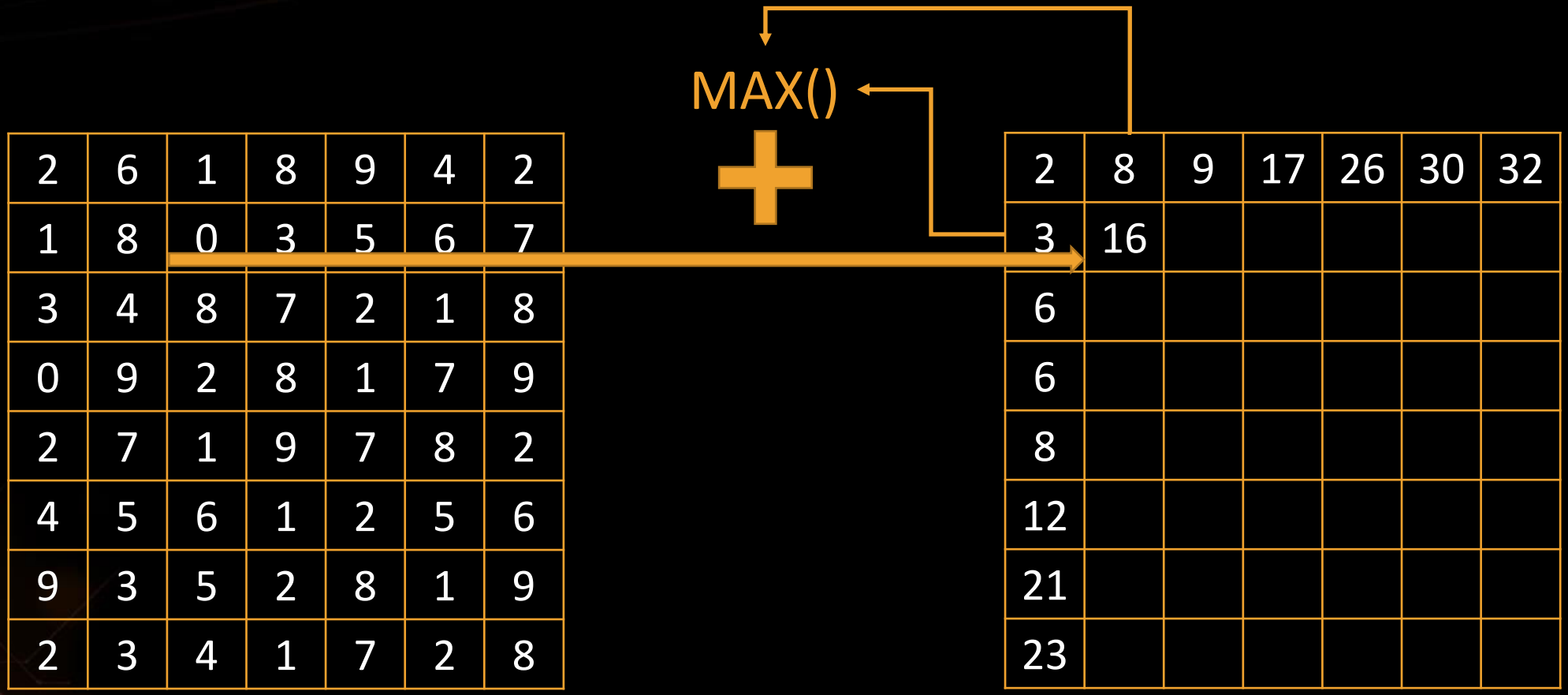
Building the DP Matrix



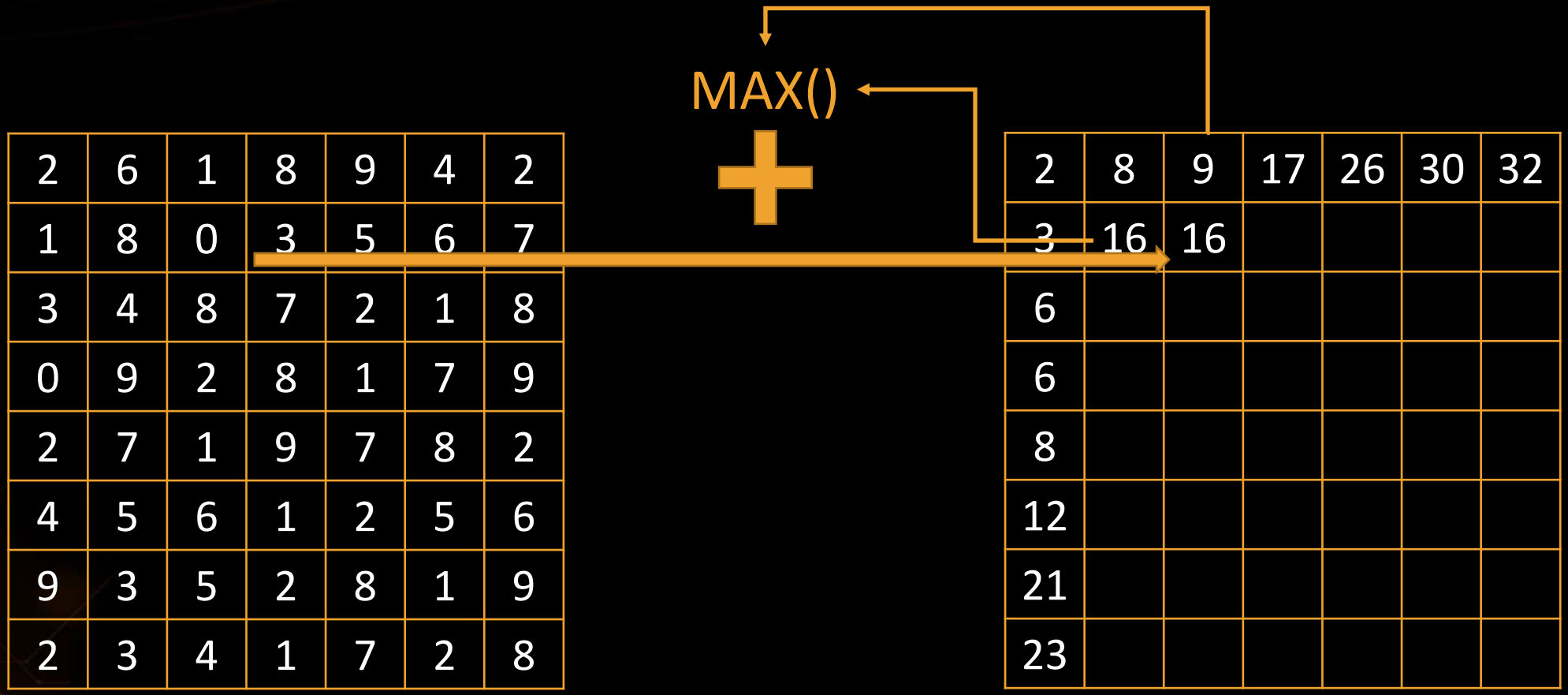
2	6	1	8	9	4	2
1	8	0	3	5	6	7
3	4	8	7	2	1	8
0	9	2	8	1	7	9
2	7	1	9	7	8	2
4	5	6	1	2	5	6
9	3	5	2	8	1	9
2	3	4	1	7	2	8

2	8	9	17	26	30	32
3						
6						
6						
8						
12						
21						
23						

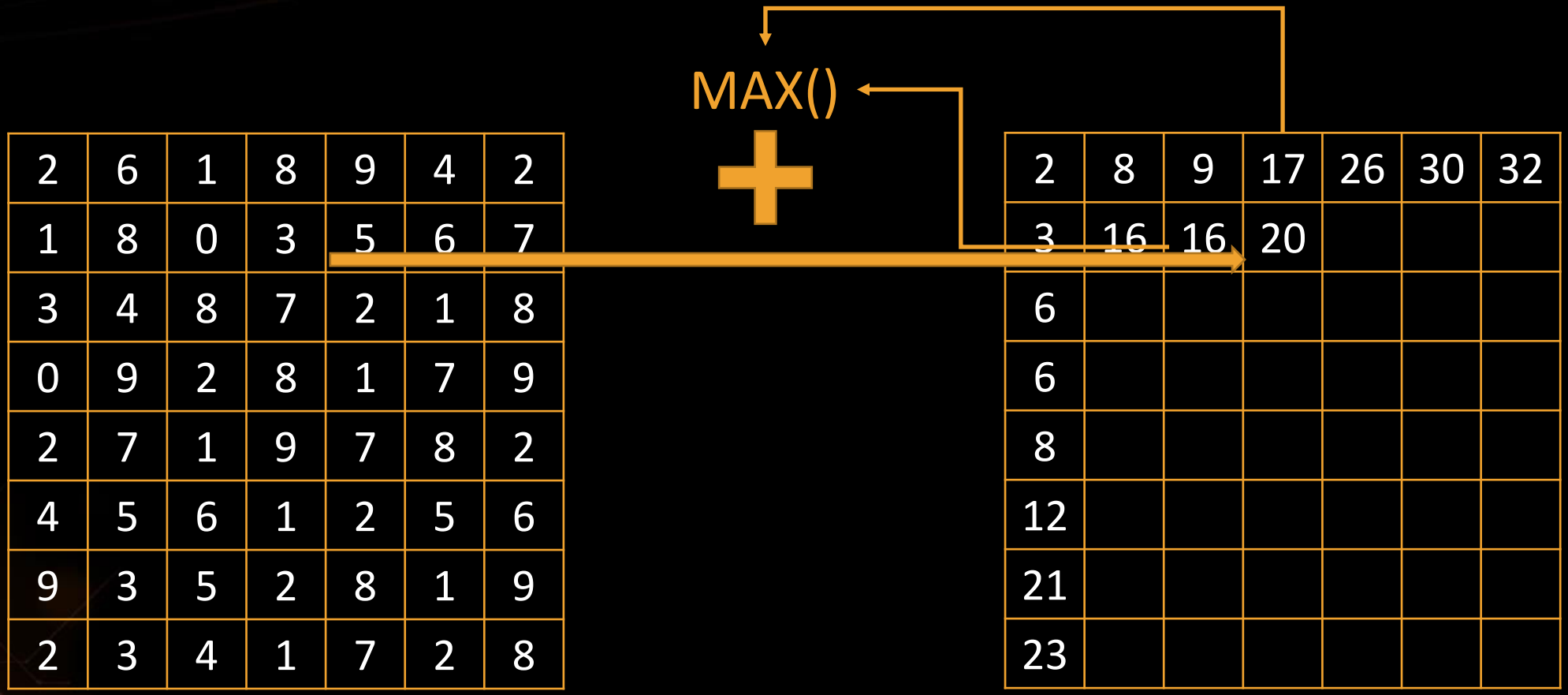
Building the DP Matrix



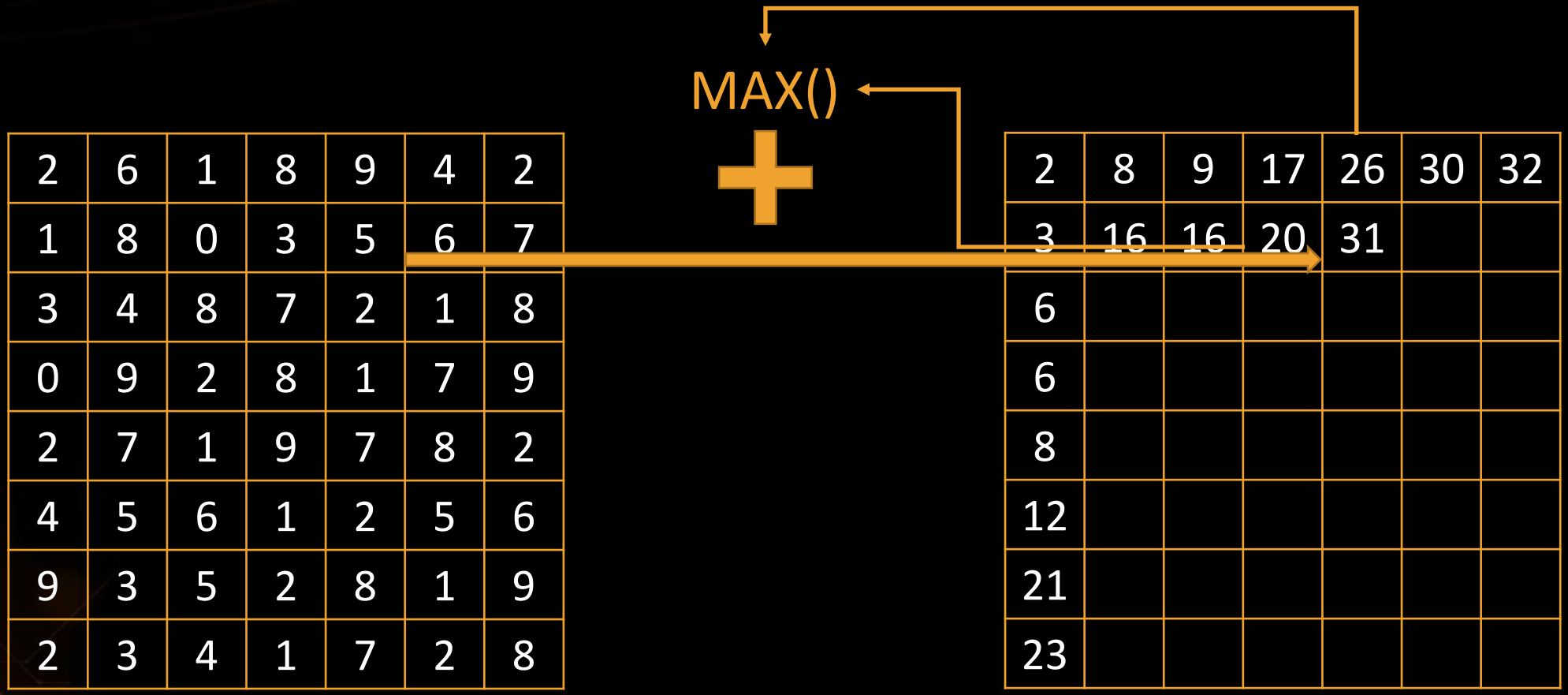
Building the DP Matrix



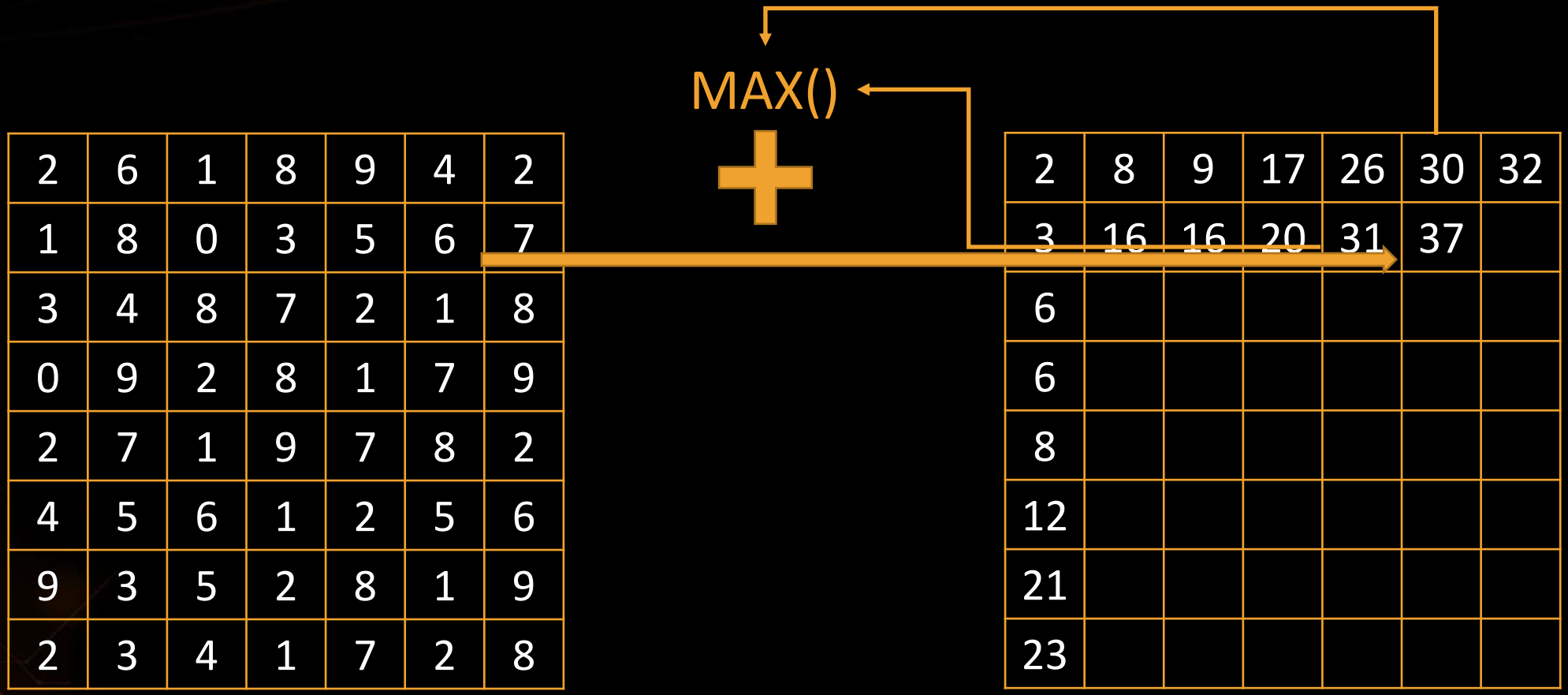
Building the DP Matrix



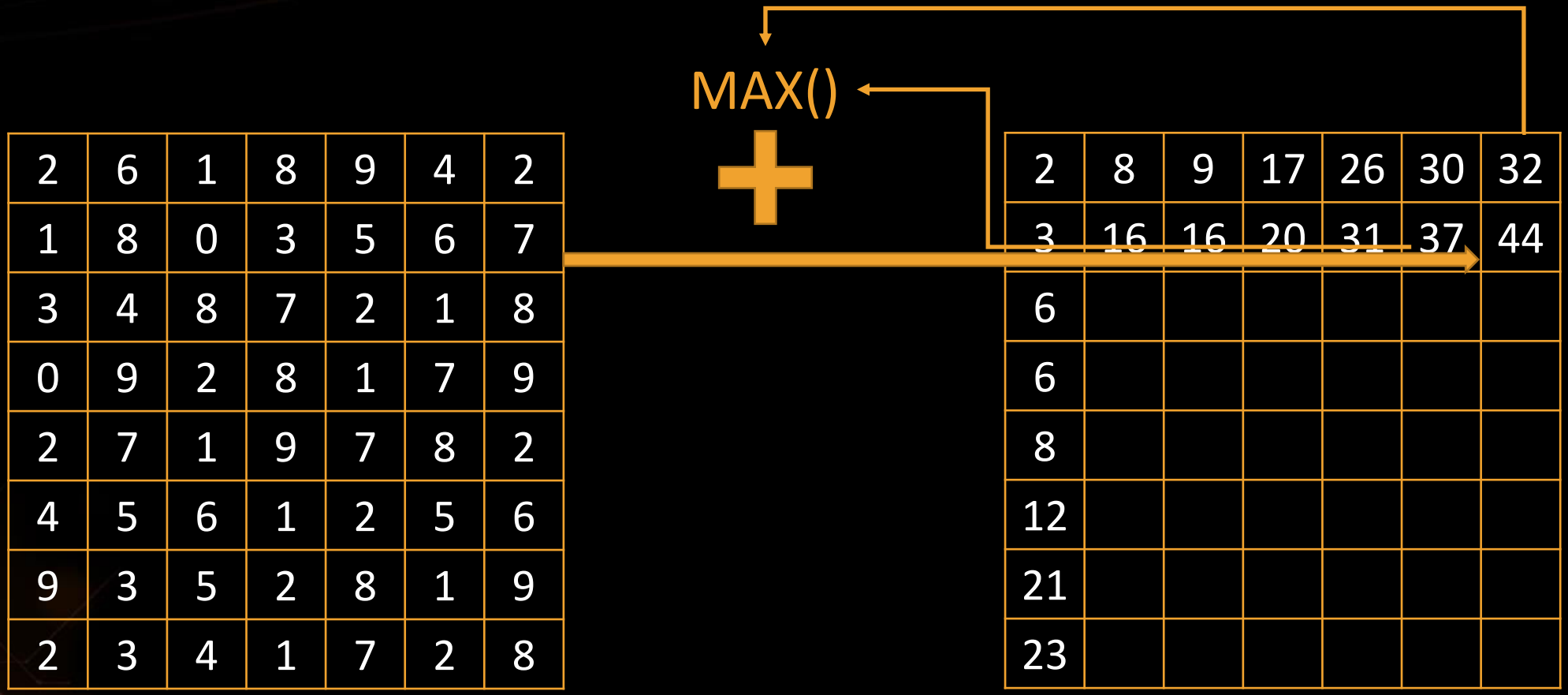
Building the DP Matrix



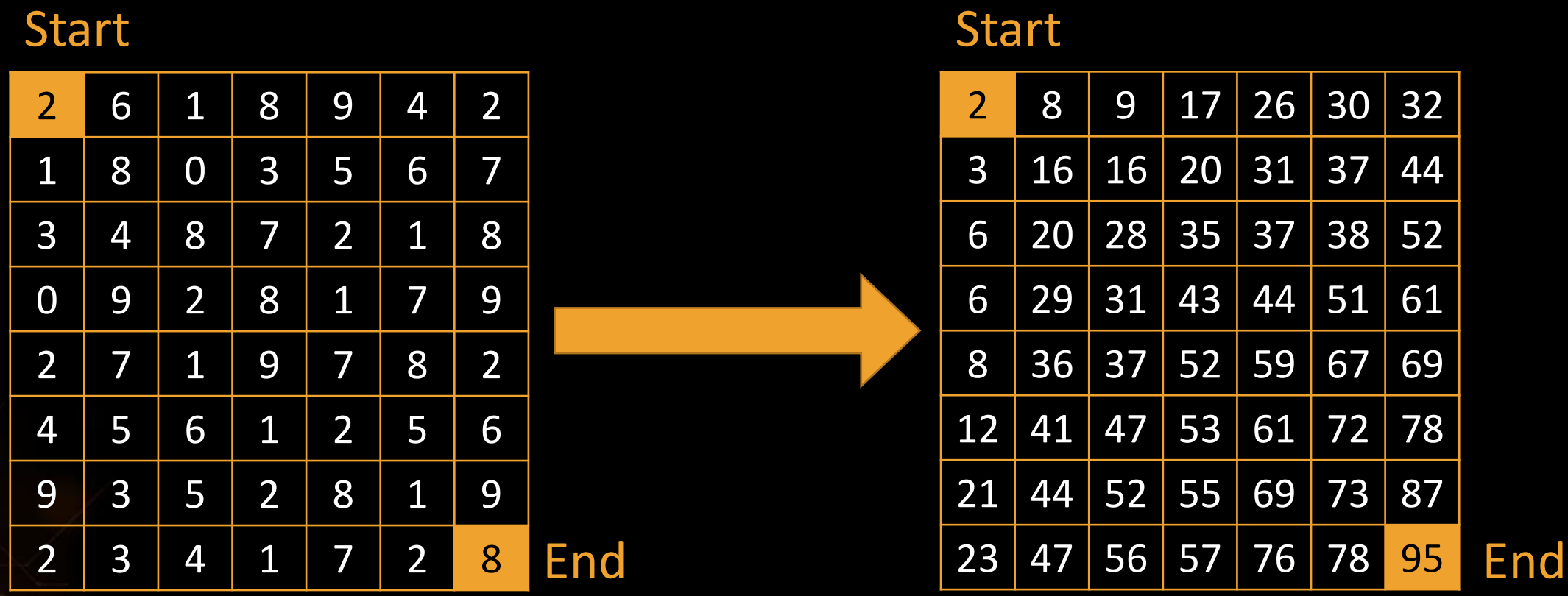
Building the DP Matrix



Building the DP Matrix



Building the DP Matrix



Finding the Path

SoftUni Foundation

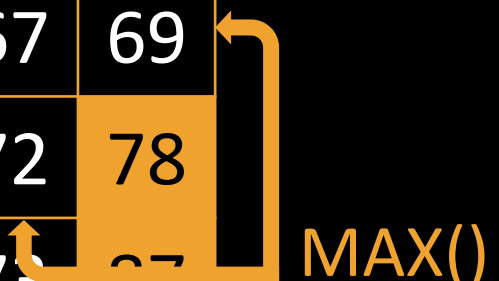
2	8	9	17	26	30	32
3	16	16	20	31	37	44
6	20	28	35	37	38	52
6	29	31	43	44	51	61
8	36	37	52	59	67	69
12	41	47	53	61	72	78
21	44	52	55	69	73	87
23	47	56	57	76	78	95

Finding the Path

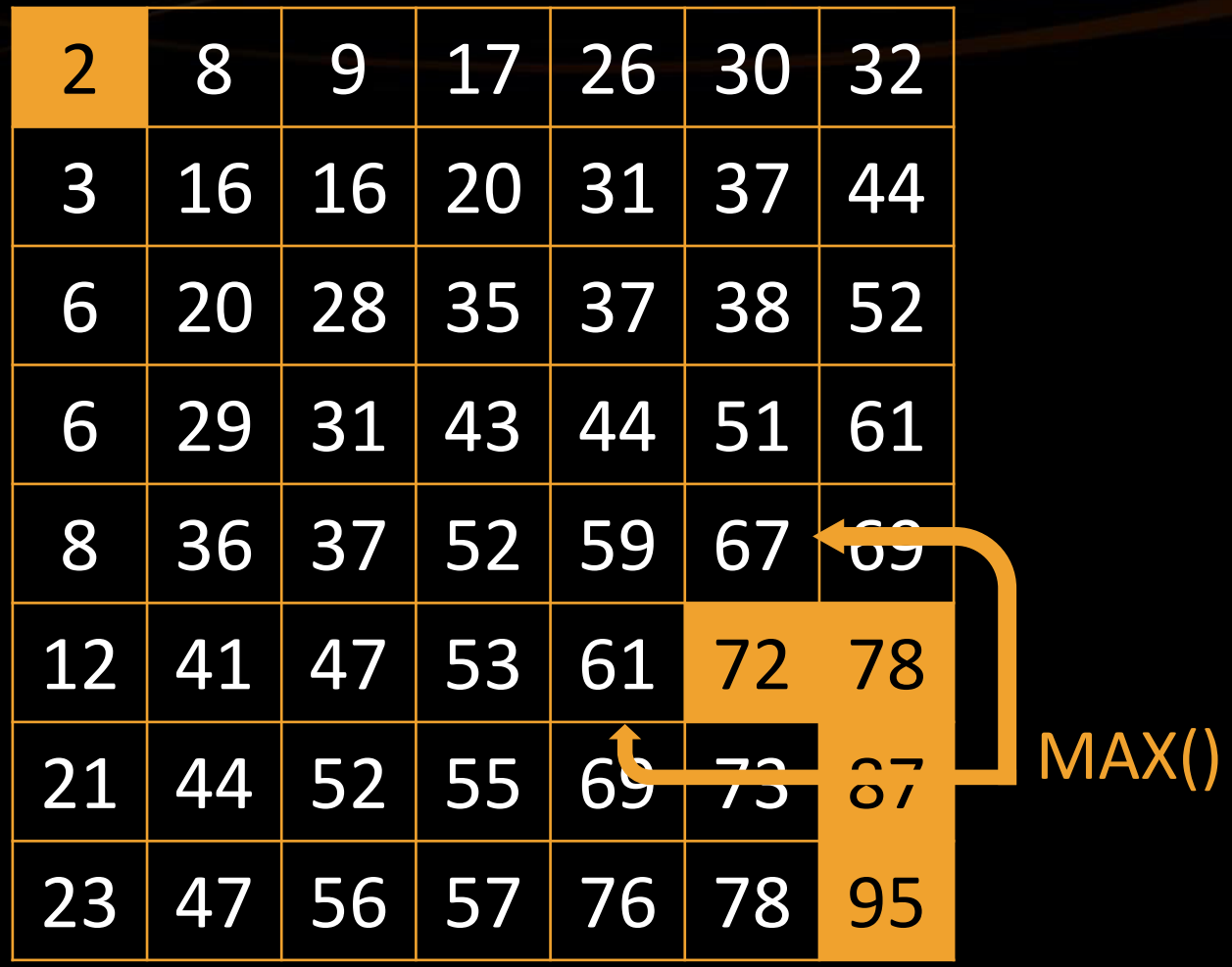
Finding the Path

SoftUni Foundation

2	8	9	17	26	30	32
3	16	16	20	31	37	44
6	20	28	35	37	38	52
6	29	31	43	44	51	61
8	36	37	52	59	67	69
12	41	47	53	61	72	78
21	44	52	55	69	75	87
23	47	56	57	76	78	95



Finding the Path



Finding the Path

SoftUni Foundation

2	8	9	17	26	30	32
3	16	16	20	31	37	44
6	20	28	35	37	38	52
6	29	31	43	44	51	61
8	36	37	52	59	67	69
12	41	47	53	61	72	78
21	44	52	55	69	73	87
23	47	56	57	76	78	95

Finding the Path

SoftUni Foundation

2	8	9	17	26	30	32
3	16	16	20	31	37	44
6	20	28	35	37	38	52
6	29	31	43	44	51	61
8	36	37	52	59	67	69
12	41	47	53	61	72	78
21	44	52	55	69	73	87
23	47	56	57	76	78	95

Finding the Path

SoftUni Foundation

2	8	9	17	26	30	32
3	16	16	20	31	37	44
6	20	28	35	37	38	52
6	29	31	43	44	51	61
8	36	37	52	59	67	69
12	41	47	53	61	72	78
21	44	52	55	69	73	87
23	47	56	57	76	78	95

Finding the Path

SoftUni Foundation

2	8	9	17	26	30	32
3	16	16	20	31	37	44
6	20	28	35	37	38	52
6	29	31	43	44	51	61
8	36	37	52	59	67	69
12	41	47	53	61	72	78
21	44	52	55	69	73	87
23	47	56	57	76	78	95

Finding the Path

SoftUni Foundation

2	8	9	17	26	30	32
3	16	16	20	31	37	44
6	20	28	35	37	38	52
6	29	31	43	44	51	61
8	36	37	52	59	67	69
12	41	47	53	61	72	78
21	44	52	55	69	73	87
23	47	56	57	76	78	95

Finding the Path

SoftUni Foundation

2	8	9	17	26	30	32
3	16	16	20	31	37	44
6	20	28	35	37	38	52
6	29	31	43	44	51	61
8	36	37	52	59	67	69
12	41	47	53	61	72	78
21	44	52	55	69	73	87
23	47	56	57	76	78	95

Finding the Path

SoftUni Foundation

2	8	9	17	26	30	32
3	16	16	20	31	37	44
6	20	28	35	37	38	52
6	29	31	43	44	51	61
8	36	37	52	59	67	69
12	41	47	53	61	72	78
21	44	52	55	69	73	87
23	47	56	57	76	78	95

Finding the Path

SoftUni Foundation

2	8	9	17	26	30	32
3	16	16	20	31	37	44
6	20	28	35	37	38	52
6	29	31	43	44	51	61
8	36	37	52	59	67	69
12	41	47	53	61	72	78
21	44	52	55	69	73	87
23	47	56	57	76	78	95

Finding the Path

SoftUni Foundation

2	8	9	17	26	30	32
3	16	16	20	31	37	44
6	20	28	35	37	38	52
6	29	31	43	44	51	61
8	36	37	52	59	67	69
12	41	47	53	61	72	78
21	44	52	55	69	73	87
23	47	56	57	76	78	95

Finding the Path

SoftUni Foundation

2	8	9	17	26	30	32
3	16	16	20	31	37	44
6	20	28	35	37	38	52
6	29	31	43	44	51	61
8	36	37	52	59	67	69
12	41	47	53	61	72	78
21	44	52	55	69	73	87
23	47	56	57	76	78	95

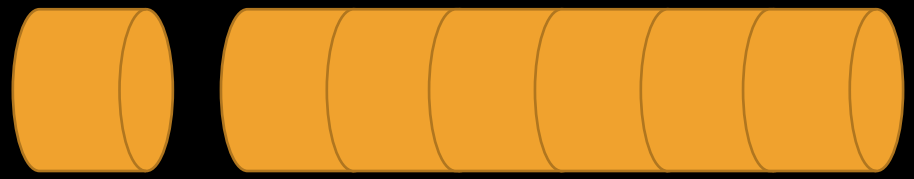
2	6	1	8	9	4	2
1	8	0	3	5	6	7
3	4	8	7	2	1	8
0	9	2	8	1	7	9
2	7	1	9	7	8	2
4	5	6	1	2	5	6
9	3	5	2	8	1	9
2	3	4	1	7	2	8

"Move Down / Right Sum" – Solution

SoftUni Foundation

```
for (int row = 0; row < rowsCount; row++)
{
  for (int col = 0; col < colsCount; col++)
  {
    long maxPrevCell = long.MinValue;
    if (col > 0 && sum[row, col - 1] > maxPrevCell)
      maxPrevCell = sum[row, col - 1];
    if (row > 0 && sum[row - 1, col] > maxPrevCell)
      maxPrevCell = sum[row - 1, col];
    sum[row, col] = cells[row, col];
    if (maxPrevCell != long.MinValue)
      sum[row, col] += maxPrevCell;
  }
}
```

SoftUni Foundation



Rod Cutting Problem

Max Profit from Cutting a Rod

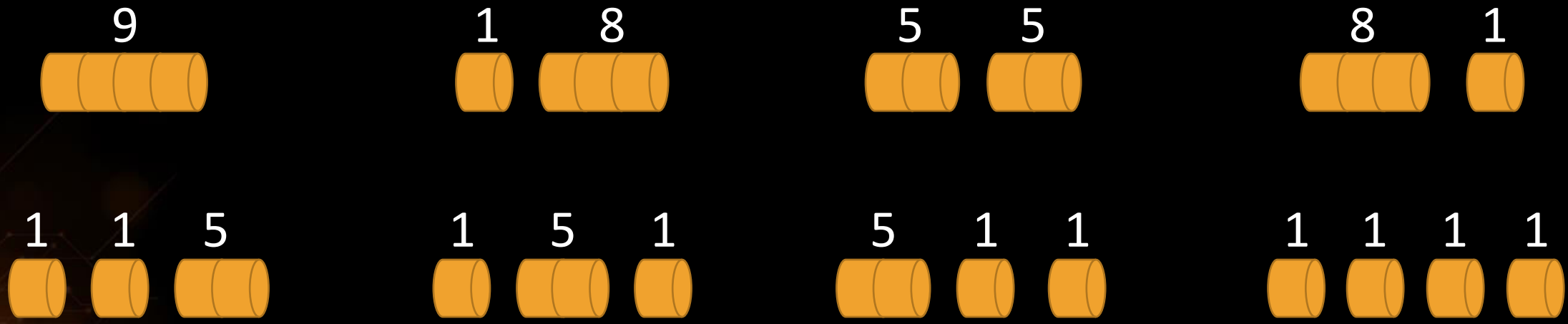
Rod Cutting

SoftUni Foundation

- Find the **best way** to cut up a rod, given the prices below

Length	0	1	2	3	4	5	6	7	8	9	10
Price	0	1	5	8	9	10	17	17	20	24	30

- Example: All possible ways to **cut rod** with **length = 4**



SoftUni Foundation

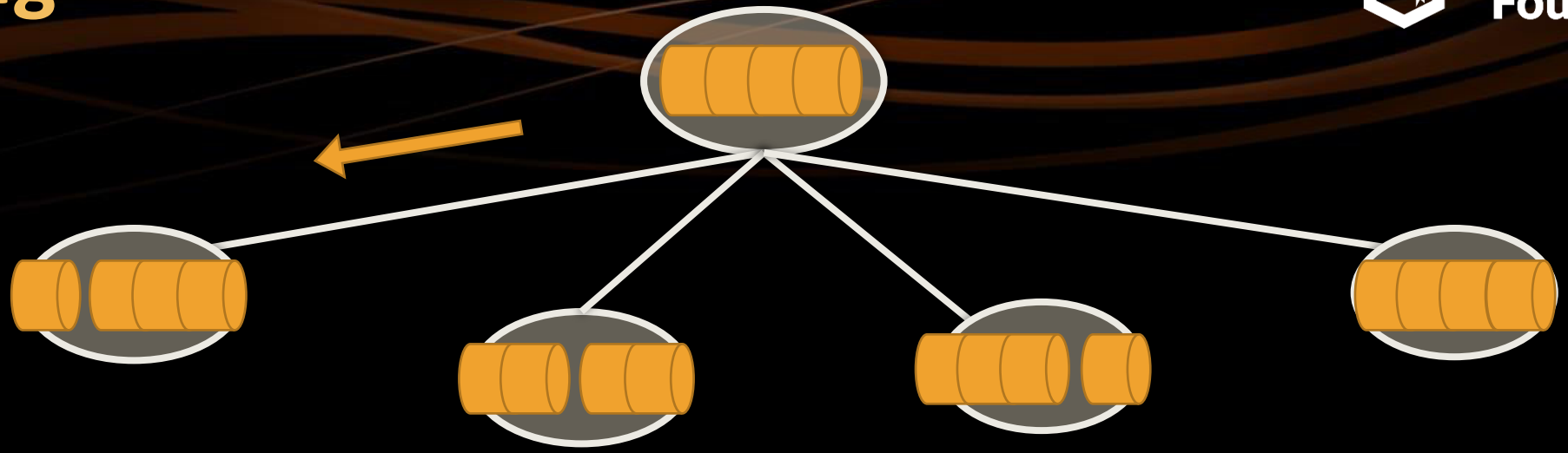
Rod Cutting

Recursive Top-Down Approach

Rod Cutting

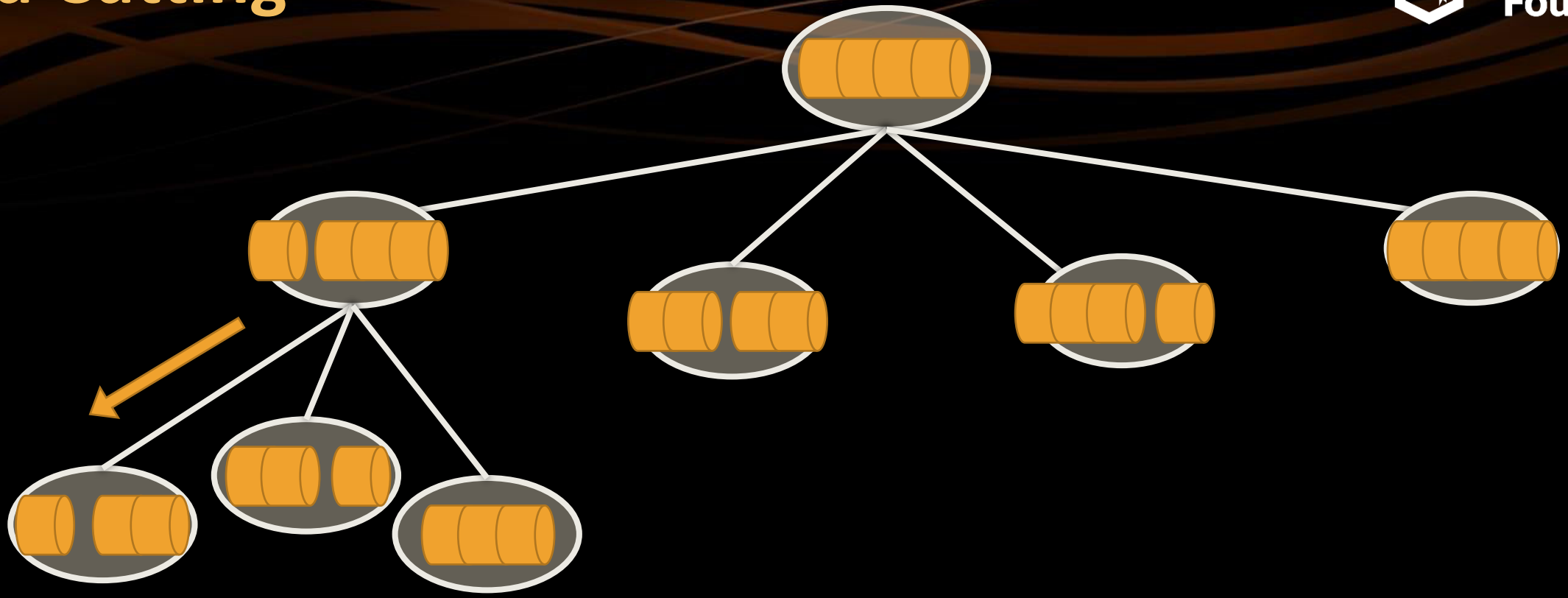
Length	0	1	2	3	4
Price	0	1	5	8	9

Rod Cutting



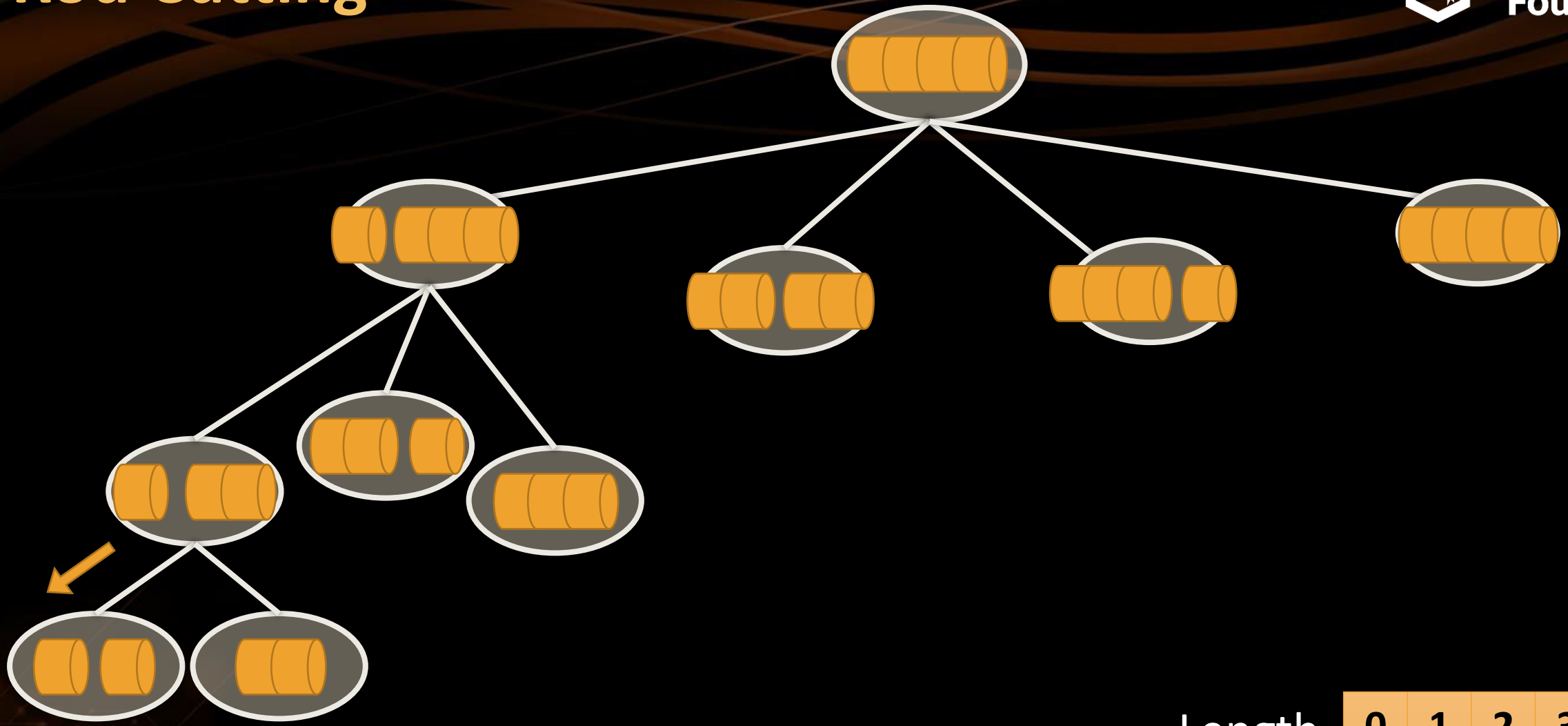
Length	0	1	2	3	4
Price	0	1	5	8	9

Rod Cutting



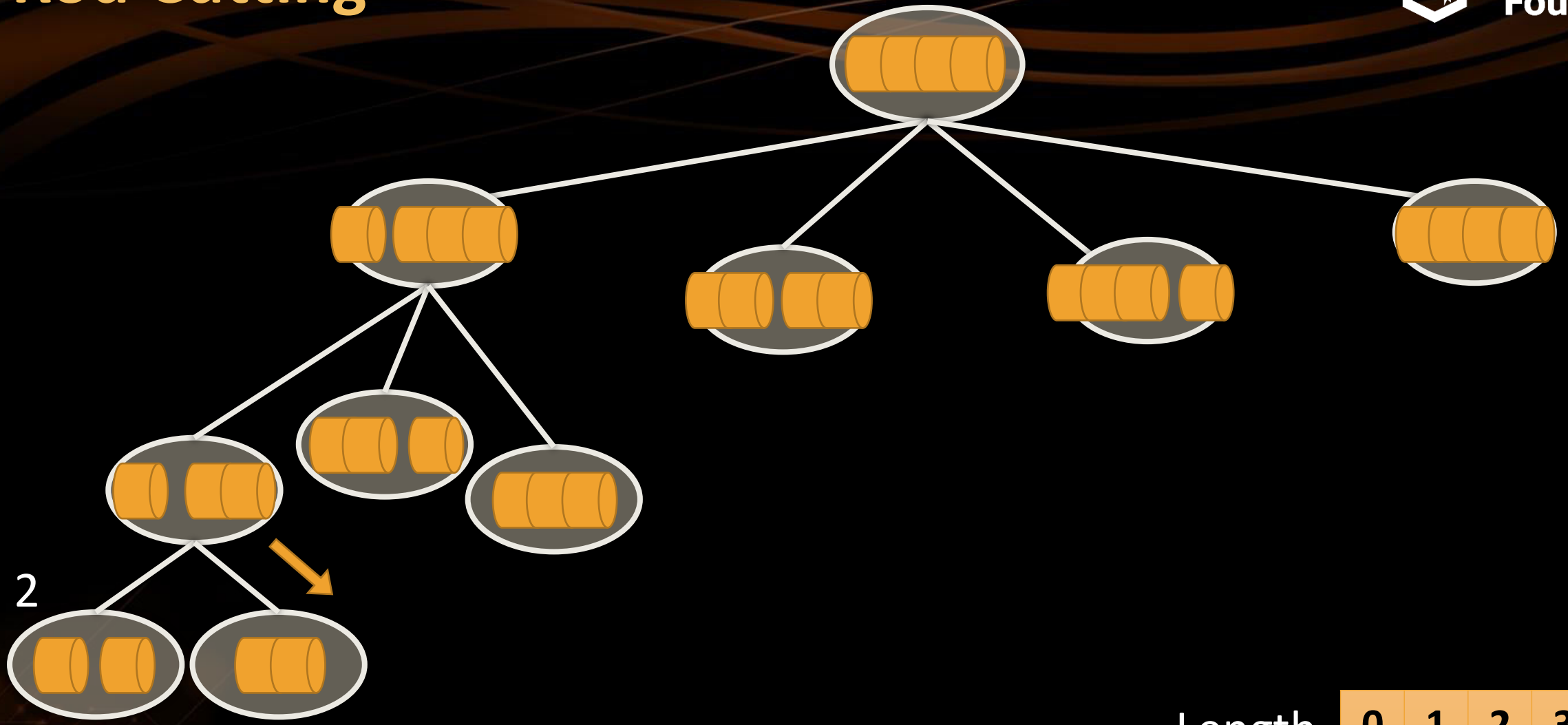
Length	0	1	2	3	4
Price	0	1	5	8	9

Rod Cutting



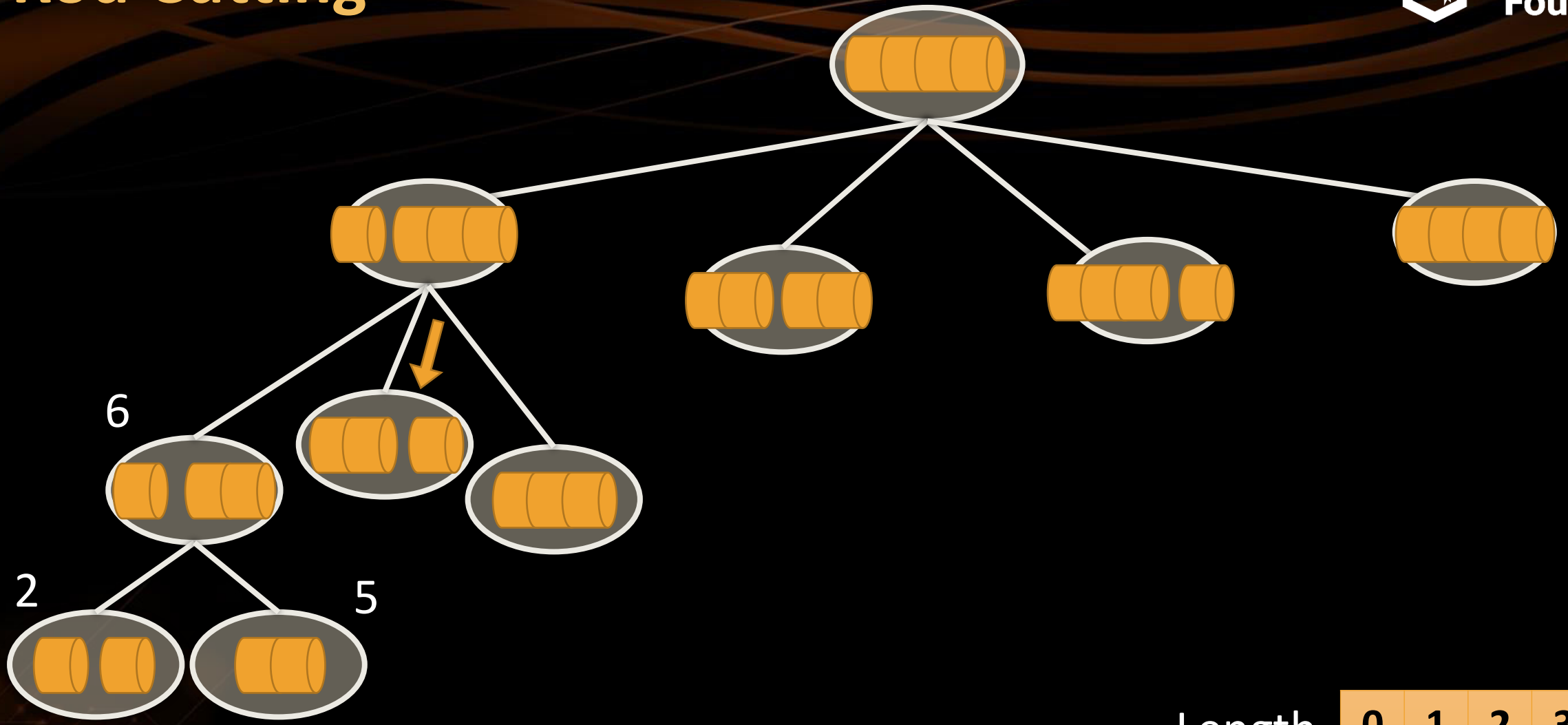
Length	0	1	2	3	4
Price	0	1	5	8	9

Rod Cutting



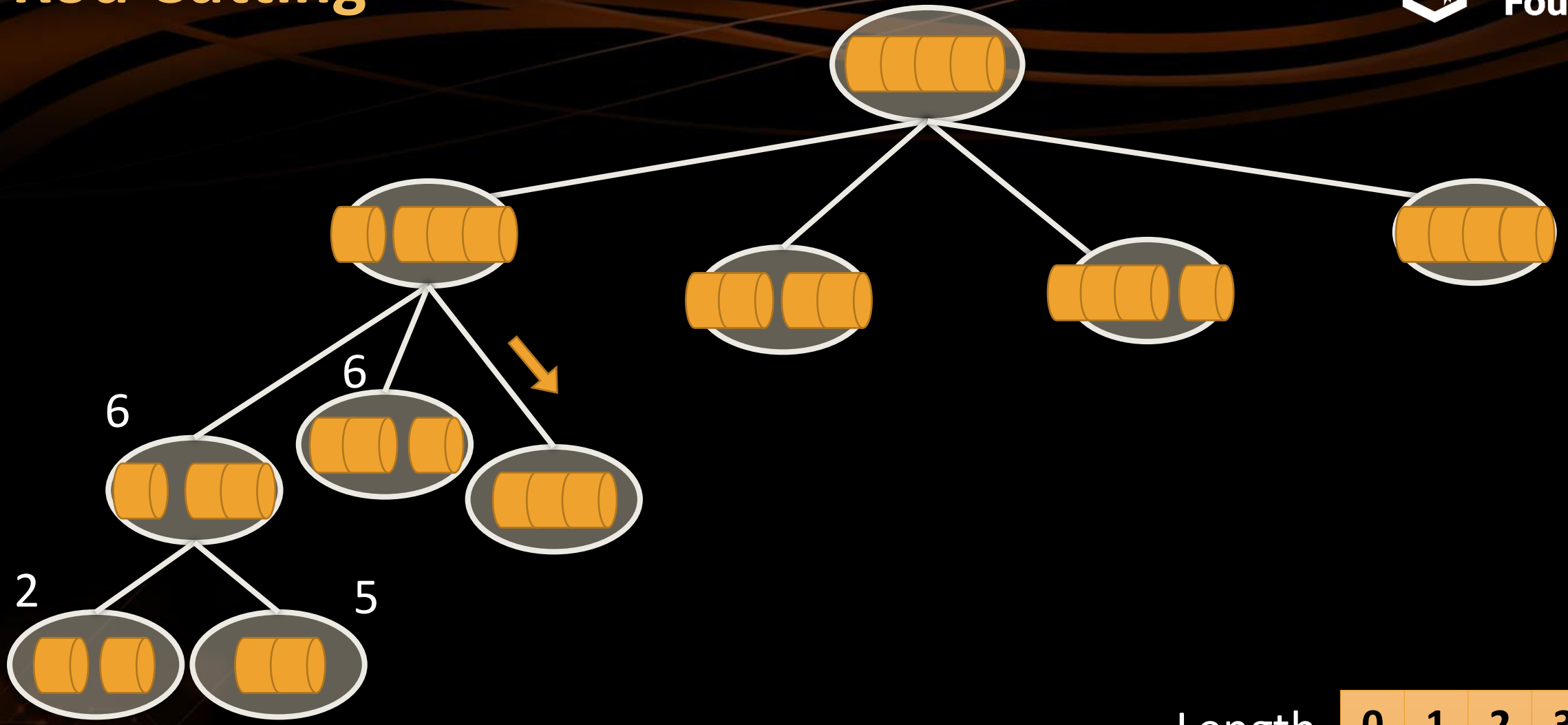
Length	0	1	2	3	4
Price	0	1	5	8	9

Rod Cutting



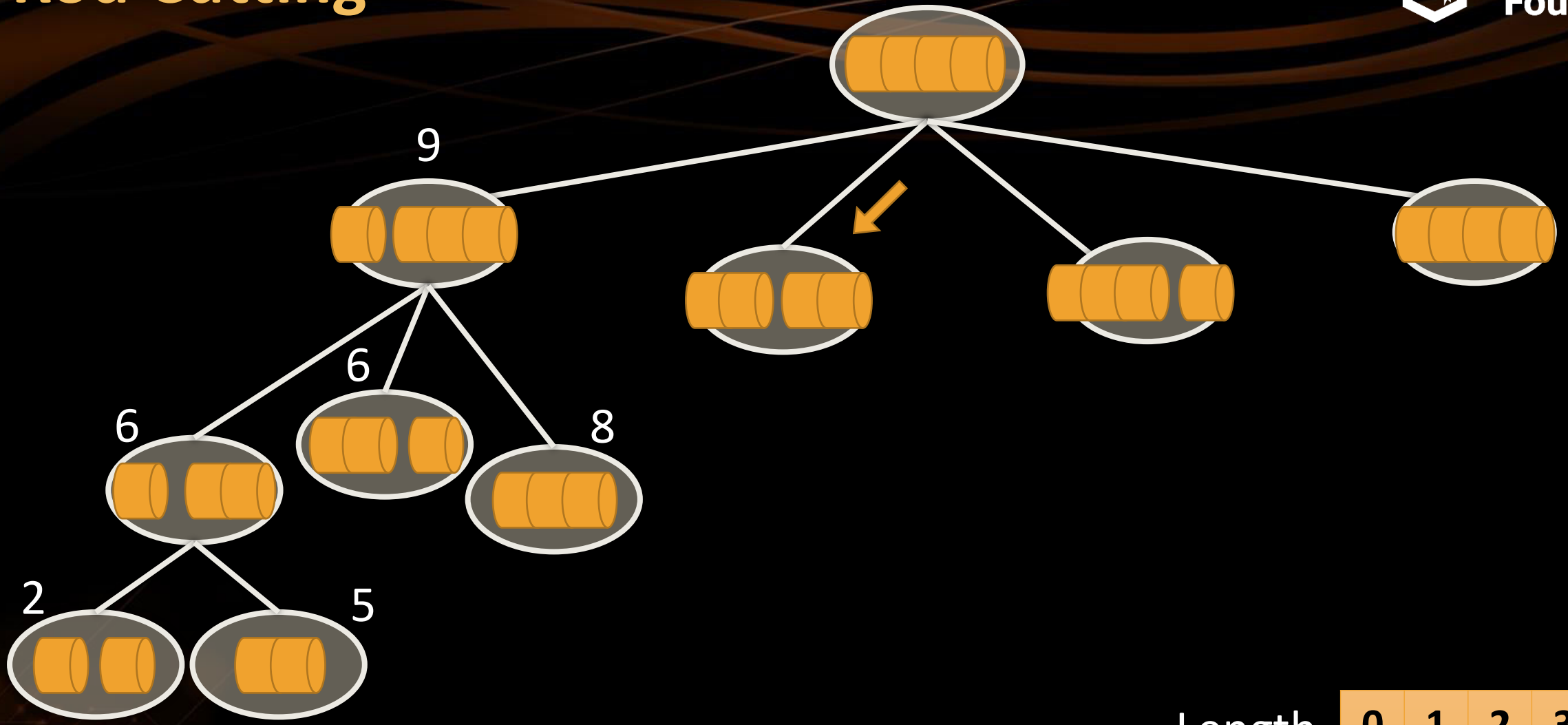
Length	0	1	2	3	4
Price	0	1	5	8	9

Rod Cutting



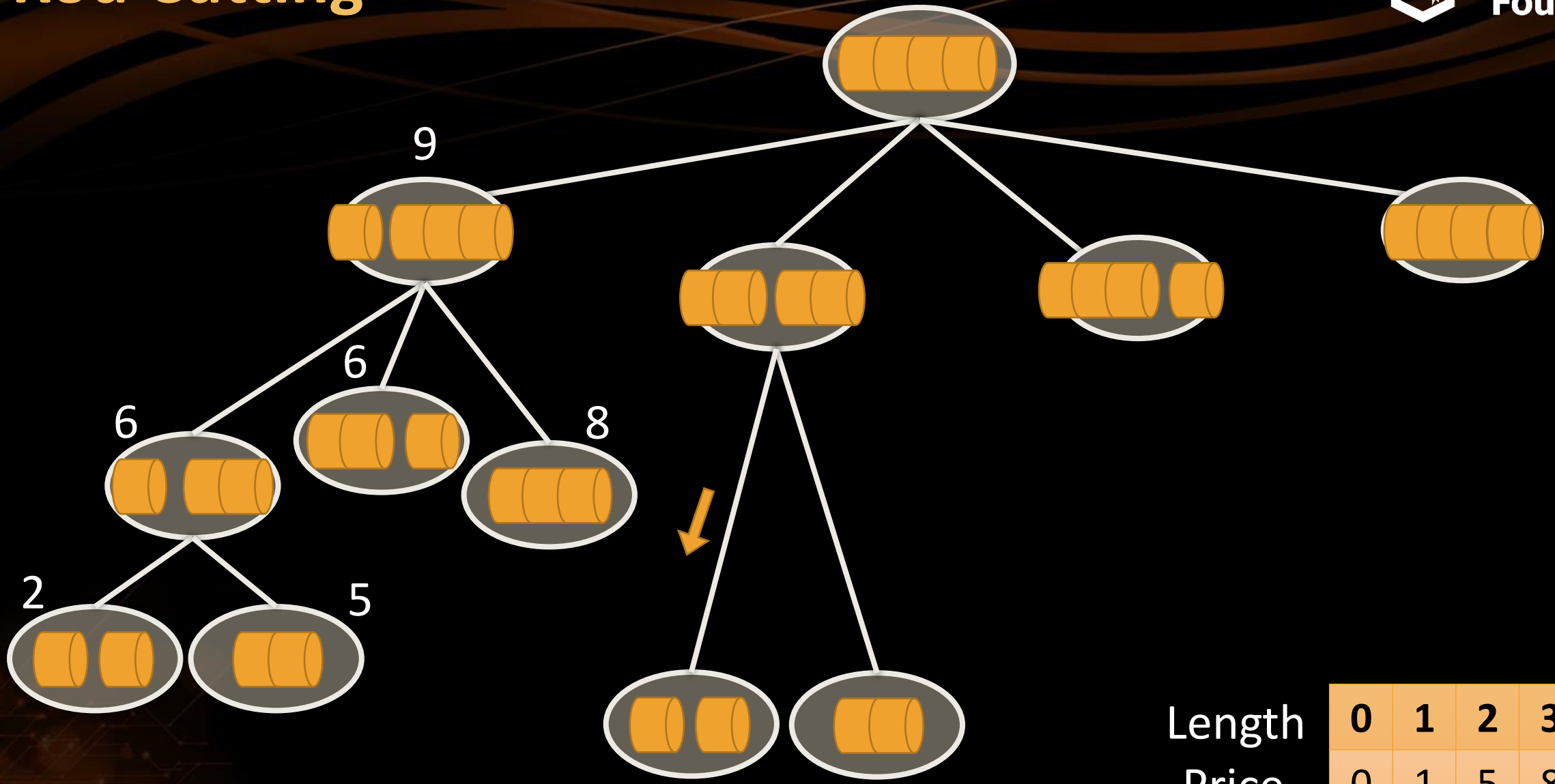
Length	0	1	2	3	4
Price	0	1	5	8	9

Rod Cutting



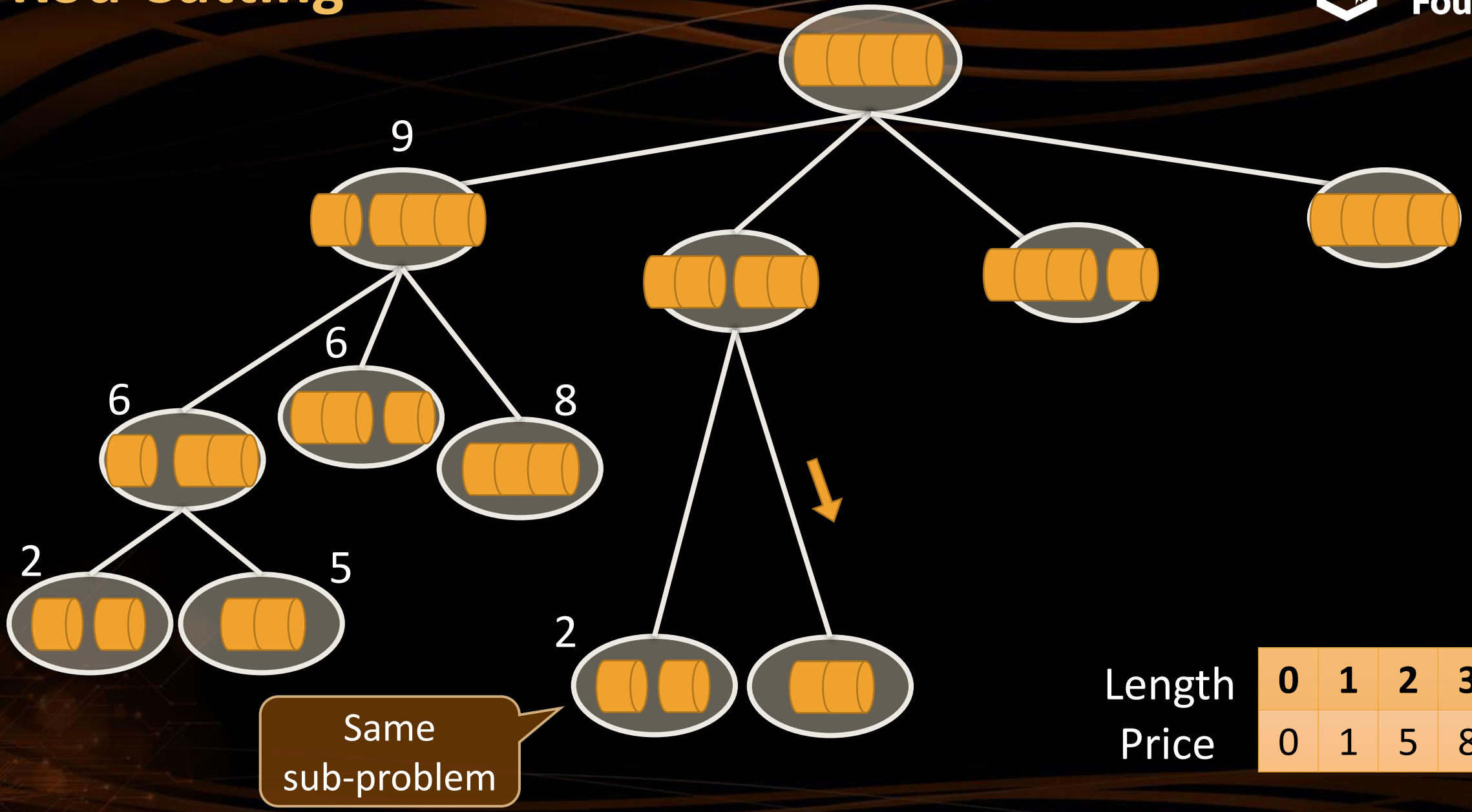
Length	0	1	2	3	4
Price	0	1	5	8	9

Rod Cutting



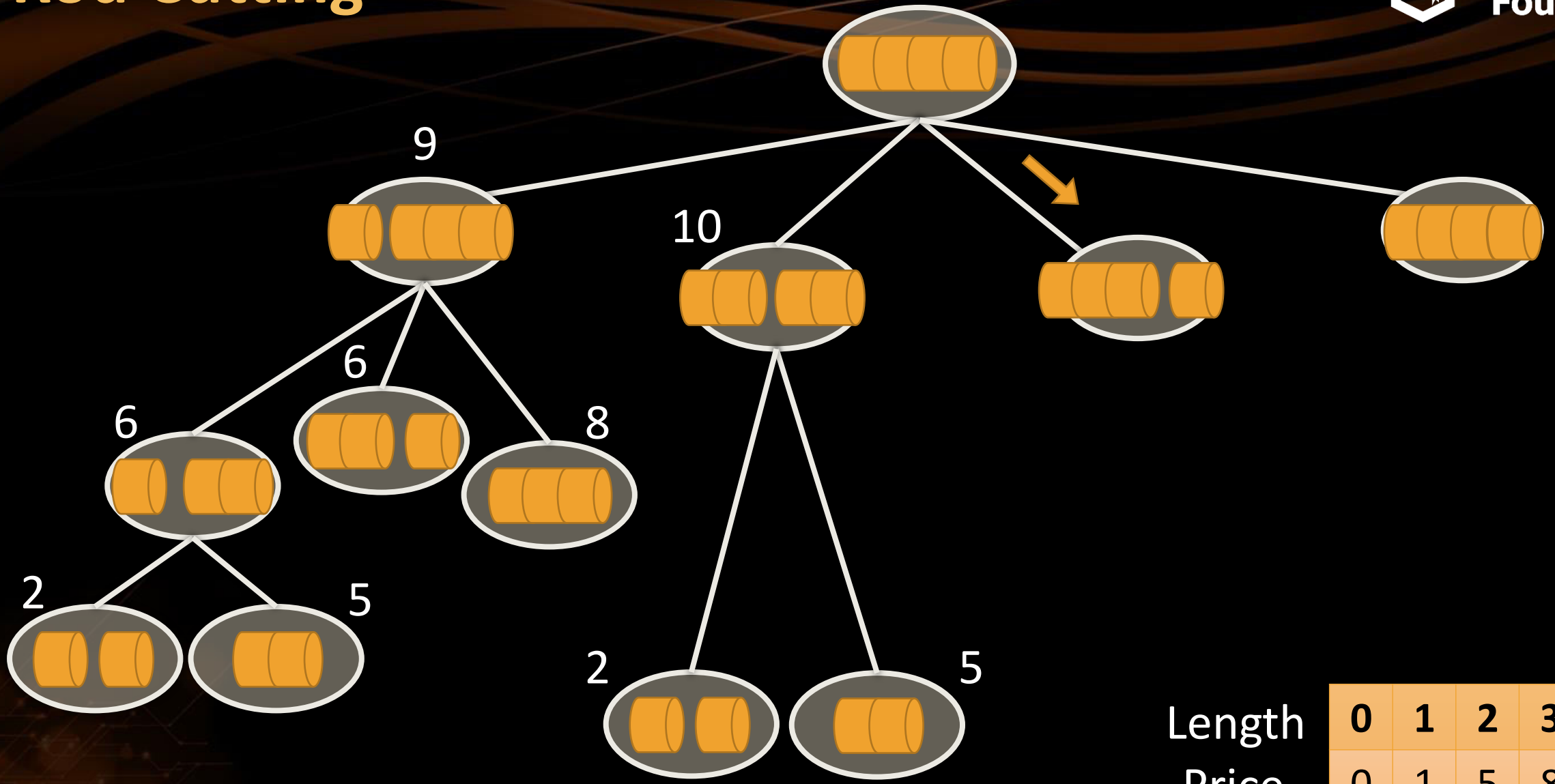
Length	0	1	2	3	4
Price	0	1	5	8	9

Rod Cutting



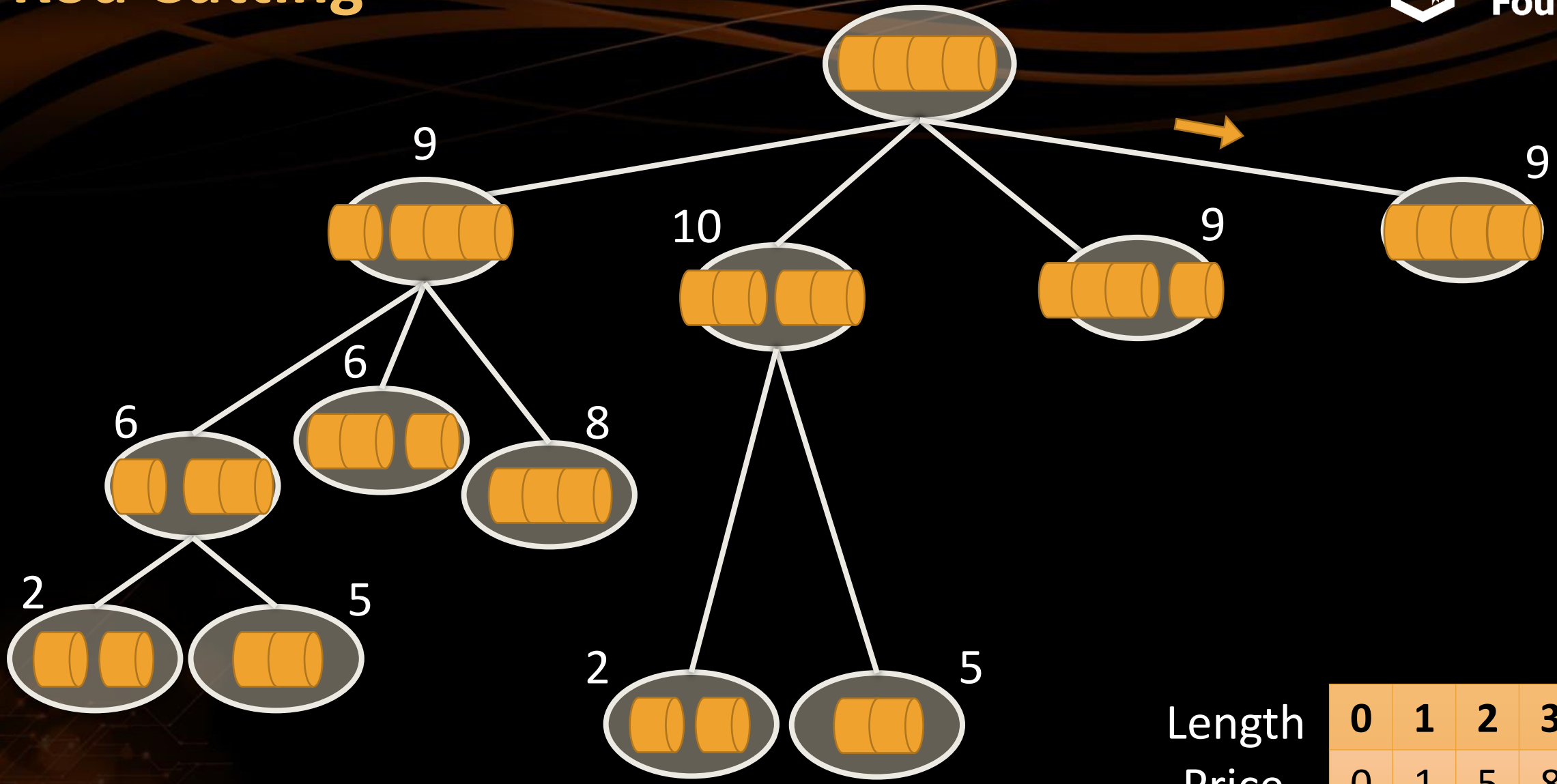
Length	0	1	2	3	4
Price	0	1	5	8	9

Rod Cutting



Length	0	1	2	3	4
Price	0	1	5	8	9

Rod Cutting



Length	0	1	2	3	4
Price	0	1	5	8	9

Rod Cutting – Reconstructing Solution

SoftUni Foundation

Length	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Price	0	1	5	8	9	10	17	17	20	24	30

Best Price	0	1	5	8	10	10	17	18	20	24	30
Best Prev	0	1	2	3	2	2	6	6	8	9	10

Rod Cutting – Recursive Solution

SoftUni Foundation

```
private static int CutRod(int n) {
  if (bestPrice[n] >= 0) return bestPrice[n];
  if (n == 0) return 0;
  var currentBest = bestPrice[n];
  for (int i = 1; i <= n; i++) {
    currentBest =
        Math.Max(currentBest, price[i] + CutRodTD(n - i));
    if (currentBest > bestPrice[n]) {
      bestPrice[n] = currentBest; bestCombo[n] = i;
    }
  }
  return bestPrice[n];
}
```

Rod Cutting – Iterative Solution

SoftUni Foundation

```
private static int CutRod(int n) {
  for (int i = 1; i <= n; i++) {
  int currentBest = bestPrice[i];
    for (int j = 1; j <= i; j++) {
      currentBest =
            Math.Max(bestPrice[i], price[j] + bestPrice[i - j]);
      if (currentBest > bestPrice[i]) {
        bestPrice[i] = currentBest;
        bestCombo[i] = j;
      }
    }
  }
  return bestPrice[n];
}
```

Rod Cutting – Reconstructing Solution

SoftUni Foundation

```
private static void ReconstructSolution(int n)
{
  while (n - bestCombo[n] != 0)
  {
    Console.Write(bestCombo[n] + " ");
    n = n - bestCombo[n];
  }

  Console.WriteLine(bestCombo[n]);
}
```

Summary

- **DP** → Solve a problem by **solving overlapping subproblems**
- **Memoization** → **Save** subproblem **solutions** for later use
- **Optimal Substructure**
  - **Subproblems** should have **optimal solutions**
  - Combine optimal solutions for subproblems
  - Get optimal solution for original problem
- **Top down** approach – **Recursive**
- **Bottom up** approach – **Iterative**

Dynamic Programming

SoftUni Foundation

Questions?

https://softuni.bg/opencourses/algorithms

License

SoftUni Foundation

- This course (slides, examples, labs, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

- Attribution: this work may contain portions from
  - "Fundamentals of Computer Programming with C#" book by Svetlin Nakov & Co. under CC-BY-SA license
  - "Data Structures and Algorithms" course by Telerik Academy under CC-BY-NC-SA license

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

Software University

SoftUni Foundation

FORUM