ROD CUTTING PROBLEM

DYNAMIC PROGRAMMING

Rod cutting problem

- Given a rod with certain length **l**
- Given the prices of different lengths
- How to cut the rod in order to **maximize** the profit?
- This is the rod cutting problem

Rod length → **l** = 5m

Prices for different lengths:

1m → $2
2m → $5
3m → $7
4m → $3

Solution to the rod cutting problem:

{2,3} so a cut the rod to get a 2m piece and a 3m piece

OR

{2,2,1} 2 2m pieces and a single 1m piece, it is going to be the same $12 profit

Total value for both solutions: **$12**

Recursion

- The naive approach is to use a simple recursive method / function
- **N-1** cuts can be made in the rod of length **N**
- There are $\mathbf{2^{N-1}}$ ways to cut the rod
- Problems: time complexity + overlapping subproblems
- Exponential time complexity: **O(2^N)** where **N** is the length of the rod in units
- (for every length we have **2** options whether to cut or not)

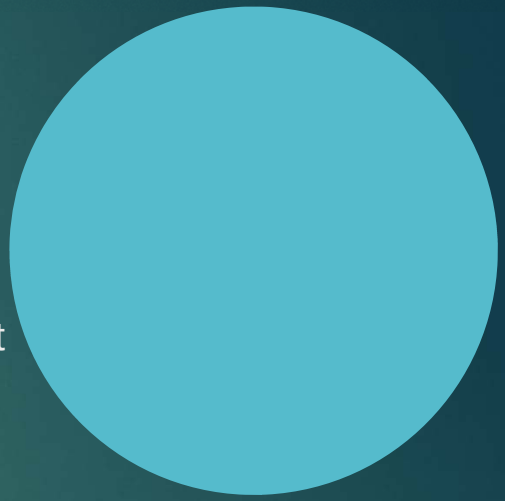
Dynamic programming

We have to create a solution matrix:

dpTable[numOfLengths+1][originalLength+1]
rows **columns**

We have to define the base cases:
- if originalLength is **0** → **0** is the profit
- if we do not consider any lengths → 0 is the profit

Complexity: **O(numOfLengths*originalLength)**



$$\text{dpTable}[i][j] = \begin{cases} 0 \text{ if } j = 0 \text{ and } 0 \text{ if } i = 0 \\ \max\{\, \text{dpTable}[i-1][j] \;;\; \text{prices}[i] + \text{dpTable}[i][j-i] \,\} \text{ if } i \leq j \\ \text{dpTable}[i-1][j] \text{ if } i > j \end{cases}$$

These are the base cases

$$\text{dpTable}[i][j] = \begin{cases} 0 \text{ if } j = 0 \text{ and } 0 \text{ if } i = 0 \\ \max\{\, \text{dpTable}[i-1][j] \,;\, \text{prices}[i] + \text{dpTable}[i][j-i] \,\} \text{ if } i \leq j \\ \text{dpTable}[i-1][j] \text{ if } i > j \end{cases}$$

The total value when total length is **j** and we have the first **i** pieces

If the piece is greater than the length of the rod → we skip it

EXAMPLE

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

	0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
$0 – 0m						
$2 – 1m						
$5 – 2m						
$7 – 3m						
3$ - 4m						

length

pieces

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
No cuts	**$0 – 0m**						
Just one	**$2 – 1m**						
First two	**$5 – 2m**						
First three	**$7 – 3m**						
All	**3$ - 4m**						

pieces **length**

Subproblems: we consider the lengths {0,1,2,3,4,5} step by step when we can have {1,2,3,4} unit lengths at the same time !!! We solve the subproblems and combine them for the final solution

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**					
First two	**$5 – 2m**	**0**					
First three	**$7 – 3m**	**0**					
All	**3$ - 4m**	**0**					

pieces **length**

Subproblems: we consider the lengths {0,1,2,3,4,5} step by step when we can have {1,2,3,4} unit lengths at the same time !!! We solve the subproblems and combine them for the final solution

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**				
First two	**$5 – 2m**	**0**					
First three	**$7 – 3m**	**0**					
All	**3$ - 4m**	**0**					

pieces — **length**

Subproblems: we consider the lengths {0,1,2,3,4,5} step by step when we can have {1,2,3,4} unit lengths at the same time !!! We solve the subproblems and combine them for the final solution

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**			
First two	**$5 – 2m**	**0**					
First three	**$7 – 3m**	**0**					
All	**3$ - 4m**	**0**					

pieces — **length**

Subproblems: we consider the lengths {0,1,2,3,4,5} step by step when we can have {1,2,3,4} unit lengths at the same time !!! We solve the subproblems and combine them for the final solution

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**		
First two	**$5 – 2m**	**0**					
First three	**$7 – 3m**	**0**					
All	**3$ - 4m**	**0**					

pieces — **length**

Subproblems: we consider the lengths {0,1,2,3,4,5} step by step when we can have {1,2,3,4} unit lengths at the same time !!! We solve the subproblems and combine them for the final solution

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	
First two	**$5 – 2m**	**0**					
First three	**$7 – 3m**	**0**					
All	**3$ - 4m**	**0**					

pieces (rows)

length (columns)

Subproblems: we consider the lengths {0,1,2,3,4,5} step by step when we can have {1,2,3,4} unit lengths at the same time !!! We solve the subproblems and combine them for the final solution

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**					
First three	**$7 – 3m**	**0**					
All	**3$ - 4m**	**0**					

pieces — **length**

Subproblems: we consider the lengths {0,1,2,3,4,5} step by step when we can have {1,2,3,4} unit lengths at the same time !!! We solve the subproblems and combine them for the final solution

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
							length
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**				
First three	**$7 – 3m**	**0**					
All	**3$ - 4m**	**0**					

pieces

$$dpTable[i][j] = \begin{cases} 0 \text{ if } j = 0 \\ \max\{ dpTable[i-1][j] ;\ prices[i] + dpTable[i][j-i] \} \text{ if } i \le j \\ dpTable[i-1][j] \text{ if } i > j \end{cases}$$

dpTable[2][1] = dpTable[1][1]

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
							length
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**			
First three	**$7 – 3m**	**0**					
All	**3$ - 4m**	**0**					

pieces

$$\textbf{dpTable[i][j]} = \begin{cases} 0 \text{ if } j = 0 \\ \max\{\, \text{dpTable}[i-1][j] \;;\; \text{prices}[i] + \text{dpTable}[i][j-i] \,\} \text{ if } i \le j \\ \text{dpTable}[i-1][j] \text{ if } i > j \end{cases}$$

dpTable[2][2] = MAX { dpTable[1][2] ; 5 + dpTable[2][0] }

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
							length
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**		
First three	**$7 – 3m**	**0**					
All	**3$ - 4m**	**0**					

pieces

$$\text{dpTable}[i][j] = \begin{cases} 0 \text{ if } j = 0 \\ \max\{\, \text{dpTable}[i-1][j] \;;\; \text{prices}[i] + \text{dpTable}[i][j-i] \,\} \text{ if } i \le j \\ \text{dpTable}[i-1][j] \text{ if } i > j \end{cases}$$

dpTable[2][3] = MAX { dpTable[1][3] ; 5 + dpTable[2][1] }

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
							length
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	
First three	**$7 – 3m**	**0**					
All	**3$ - 4m**	**0**					

pieces

$$\textbf{dpTable[i][j]} = \begin{cases} 0 \text{ if } j = 0 \\ \max\{\, \text{dpTable}[i-1][j] \;;\; \text{prices}[i] + \text{dpTable}[i][j-i] \,\} \text{ if } i \le j \\ \text{dpTable}[i-1][j] \text{ if } i > j \end{cases}$$

dpTable[2][4] = MAX { dpTable[1][4] ; 5 + dpTable[2][2] }

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
							length
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**					
All	**3$ - 4m**	**0**					

pieces

$$\text{dpTable}[i][j] = \begin{cases} 0 \text{ if } j = 0 \\ \max\{\, \text{dpTable}[i-1][j] \;;\; \text{prices}[i] + \text{dpTable}[i][j-i] \,\} \text{ if } i \le j \\ \text{dpTable}[i-1][j] \text{ if } i > j \end{cases}$$

dpTable[2][5] = MAX { dpTable[1][5] ; 5 + dpTable[2][3] }

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
	length						
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**				
All	**3$ - 4m**	**0**					

pieces

$$\text{dpTable}[i][j] = \begin{cases} 0 \text{ if } j = 0 \\ \max\{\, \text{dpTable}[i-1][j] \;;\; \text{prices}[i] + \text{dpTable}[i][j-i] \,\} \text{ if } i \le j \\ \text{dpTable}[i-1][j] \text{ if } i > j \end{cases}$$

dpTable[3][1] = dpTable[2][1]

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
	length						
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**			
All	**3$ - 4m**	**0**					

pieces

$$\text{dpTable}[i][j] = \begin{cases} 0 \text{ if } j = 0 \\ \max\{\, \text{dpTable}[i-1][j] \;;\; \text{prices}[i] + \text{dpTable}[i][j-i] \,\} \text{ if } i \le j \\ \text{dpTable}[i-1][j] \text{ if } i > j \end{cases}$$

dpTable[3][2] = dpTable[2][2]

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
							length
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**		
All	**3$ - 4m**	**0**					

pieces

$$\text{dpTable}[i][j] = \begin{cases} 0 \text{ if } j = 0 \\ \max\{\, \text{dpTable}[i-1][j] \;;\; \text{prices}[i] + \text{dpTable}[i][j-i] \,\} \text{ if } i \le j \\ \text{dpTable}[i-1][j] \text{ if } i > j \end{cases}$$

dpTable[3][3] = MAX { dpTable[2][3] ; 7 + dpTable[3][0] }

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
							length
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**	**10**	
All	**3$ - 4m**	**0**					

pieces

$$\textbf{dpTable[i][j]} = \begin{cases} 0 \text{ if } j = 0 \\ \max\{\, \text{dpTable}[i-1][j] \;;\; \text{prices}[i] + \text{dpTable}[i][j-i] \,\} \text{ if } i \le j \\ \text{dpTable}[i-1][j] \text{ if } i > j \end{cases}$$

dpTable[3][4] = MAX { dpTable[2][4] ; 7 + dpTable[3][1] }

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
							length
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**	**10**	**12**
All	**3$ - 4m**	**0**					

pieces

$$\textbf{dpTable[i][j]} = \begin{cases} 0 \text{ if } j = 0 \\ \max\{\, \text{dpTable}[i-1][j] \;;\; \text{prices}[i] + \text{dpTable}[i][j-i] \,\} \text{ if } i \le j \\ \text{dpTable}[i-1][j] \text{ if } i > j \end{cases}$$

dpTable[3][5] = MAX { dpTable[2][5] ; 7 + dpTable[3][2] }

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
							length
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**	**10**	**12**
All	**3$ - 4m**	**0**	**2**				

pieces

$$\text{dpTable}[i][j] = \begin{cases} 0 \text{ if } j = 0 \\ \max\{\, \text{dpTable}[i-1][j] \;;\; \text{prices}[i] + \text{dpTable}[i][j-i] \,\} \text{ if } i \le j \\ \text{dpTable}[i-1][j] \text{ if } i > j \end{cases}$$

dpTable[4][1] = dpTable[3][1]

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
		length					
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**	**10**	**12**
All	**3$ - 4m**	**0**	**2**	**5**			

pieces

$$\text{dpTable}[i][j] = \begin{cases} 0 \text{ if } j = 0 \\ \max\{\, \text{dpTable}[i-1][j] \;;\; \text{prices}[i] + \text{dpTable}[i][j-i] \,\} \text{ if } i \le j \\ \text{dpTable}[i-1][j] \text{ if } i > j \end{cases}$$

dpTable[4][2] = dpTable[3][2]

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
							length
No cuts	**$0 – 0m**	0	0	0	0	0	0
Just one	**$2 – 1m**	0	2	4	6	8	10
First two	**$5 – 2m**	0	2	5	7	10	12
First three	**$7 – 3m**	0	2	5	7	10	12
All	**3$ - 4m**	0	2	5	7		

pieces

$$dpTable[i][j] = \begin{cases} 0 \text{ if } j = 0 \\ \max\{\, dpTable[i-1][j] \;;\; prices[i] + dpTable[i][j-i] \,\} \text{ if } i \le j \\ dpTable[i-1][j] \text{ if } i > j \end{cases}$$

dpTable[4][3] = dpTable[3][3]

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
							length
No cuts	**$0 – 0m**	0	0	0	0	0	0
Just one	**$2 – 1m**	0	2	4	6	8	10
First two	**$5 – 2m**	0	2	5	7	10	12
First three	**$7 – 3m**	0	2	5	7	10	12
All	**3$ - 4m**	0	2	5	7	10	

pieces

$$dpTable[i][j] = \begin{cases} 0 \text{ if } j = 0 \\ \max\{\, dpTable[i-1][j] \;;\; prices[i] + dpTable[i][j-i] \,\} \text{ if } i \le j \\ dpTable[i-1][j] \text{ if } i > j \end{cases}$$

dpTable[4][4] = MAX { dpTable[3][4] ; 3 + dpTable[4][0] }

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
							length
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**	**10**	**12**
All	**3$ - 4m**	**0**	**2**	**5**	**7**	**10**	**12**

pieces

$$\textbf{dpTable[i][j]} = \begin{cases} 0 \text{ if } j = 0 \\ \max\{\, \text{dpTable}[i-1][j] \;;\; \text{prices}[i] + \text{dpTable}[i][j-i] \,\} \text{ if } i \le j \\ \text{dpTable}[i-1][j] \text{ if } i > j \end{cases}$$

dpTable[4][5] = MAX { dpTable[3][5] ; 3 + dpTable[4][1] }

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**	**10**	**12**
All	**3$ - 4m**	**0**	**2**	**5**	**7**	**10**	**12**

length

pieces

OK, we can make a **$12** profit ... but what are the optimal cuts?

SOLUTION:

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**	**10**	**12**
All	**3$ - 4m**	**0**	**2**	**5**	**7**	**10**	**12**

length

pieces

OK, we can make a **$12** profit ... but what are the optimal cuts?

SOLUTION:

It is coming from the cell above: it means there is no 4m cut in the solution

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**	**10**	**12**
All	**3$ - 4m**	**0**	**2**	**5**	**7**	**10**	**12**

length

pieces

OK, we can make a **$12** profit … but what are the optimal cuts?

SOLUTION:

It is coming from the cell above: it means there is no 3m cut in the solution

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**	**10**	**12**
All	**3$ - 4m**	**0**	**2**	**5**	**7**	**10**	**12**

length

pieces

OK, we can make a **$12** profit ... but what are the optimal cuts?

SOLUTION: 2m,

It is NOT coming from the cell above: it means there is a 2m cut in the solution

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
							length
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**	**10**	**12**
All	**3$ - 4m**	**0**	**2**	**5**	**7**	**10**	**12**

pieces

OK, we can make a **$12** profit … but what are the optimal cuts?

SOLUTION: 2m, 2m,

It is NOT coming from the cell above: it means there is a 2m cut again in the solution

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
							length
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**	**10**	**12**
All	**3$ - 4m**	**0**	**2**	**5**	**7**	**10**	**12**

pieces

OK, we can make a **$12** profit ... but what are the optimal cuts?

SOLUTION: 2m, 2m,

It is coming from the cell above: it means there is no more 2m cut in the solution

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
							length
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**	**10**	**12**
All	**3$ - 4m**	**0**	**2**	**5**	**7**	**10**	**12**

pieces

OK, we can make a **$12** profit ... but what are the optimal cuts?

SOLUTION: 2m, 2m, 1m

It is NOT coming from the cell above: it means there is a 1m cut in the solution

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**	**10**	**12**
All	**3$ - 4m**	**0**	**2**	**5**	**7**	**10**	**12**

length

pieces

OK, we can make a **$12** profit ... but what are the optimal cuts?

It is NOT coming from the cell above: it means there is a 1m cut in the solution

SOLUTION: 2m, 2m, 1m

We have to subtract the given price ($2) corresponding to the length (1m) from $2 ... go to that position
$2 - $2 = $0 → so we go to dpTable[1][0] !!!

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**	**10**	**12**
All	**3$ - 4m**	**0**	**2**	**5**	**7**	**10**	**12**

length

pieces

OK, we can make a **$12** profit ... but what are the optimal cuts?

It is NOT coming from the cell above: it means there is a 1m cut in the solution

SOLUTION: 2m, 2m, 1m

We have to subtract the given price ($2) corresponding to the length (1m) from $2 ... go to that position
$2 - $2 = $0 → so we go to dpTable[1][0] !!!

l = 5m
0m → $0 1m → $2 2m → $5 3m → 7$ 4m → 3$
prices[] = {0, 2, 5, 7, 3}

numOfColumns = l +1
numOfRows = prices.length+1

		0 [m]	1 [m]	2 [m]	3 [m]	4 [m]	5 [m]
No cuts	**$0 – 0m**	**0**	**0**	**0**	**0**	**0**	**0**
Just one	**$2 – 1m**	**0**	**2**	**4**	**6**	**8**	**10**
First two	**$5 – 2m**	**0**	**2**	**5**	**7**	**10**	**12**
First three	**$7 – 3m**	**0**	**2**	**5**	**7**	**10**	**12**
All	**3$ - 4m**	**0**	**2**	**5**	**7**	**10**	**12**

length

pieces

OK, we can make a **$12** profit ... but what are the optimal cuts?

SOLUTION: 2m, 2m, 1m ... We will have 2 2m length cut and a single 1m cut, this is the optimal solution and we can make $12 profit