



Bölüm 6: Dinamik Programlama

Algoritmalar



Dinamik Programlama

- Bazen karmaşık problemleri çözmek için akıllı yöntemlere ihtiyaç olur.
- Dinamik programlama bu akıllı yöntemlerden biridir.
- Büyük problemleri küçük, kolay çözülebilir parçalara bölgerek
 - adım adım çözümeyi sağlar.





Dinamik Programlama

- Karmaşık problemleri küçük, tekrarlayan alt problemlere bölgerek çözer.
- Bu yöntem, alt problemleri çözerek oluşturulan bir tablo kullanır.
- Aynı alt problem ile tekrar karşılaşıldığında,
 - tablodan bakılarak çözüm bulunur,
 - böylece zaman kazanılır.
- Optimizasyon problemlerinde sıkılıkla kullanılır.
- *Every problem of dynamic programming can be solved by plain recursion.*

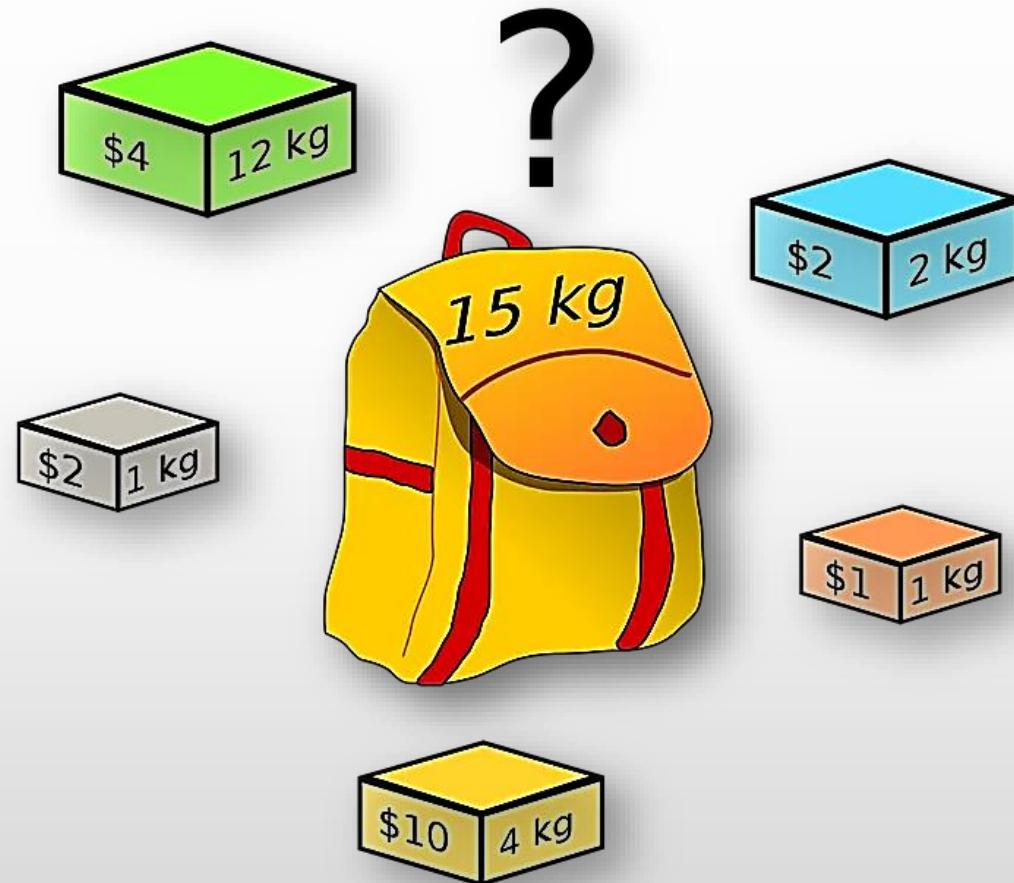


Dinamik Programlama Uygulamaları

- En kısa yol bulma: GPS cihazları, haritalar
- Dizi elemanlarının bir sıraya göre dizilmesi: Bilgisayar programları
- RNA dizilerinin yapısal özelliklerinin belirlenmesi: Biyoinformatik
- Yapay zeka uygulamaları: Satranç oynayan bilgisayarlar, makine çevirisi

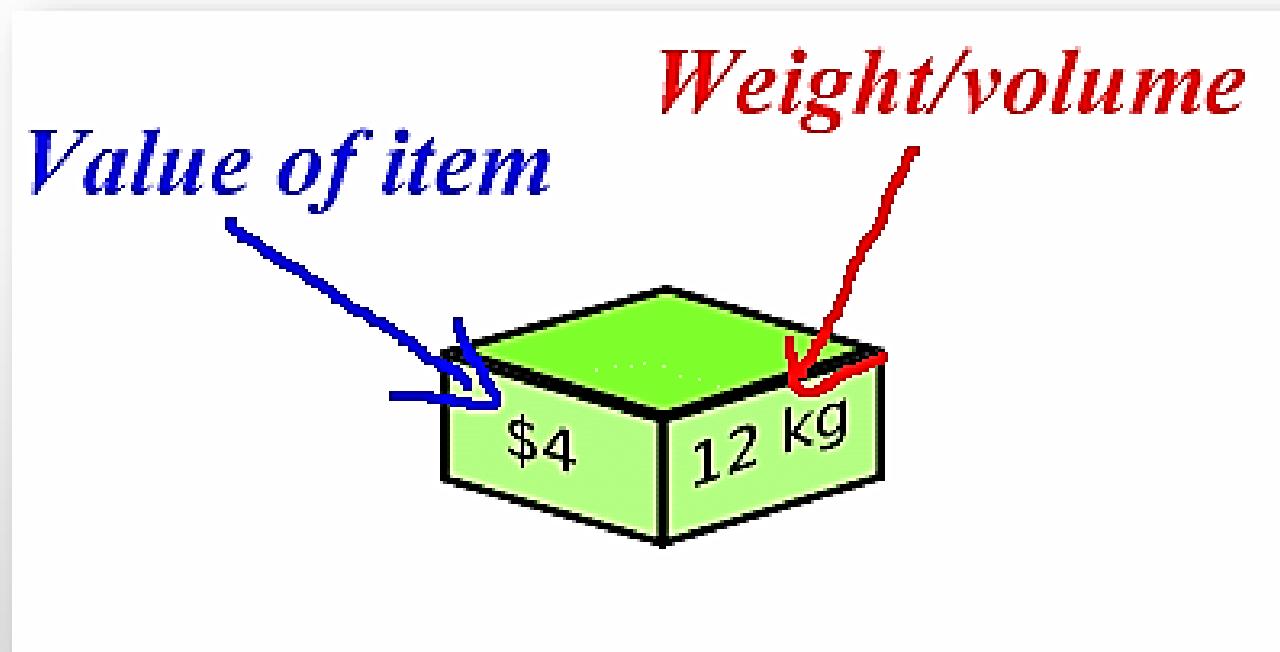


0/1 Knapsack





0/1 Knapsack





Knapsack

i	v	w		w	0	1	2	3	4	5	6
				i	0	1	2	3	4	5	6
1	5	4			0	0	0	0	0	0	0
2	4	3			1	0	0	0			
3	3	2			2	0					
4	2	1			3	0					
<i>Capacity=6</i>					4	0					



Knapsack

i	v	w		w	0	1	2	3	4	5	6
1	5	4		i	0	0	0	0	0	0	0
2	4	3		1	0	0	0	0			
3	3	2		2	0						
4	2	1		3	0						
<i>Capacity=6</i>				4	0						



Knapsack

i	v	w		w	0	1	2	3	4	5	6
				i	0	1	2	3	4	5	6
1	5	4			0	0	0	0	0	0	0
2	4	3			1	0	0	0	0	5	
3	3	2			2	0					
4	2	1			3	0					
<i>Capacity=6</i>					4	0					



Knapsack

i	v	w		i	0	1	2	3	4	5	6
1	5	4		0	0	0	0	0	0	0	0
2	4	3		1	0	0	0	0	5	5	
3	3	2		2	0						
4	2	1		3	0						
<i>Capacity=6</i>				4	0						



Knapsack

i	v	w		i	0	1	2	3	4	5	6
1	5	4		0	0	0	0	0	0	0	0
2	4	3		1	0	0	0	0	5	5	5
3	3	2		2	0						
4	2	1		3	0						
<i>Capacity=6</i>				4	0						



Knapsack

i	v	w		i	0	1	2	3	4	5	6
1	5	4		0	0	0	0	0	0	0	0
2	4	3		1	0	0	0	0	5	5	5
3	3	2		2	0	0					
4	2	1		3	0						
<i>Capacity=6</i>				4	0						



Knapsack

i	v	w		w	0	1	2	3	4	5	6
				i	0	1	2	3	4	5	6
1	5	4			0	0	0	0	0	0	0
2	4	3			1	0	0	0	0	5	5
3	3	2			2	0	0	0			
4	2	1			3	0					
<i>Capacity=6</i>					4	0					



Knapsack

i	v	w								
			w	0	1	2	3	4	5	6
i	0			0	0	0	0	0	0	0
1	5	4								
2	4	3		0	0	0	0	5	5	5
3	3	2		0	0	0	4			
4	2	1		0						
<i>Capacity=6</i>				0						



Knapsack

i	v	w								
			w	0	1	2	3	4	5	6
1	5	4	i	0	0	0	0	0	0	0
2	4	3	1	0	0	0	0	5	5	5
3	3	2	2	0	0	0	4	5		
4	2	1	3	0						
<i>Capacity=6</i>			4	0						



Knapsack

i	v	w		w	0	1	2	3	4	5	6
				i	0	1	2	3	4	5	6
1	5	4			0	0	0	0	0	0	0
2	4	3			1	0	0	0	5	5	5
3	3	2			2	0	0	4	5	5	
4	2	1			3	0					
<i>Capacity=6</i>					4	0					



Knapsack

i	v	w		w	0	1	2	3	4	5	6
				i	0	1	2	3	4	5	6
1	5	4			0	0	0	0	0	0	0
2	4	3			1	0	0	0	5	5	5
3	3	2			2	0	0	4	5	5	5
4	2	1			3	0					
<i>Capacity=6</i>					4	0					



Knapsack

i	v	w		w	0	1	2	3	4	5	6
			i	0	0	0	0	0	0	0	0
1	5	4		1	0	0	0	0	0	0	0
2	4	3		2	0	0	0	0	5	5	5
3	3	2		3	0	0	4	5	5	5	5
4	2	1		4	0						
<i>Capacity=6</i>											



Knapsack

i	v	w		w	0	1	2	3	4	5	6
			i	0	0	0	0	0	0	0	0
1	5	4		1	0	0	0	0	0	0	0
2	4	3		2	0	0	0	0	5	5	5
3	3	2		3	0	0	0	4	5	5	5
4	2	1		4	0						
<i>Capacity=6</i>											



Knapsack

i	v	w								
			w	0	1	2	3	4	5	6
1	5	4	i	0	0	0	0	0	0	0
2	4	3	1	0	0	0	0	5	5	5
3	3	2	2	0	0	0	4	5	5	5
4	2	1	3	0	0	3	4			
<i>Capacity=6</i>			4	0						



Knapsack

i	v	w		w	0	1	2	3	4	5	6
			i	0	0	0	0	0	0	0	0
1	5	4		1	0	0	0	0	5	5	5
2	4	3		2	0	0	0	4	5	5	5
3	3	2		3	0	0	3	4	5		
4	2	1		4	0						
<i>Capacity=6</i>											



Knapsack

i	v	w								
			w	0	1	2	3	4	5	6
i	0			0	0	0	0	0	0	0
1	5	4		0	0	0	0	0	0	0
2	4	3		0	0	0	0	5	5	5
3	3	2		0	0	0	4	5	5	5
4	2	1		0	0	3	4	5	7	
<i>Capacity=6</i>				0						



Knapsack

i	v	w		i	w	0	1	2	3	4	5	6
1	5	4		0	0	0	0	0	0	0	0	0
2	4	3		1	0	0	0	0	5	5	5	
3	3	2		2	0	0	0	4	5	5	5	
4	2	1		3	0	0	3	4	5	7	8	
<i>Capacity=6</i>				4	0							



Knapsack

i	v	w		i	w	0	1	2	3	4	5	6
1	5	4		0	0	0	0	0	0	0	0	0
2	4	3		1	0	0	0	0	5	5	5	
3	3	2		2	0	0	0	4	5	5	5	
4	2	1	<i>Capacity=6</i>	3	0	0	3	4	5	7	8	
				4	0	2						



Knapsack

i	v	w		w	0	1	2	3	4	5	6
				i	0	1	2	3	4	5	6
1	5	4		0	0	0	0	0	0	0	0
2	4	3		1	0	0	0	0	5	5	5
3	3	2		2	0	0	0	4	5	5	5
4	2	1	$Capacity=6$	3	0	0	3	4	5	7	8
				4	0	2	3				



Knapsack

i	v	w								
			w	0	1	2	3	4	5	6
i	0			0	0	0	0	0	0	0
1	5	4		0	0	0	0	0	0	0
2	4	3		0	0	0	0	5	5	5
3	3	2		0	0	0	4	5	5	5
4	2	1		0	0	3	4	5	7	8
<i>Capacity=6</i>				0	2	3	5			



Knapsack

i	v	w								
			w	0	1	2	3	4	5	6
i	0			0	0	0	0	0	0	0
1	5	4		0	0	0	0	0	0	0
2	4	3		0	0	0	0	5	5	5
3	3	2		0	0	0	4	5	5	5
4	2	1		0	0	3	4	5	7	8
<i>Capacity=6</i>				0	2	3	5	6		



Knapsack

i	v	w		i	w	0	1	2	3	4	5	6
1	5	4		0	0	0	0	0	0	0	0	0
2	4	3		1	0	0	0	0	5	5	5	
3	3	2		2	0	0	0	4	5	5	5	
4	2	1	<i>Capacity=6</i>	3	0	0	3	4	5	7	8	
				4	0	2	3	5	6	7		



Knapsack

i	v	w		i	w	0	1	2	3	4	5	6
1	5	4		0	0	0	0	0	0	0	0	0
2	4	3		1	0	0	0	0	5	5	5	
3	3	2		2	0	0	0	4	5	5	5	
4	2	1	<i>Capacity=6</i>	3	0	0	3	4	5	7	8	
				4	0	2	3	5	6	7	9	



Knapsack

i	v	w		i	w	0	1	2	3	4	5	6
1	5	4		0	0	0	0	0	0	0	0	0
2	4	3		1	0	0	0	0	5	5	5	
3	3	2		2	0	0	0	4	5	5	5	
4	2	1		3	0	0	3	4	5	7	8	
<i>Capacity=6</i>				4	0	2	3	5	6	7	8	9



Knapsack

i	v	w		i	w	0	1	2	3	4	5	6
1	5	4		0	0	0	0	0	0	0	0	0
2	4	3		1	0	0	0	0	5	5	5	
3	3	2		2	0	0	0	4	5	5	5	
4	2	1		3	0	0	3	4	5	7	8	
<i>Capacity=6</i>				4	0	2	3	5	6	7	9	



Knapsack

i	v	w								
			w	0	1	2	3	4	5	6
1	5	4	i	0	0	0	0	0	0	0
2	4	3	1	0	0	0	0	5	5	5
3	3	2	2	0	0	0	4	5	5	5
4	2	1	3	0	0	3	4	5	7	8
<i>Capacity=6</i>			4	0	2	3	5	6	7	9



Knapsack

i	v	w								
			w	0	1	2	3	4	5	6
1	5	4		0	0	0	0	0	0	0
2	4	3		0	0	0	0	5	5	5
3	3	2		0	0	0	4	5	5	5
4	2	1		0	0	3	4	5	7	8
<i>Capacity=6</i>				0	2	3	5	6	7	9



Knapsack

i	v	w								
			w	0	1	2	3	4	5	6
1	5	4	i	0	0	0	0	0	0	0
2	4	3	1	0	0	0	0	5	5	5
3	3	2	2	0	0	0	4	5	5	5
4	2	1	3	0	0	3	4	5	7	8
<i>Capacity=6</i>			4	0	2	3	5	6	7	9





Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0								
50	3	1	0								
60	4	2	0								
90	5	3	0								

With '0' capacity, maximum profit we can have for every subarray is '0'

1 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15							
50	3	1	0								
60	4	2	0								
90	5	3	0								

Capacity = 1, Index = 0, i.e., if we consider the sub-array till index '0', maximum profit will be '15', as we can fit the item with weight '1' in the knapsack



2 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30						
50	3	1	0								
60	4	2	0								
90	5	3	0								

Capacity = 2, Index = 0, => profits[index] + dp[index][1], we are not considering dp[index-1][Capacity] as Index is not bigger than '0'

3 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45					
50	3	1	0								
60	4	2	0								
90	5	3	0								

Capacity = 3, Index = 0, => profits[index] + dp[index][1]

4 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0								
60	4	2	0								
90	5	3	0								

Capacity = 4-8, Index = 0, => profits[index] + dp[index][1]

5 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15							
60	4	2	0								
90	5	3	0								

Capacity = 1, Index = 1, since item at index '1' has weight '3', which is greater than the capacity '1', so we will take the dp[index-1][capacity]

6 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30						
60	4	2	0								
90	5	3	0								

Capacity = 2, Index = 1, since item at index '1' has weight '3', which is greater than the capacity '2', so we will take the dp[index-1][capacity]

7 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50					
60	4	2	0								
90	5	3	0								

Capacity = 3, Index = 1, from the formula discussed above: $\max(dp[0][3], profit[1] + dp[1][0])$

8 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50	65				
60	4	2	0								
90	5	3	0								

Capacity = 4, Index = 1, from the formula discussed above: $\max(dp[0][4], profit[1] + dp[1][1])$

9 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50	65	80			
60	4	2	0								
90	5	3	0								

Capacity = 5, Index = 1, from the formula discussed above: $\max(dp[0][5], profit[1] + dp[1][2])$





Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50	65	80	100		
60	4	2	0								
90	5	3	0								

Capacity = 6, Index = 1, from the formula discussed above: $\max\{ dp[0][6], profit[1] + dp[1][3] \}$



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8	⚙️	💡
15	1	0	0	15	30	45	60	75	90	105	120		
50	3	1	0	15	30	50	65	80	100	115			
60	4	2	0										
90	5	3	0										

Capacity = 7, Index = 1, from the formula discussed above: $\max(dp[0][7], profit[1] + dp[1][4])$

12 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50	65	80	100	115	130
60	4	2	0								
90	5	3	0								

Capacity = 8, Index = 1, from the formula discussed above: $\max(dp[0][8], profit[1] + dp[1][5])$

13 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50	65	80	100	115	130
60	4	2	0	15	30	50					
90	5	3	0								

Capacity = 1-3, Index =2, since item at index '2' has weight '4', which is greater than the capacity, so we will take the dp[index-1][capacity]

14 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50	65	80	100	115	130
60	4	2	0	15	30	50	65				
90	5	3	0								

Capacity = 4, Index = 2, from the formula discussed above: $\max(dp[1][4], profit[2] + dp[2][0])$

15 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50	65	80	100	115	130
60	4	2	0	15	30	50	65	80			
90	5	3	0								

Capacity = 5, Index = 2; from the formula discussed above: $\max\{ dp[1][5], profit[2] + dp[2][1] \}$

16 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50	65	80	100	115	130
60	4	2	0	15	30	50	65	80	100		
90	5	3	0								

Capacity = 6, Index = 2, from the formula discussed above: $\max(dp[1][6], profit[2] + dp[2][2])$

17 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50	65	80	100	115	130
60	4	2	0	15	30	50	65	80	100	115	
90	5	3	0								

Capacity = 7, Index = 2, from the formula discussed above: $\max(dp[1][7], profit[2] + dp[2][3])$

18 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50	65	80	100	115	130
60	4	2	0	15	30	50	65	80	100	115	130
90	5	3	0								

Capacity = 8, Index = 2, from the formula discussed above: $\max\{ dp[1][8], profit[2] + dp[2][4] \}$

19 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50	65	80	100	115	130
60	4	2	0	15	30	50	65	80	100	115	130
90	5	3	0	15	30	50	65				

Capacity = 1-4, Index =3, since item at index '3' has weight '5', which is greater than the capacity, so we will take the dp[index-1][capacity]

20 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50	65	80	100	115	130
60	4	2	0	15	30	50	65	80	100	115	130
90	5	3	0	15	30	50	65	90			

Capacity = 5, Index =3, from the formula discussed above: $\max\{ dp[2][5], profit[3] + dp[3][0] \}$

21 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50	65	80	100	115	130
60	4	2	0	15	30	50	65	80	100	115	130
90	5	3	0	15	30	50	65	90	105		

Capacity = 6, Index = 3, from the formula discussed above: $\max(dp[2][6], profit[3] + dp[3][1])$

22 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50	65	80	100	115	130
60	4	2	0	15	30	50	65	80	100	115	130
90	5	3	0	15	30	50	65	90	105	120	

Capacity = 7, Index =3, from the formula discussed above: $\max(dp[2][7], profit[3] + dp[3][2])$

23 of 24



Unbounded Knapsack

profit	weight	index	0	1	2	3	4	5	6	7	8
15	1	0	0	15	30	45	60	75	90	105	120
50	3	1	0	15	30	50	65	80	100	115	130
60	4	2	0	15	30	50	65	80	100	115	130
90	5	3	0	15	30	50	65	90	105	120	140

Capacity = 8, Index = 3, from the formula discussed above: $\max(dp[2][8], profit[3] + dp[3][3])$

24 of 24





Subset Sum

10	0	5	8	6	2	4
----	---	---	---	---	---	---

sum = 15

10	0	5	8	6	2	4
----	---	---	---	---	---	---

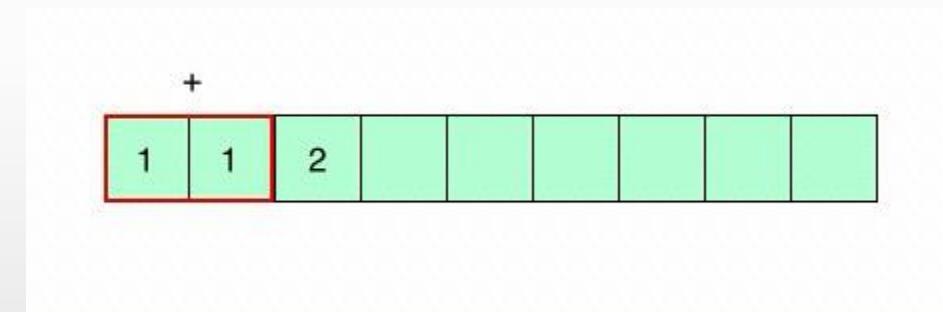
$$5 + 8 + 2 = 15$$





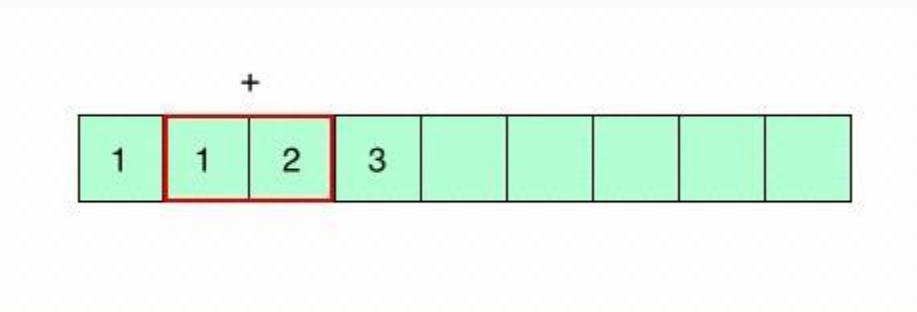


Fibonacci



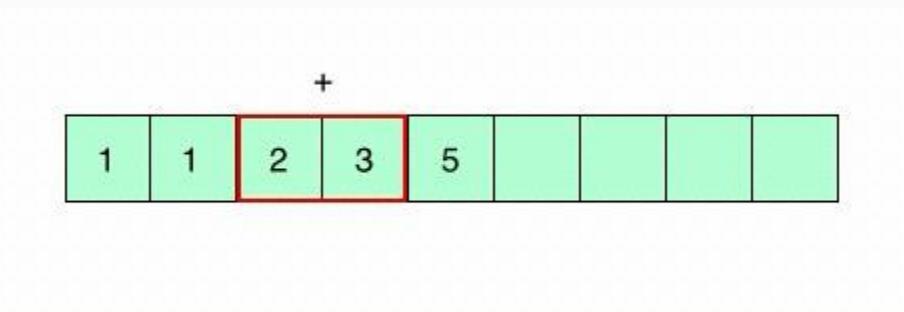


Fibonacci



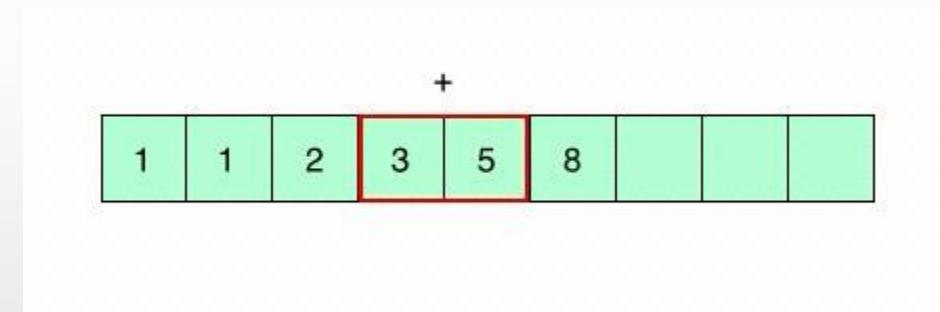


Fibonacci



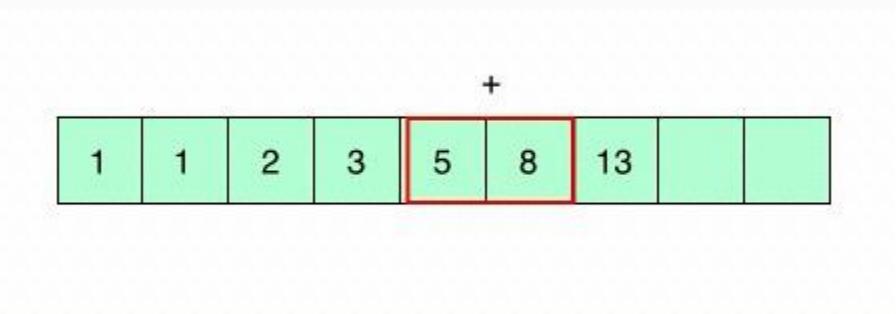


Fibonacci



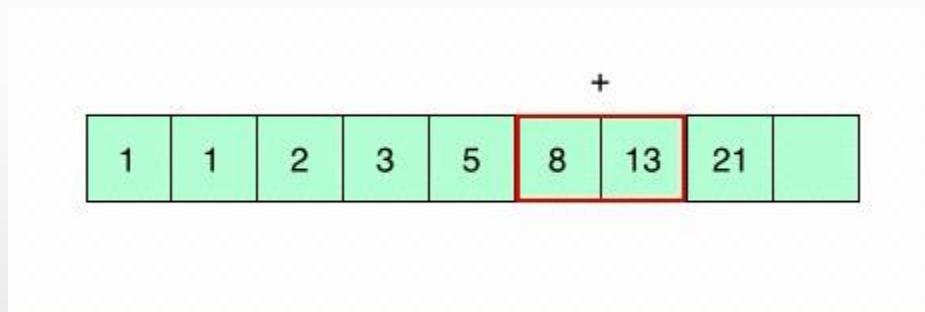


Fibonacci



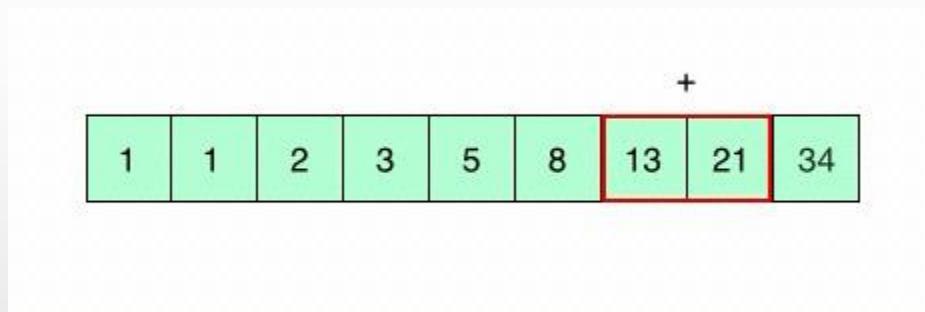


Fibonacci





Fibonacci



Fibonacci

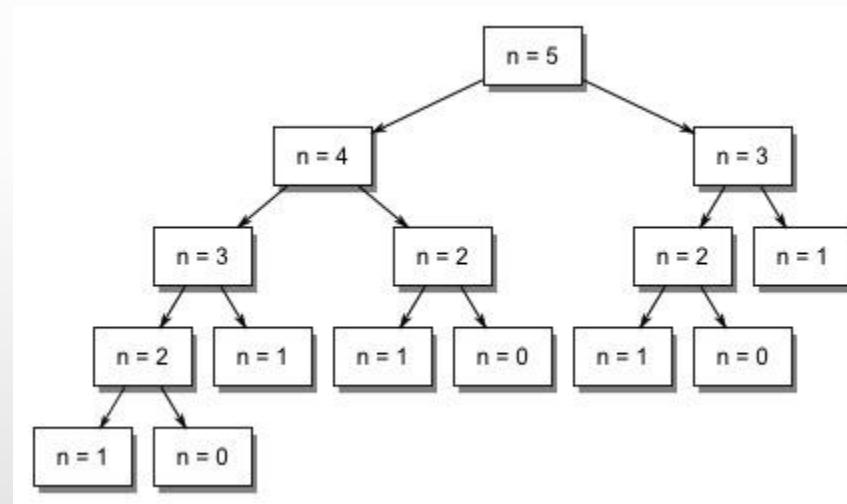


1	1	2	3	5	8	13	21	34
---	---	---	---	---	---	----	----	----



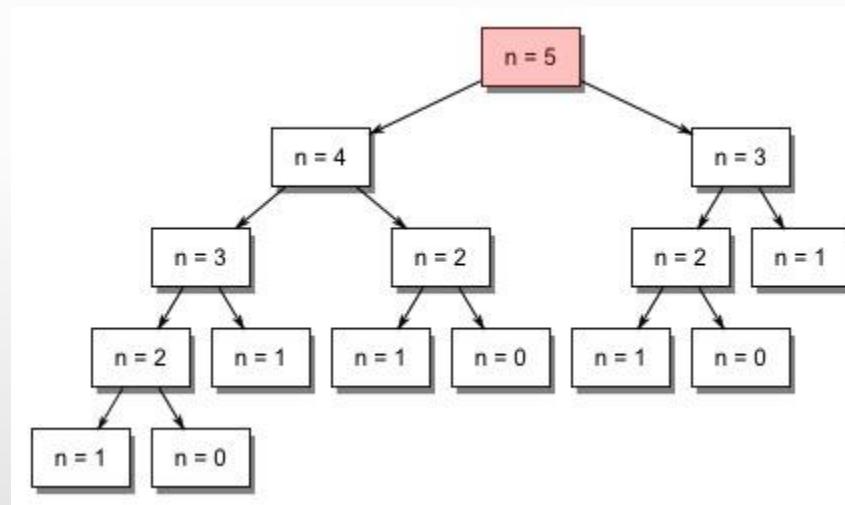


Fibonacci



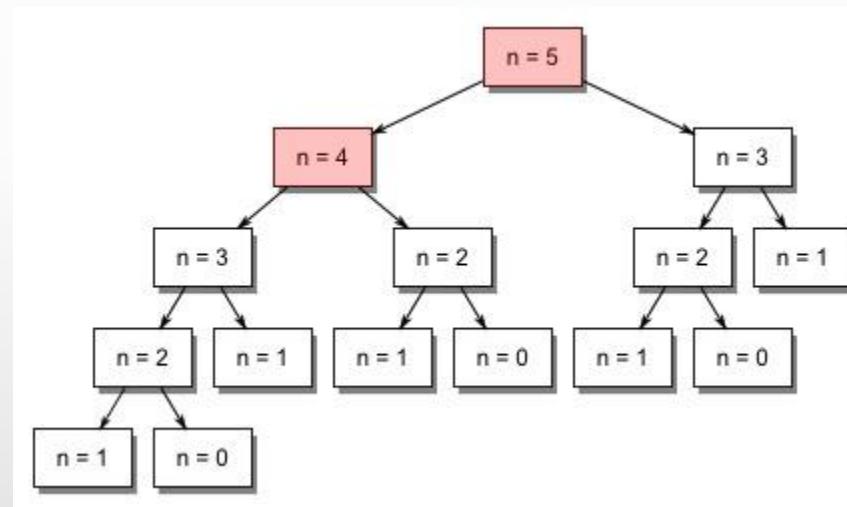


Fibonacci



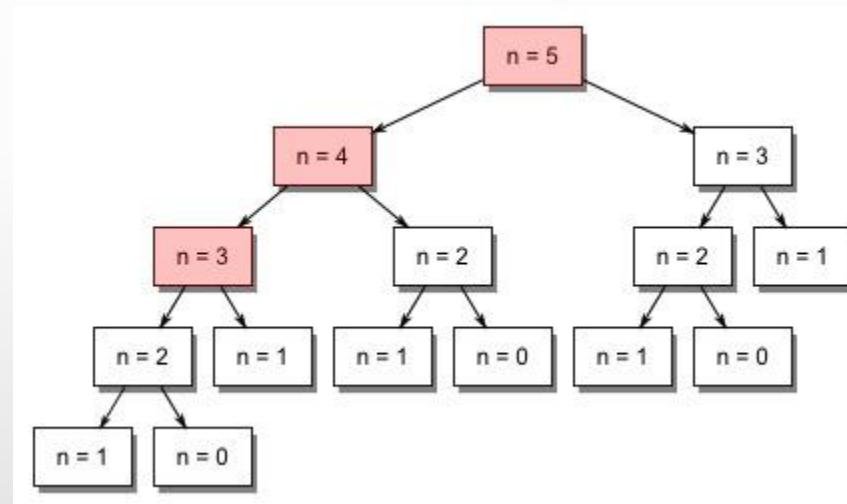


Fibonacci



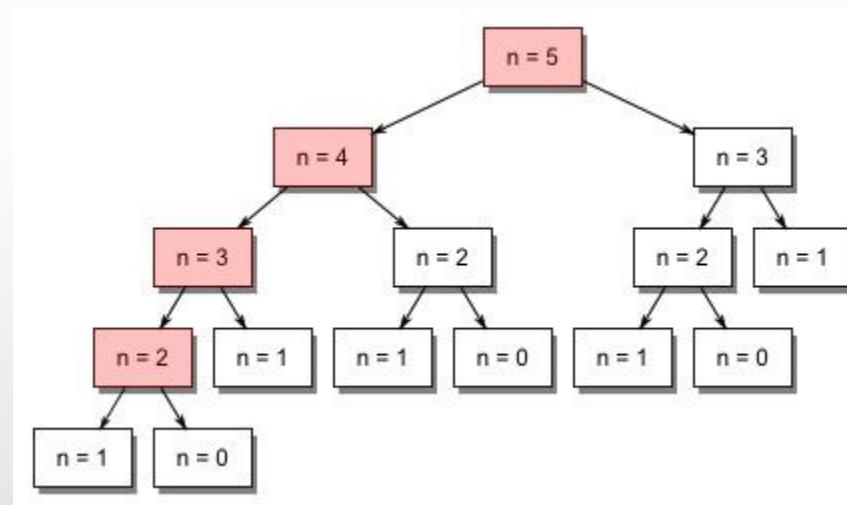


Fibonacci



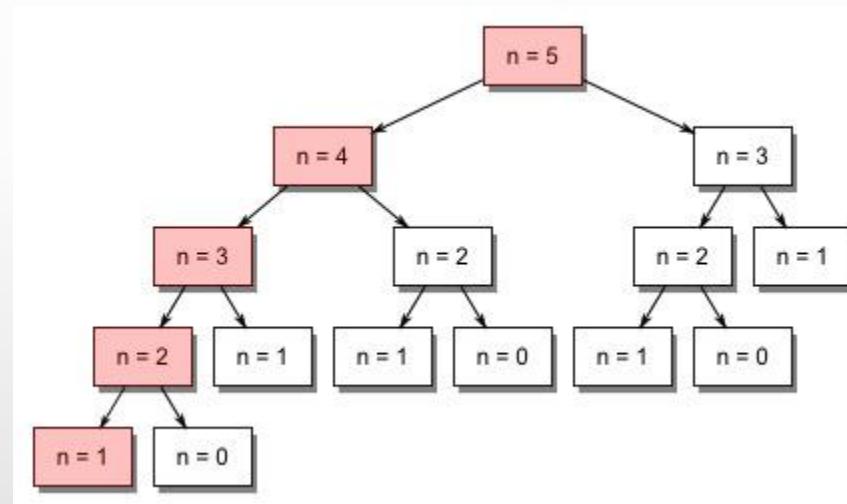


Fibonacci



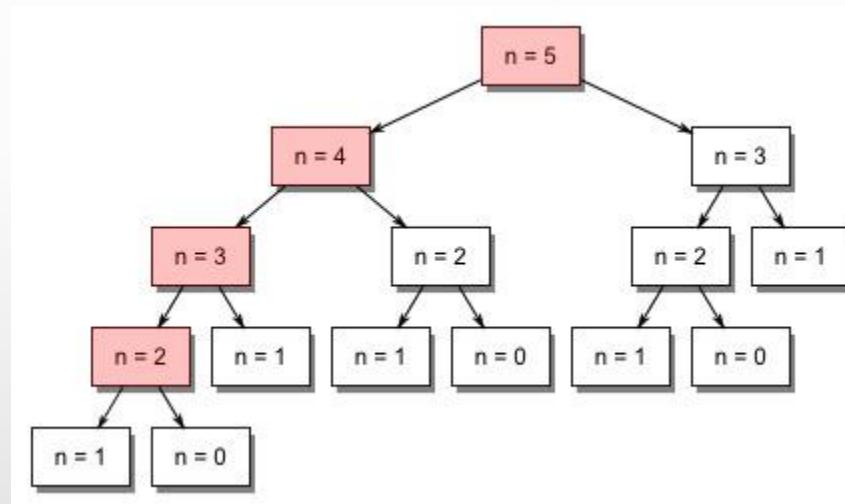


Fibonacci



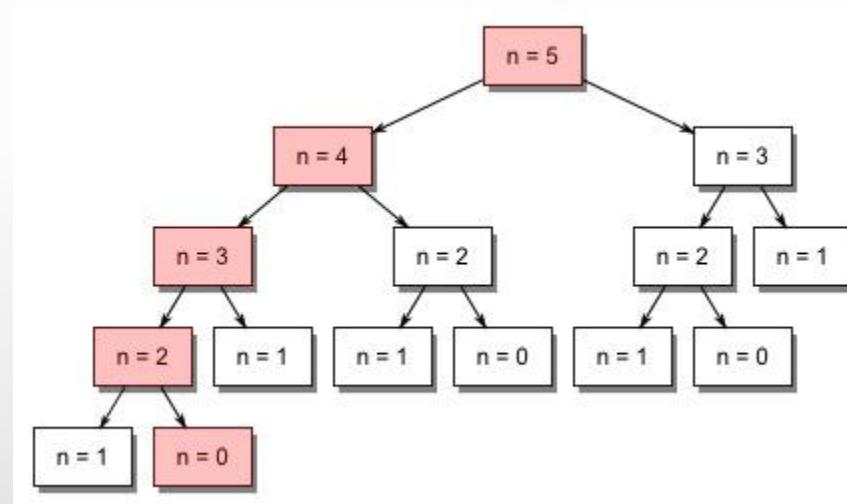


Fibonacci



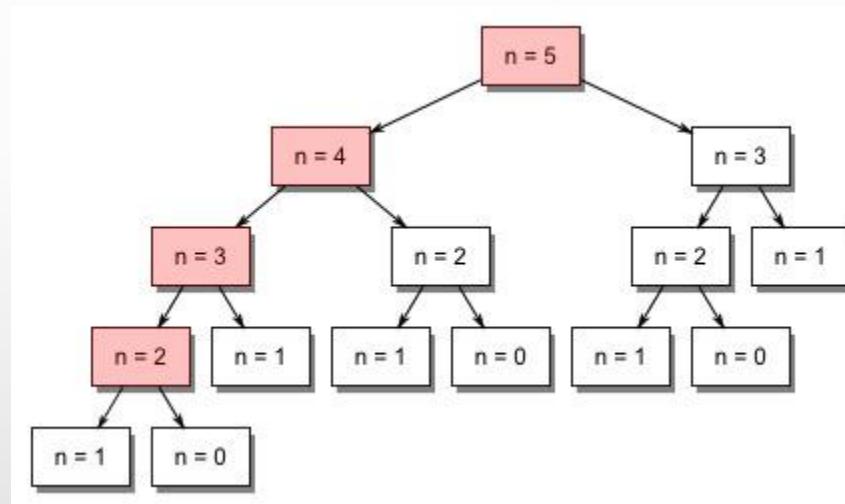


Fibonacci



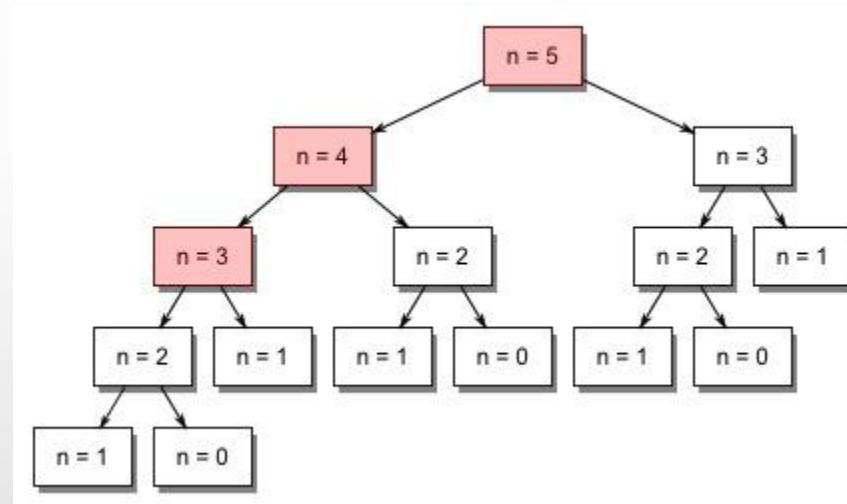


Fibonacci



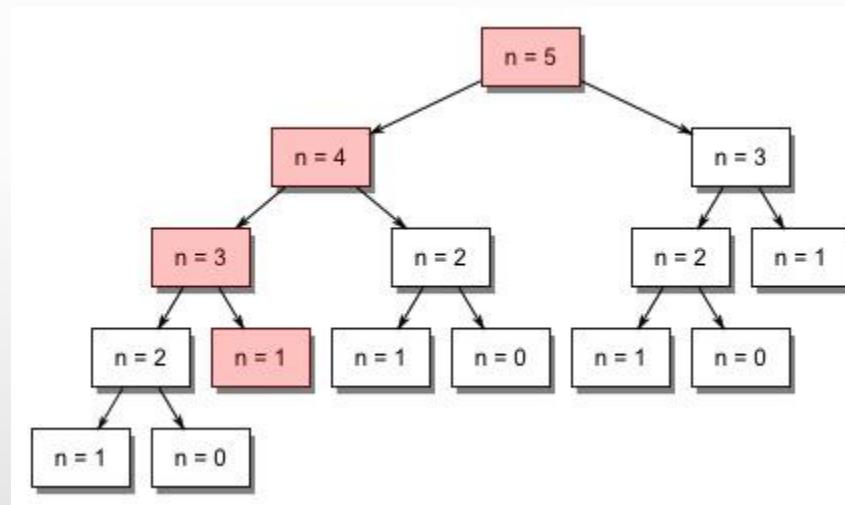


Fibonacci



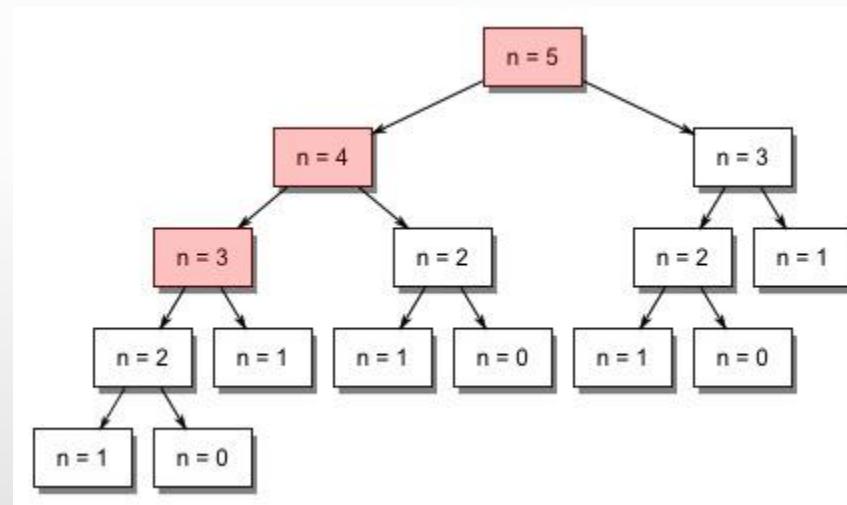


Fibonacci



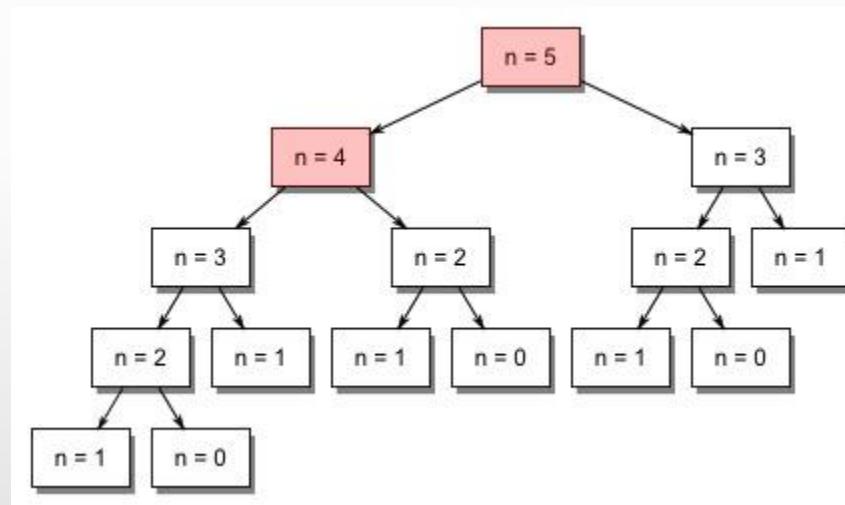


Fibonacci



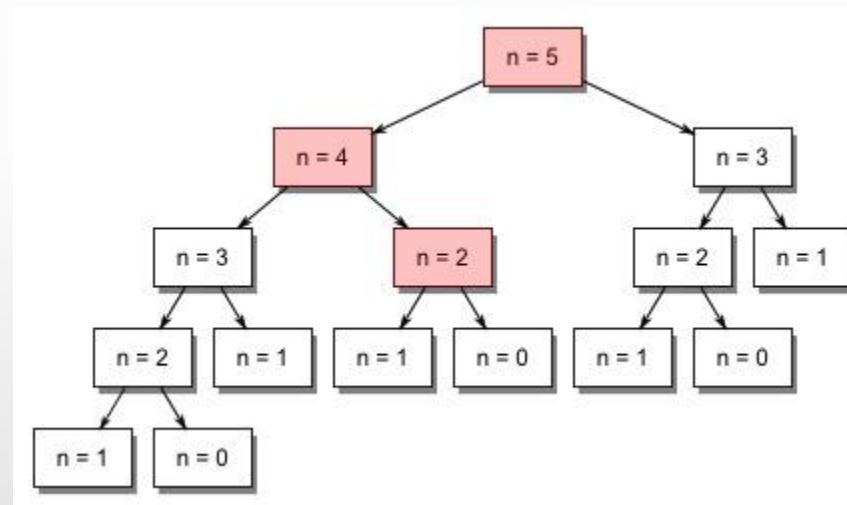


Fibonacci



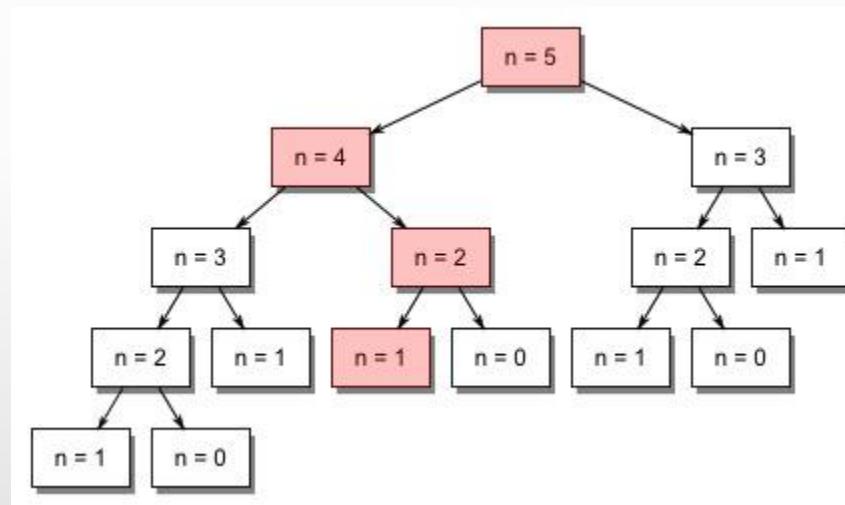


Fibonacci



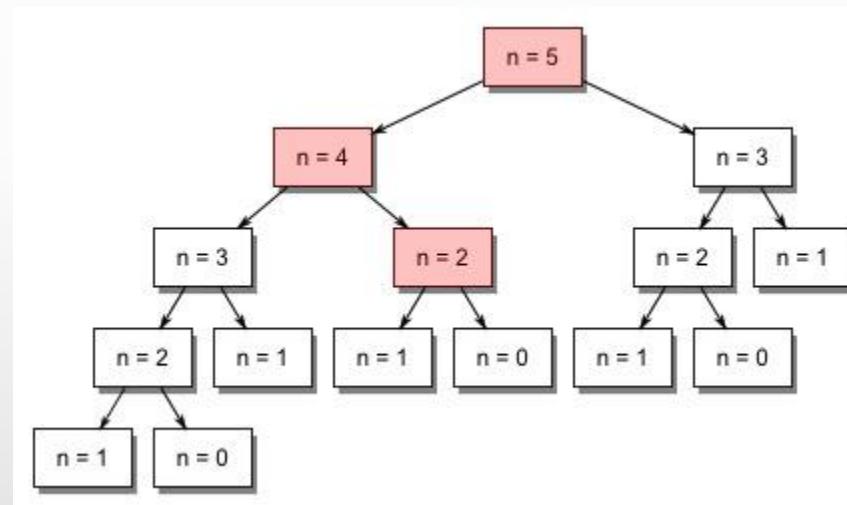


Fibonacci



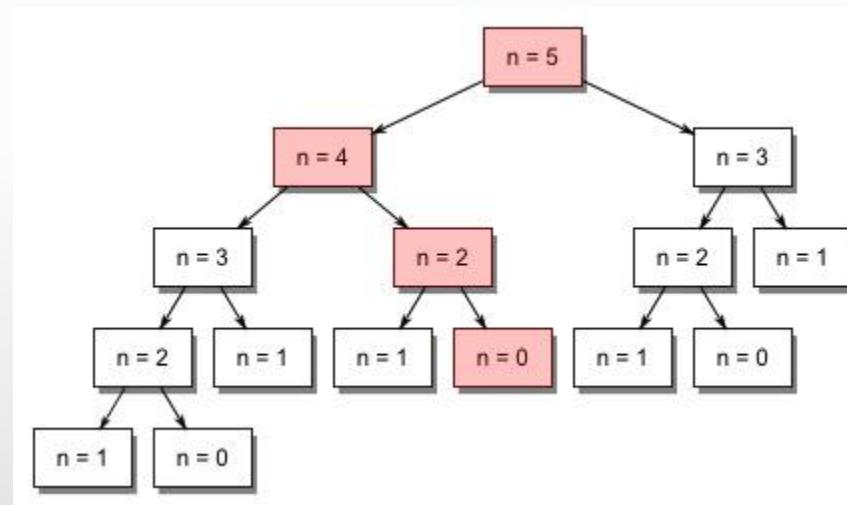


Fibonacci



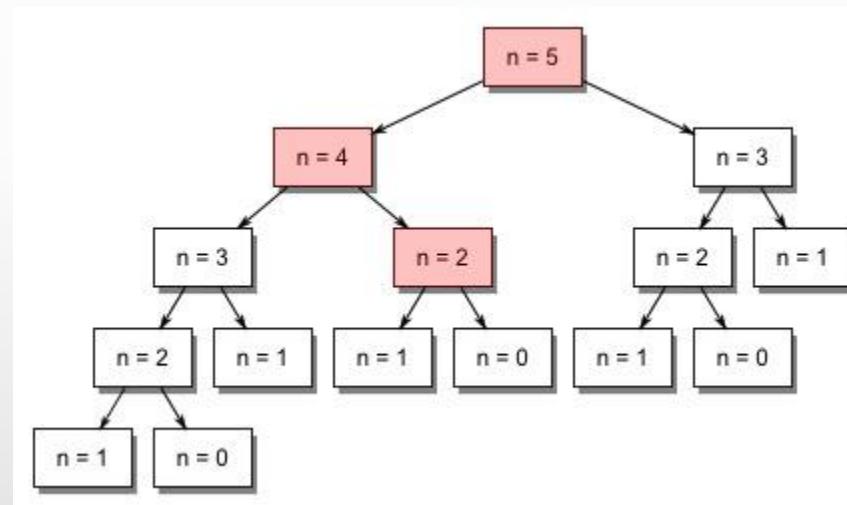


Fibonacci



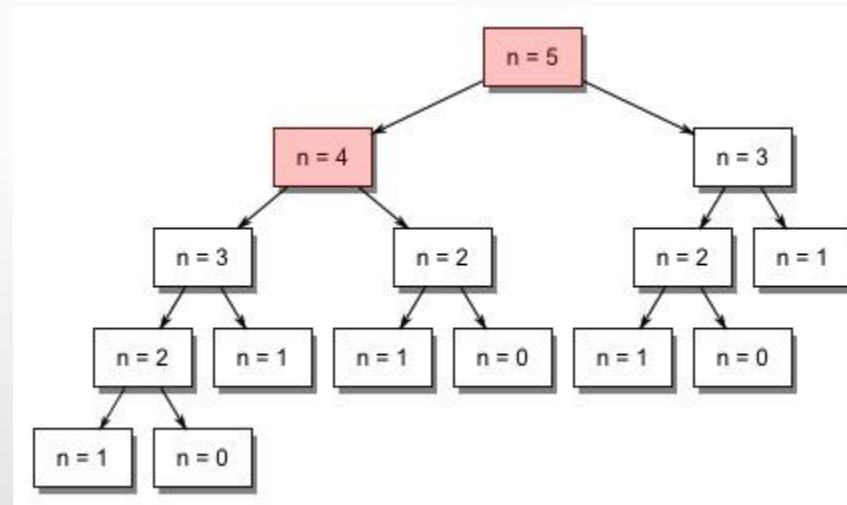


Fibonacci



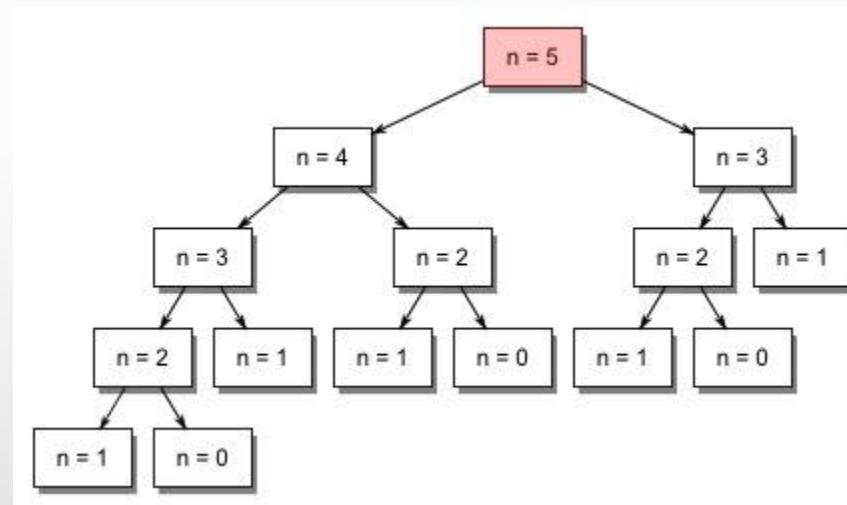


Fibonacci



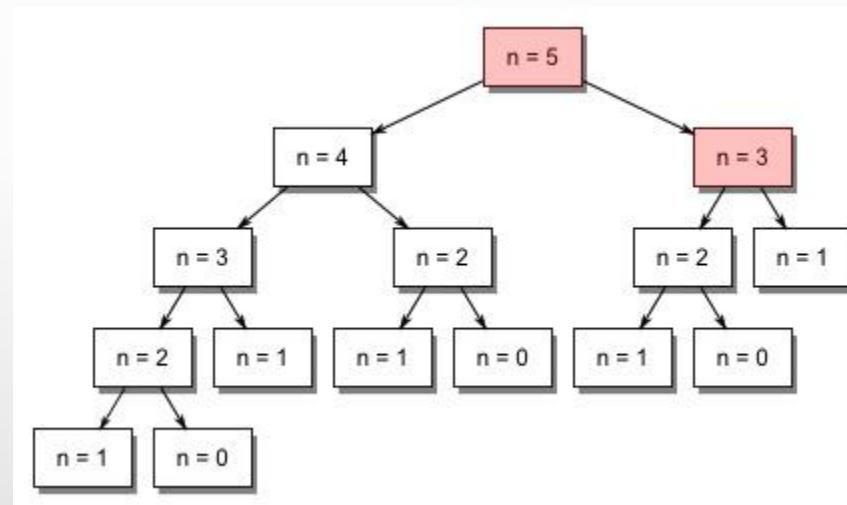


Fibonacci



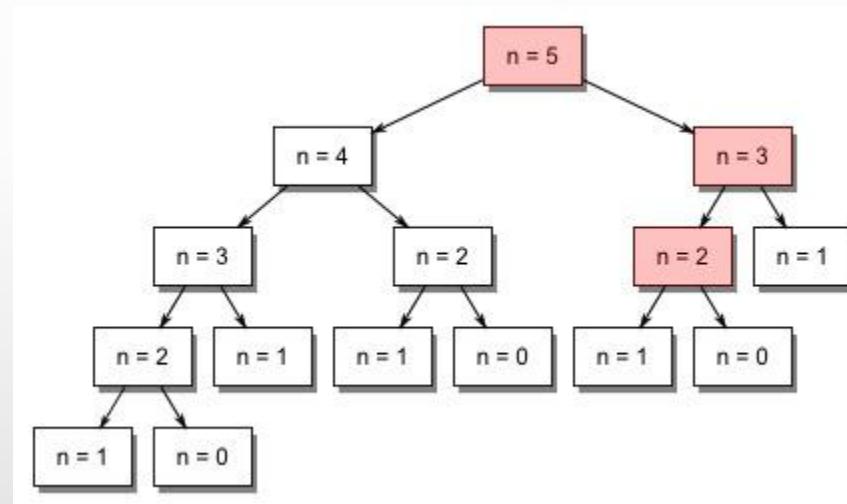


Fibonacci



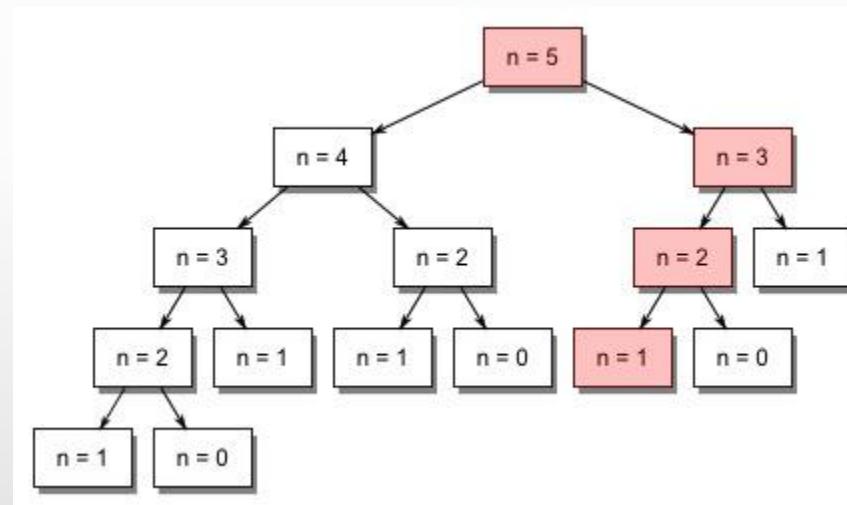


Fibonacci



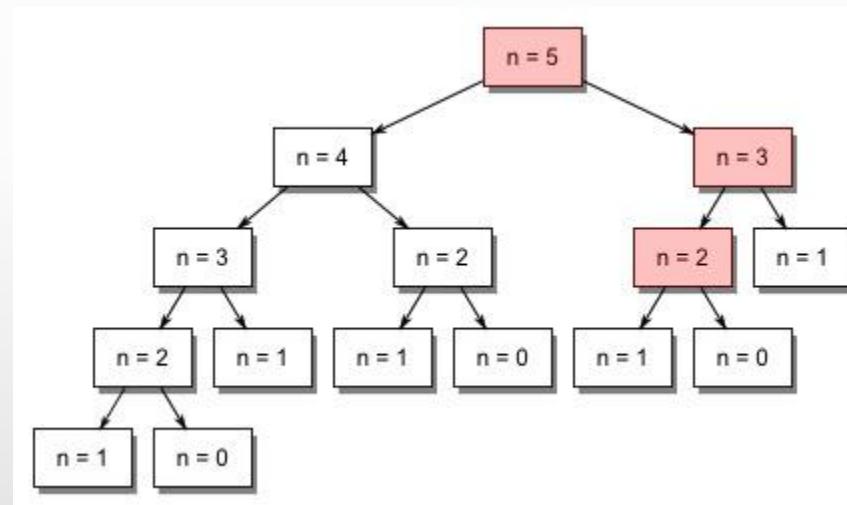


Fibonacci



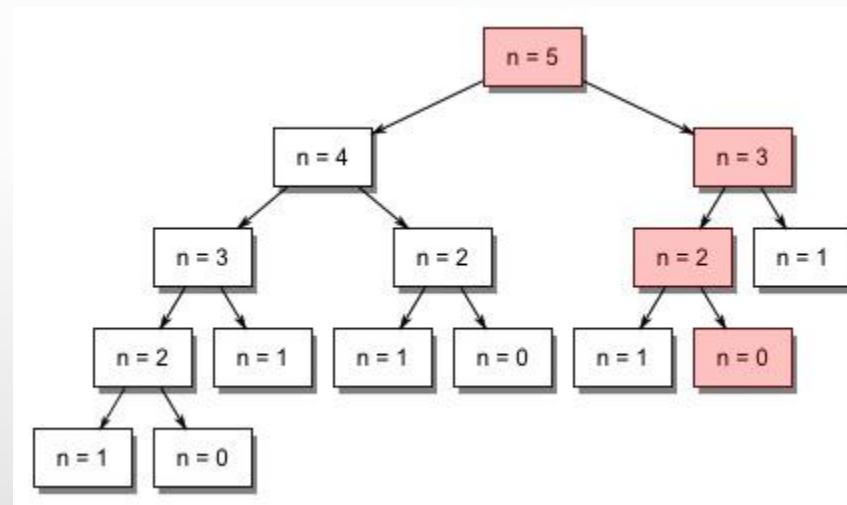


Fibonacci



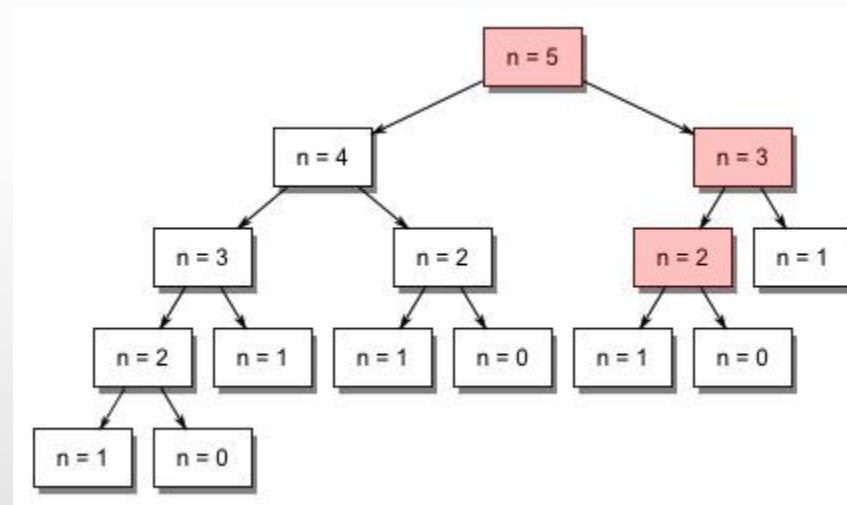


Fibonacci



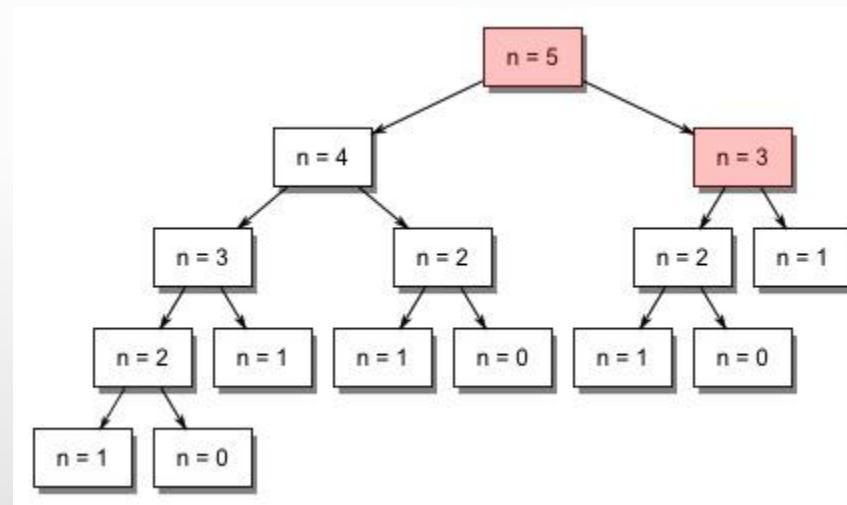


Fibonacci



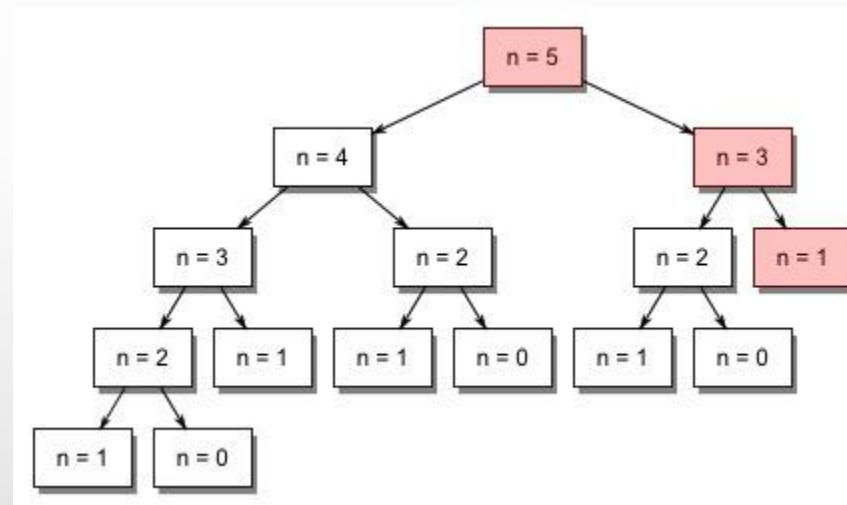


Fibonacci



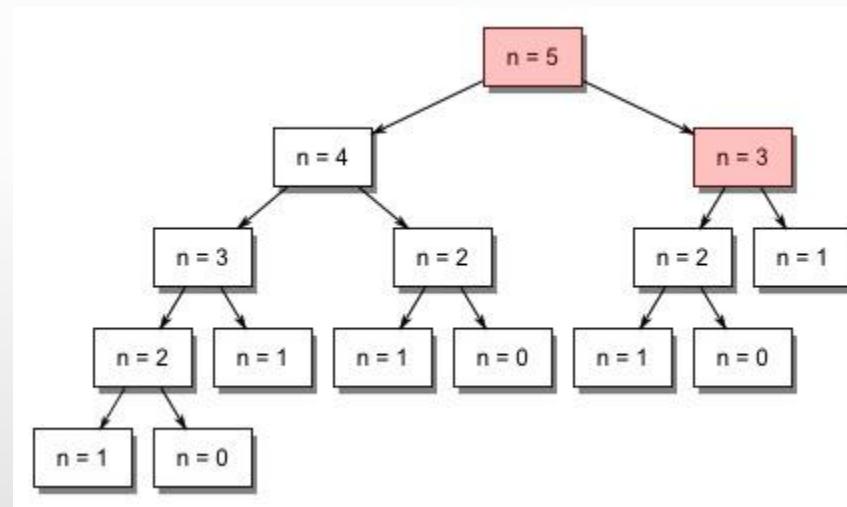


Fibonacci



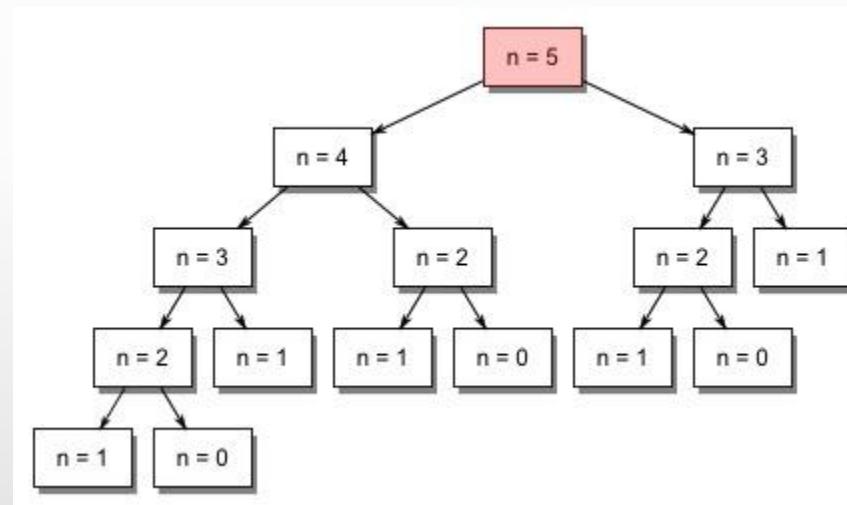


Fibonacci





Fibonacci





Longest Increasing Subsequence



0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 8



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 8



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 4

~~0, 8~~ Inactive



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 4



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 4

0, 4, 12



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 4

0, 4, 12



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 2

~~0, 4~~ Inactive

0, 4, 12



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 2

0, 4, 12



Longest Increasing Subsequence

0, 8, 4, 12, 2, **10**, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 2

0, 2, **10**

~~0, 4, 12~~ Inactive



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 2

0, 2, 10



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 2

0, 2, 6

~~0, 2, 10~~ Inactive



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 2

0, 2, 6



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, **14**, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 2

0, 2, 6

0, 2, 6, **14**



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 2

0, 2, 6

0, 2, 6, 14



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 1

~~0, 2~~ Inactive

0, 2, 6

0, 2, 6, 14



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, **9**, 5, 13, 3, 11, 7, 15

0

0, 1

0, 2, 6

0, 2, 6, 14



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, **9**, 5, 13, 3, 11, 7, 15

0

0, 1

0, 2, 6

0, 2, 6, **9**

~~0, 2, 6, 14~~ Inactive



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 1

0, 2, 6

0, 2, 6, 9



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 1

0, 2, 5

~~0, 2, 6~~ Inactive

0, 2, 6, 9



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, **13**, 3, 11, 7, 15

0

0, 1

0, 2, 5

0, 2, 6, 9



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, **13**, 3, 11, 7, 15

0

0, 1

0, 2, 5

0, 2, 6, 9

0, 2, 6, 9, **13**



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 1

0, 2, 5

0, 2, 6, 9

0, 2, 6, 9, 13



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, **3**, 11, 7, 15

0

0, 1

0, 1, **3**

~~0, 2, 5~~ Inactive

0, 2, 6, 9

0, 2, 6, 9, 13



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, **11**, 7, 15

0

0, 1

0, 1, 3

0, 2, 6, 9

0, 2, 6, 9, 13



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, **11**, 7, 15

0

0, 1

0, 1, 3

0, 2, 6, 9

0, 2, 6, 9, **11**

~~0, 2, 6, 9, 13~~ Inactive



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, **7**, 15

0

0, 1

0, 1, 3

0, 2, 6, 9

0, 2, 6, 9, 11



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 1

0, 1, 3

0, 1, 3, 7

~~0, 2, 6, 9~~ Inactive

0, 2, 6, 9, 11



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, **15**

0

0, 1

0, 1, 3

0, 1, 3, 7

0, 2, 6, 9, 11



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, **15**

0

0, 1

0, 1, 3

0, 1, 3, 7

0, 2, 6, 9, 11

0, 2, 6, 9, 11, **15**



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 1

0, 1, 3

0, 1, 3, 7

0, 2, 6, 9, 11

0, 2, 6, 9, 11, 15



Longest Increasing Subsequence

0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15

0

0, 1

0, 1, 3

0, 1, 3, 7

0, 2, 6, 9, 11

0, 2, 6, 9, 11, 15 Longest Increasing subseq





Longest Common Substring

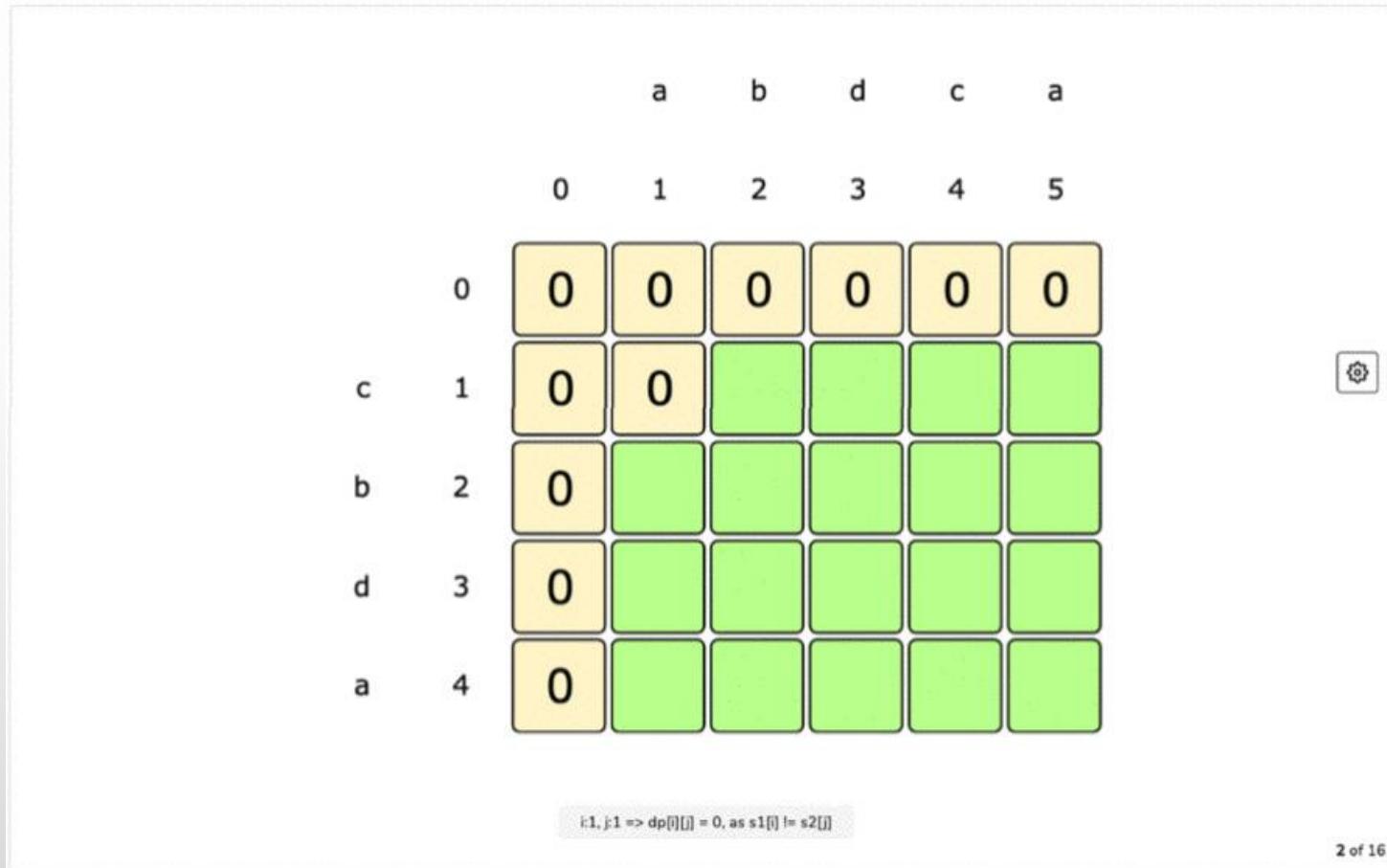
		a	b	d	c	a	
		0	1	2	3	4	5
0		0	0	0	0	0	0
c		0	0	0	0	0	0
b		0	0	0	0	0	0
d		0	0	0	0	0	0
a		0	0	0	0	0	0

i:0, j:0-5 and i:0-4, j:0 => $dp[i][j] = 0$, as we don't have any common substring when one of the string is of zero length

1 of 16

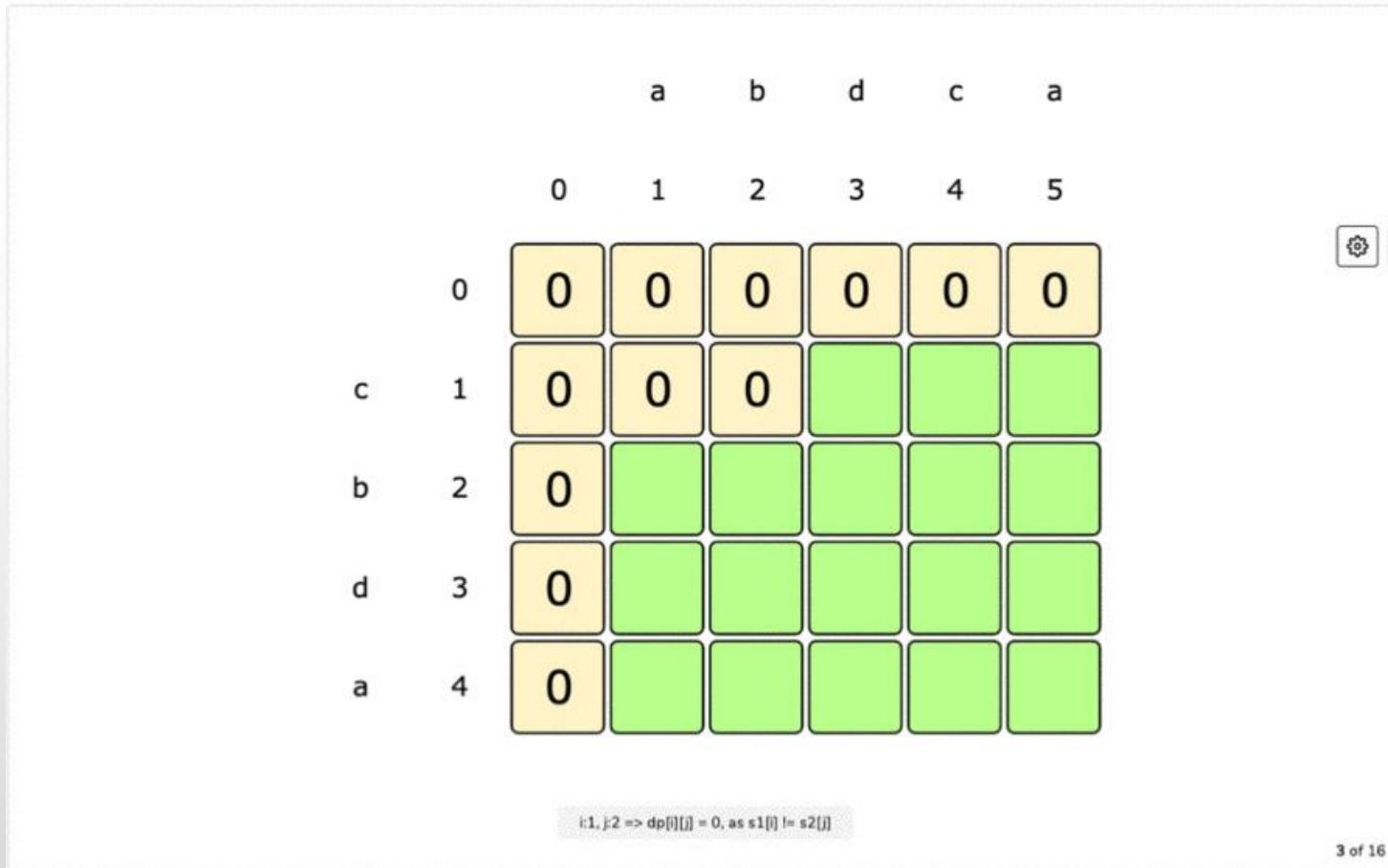


Longest Common Substring



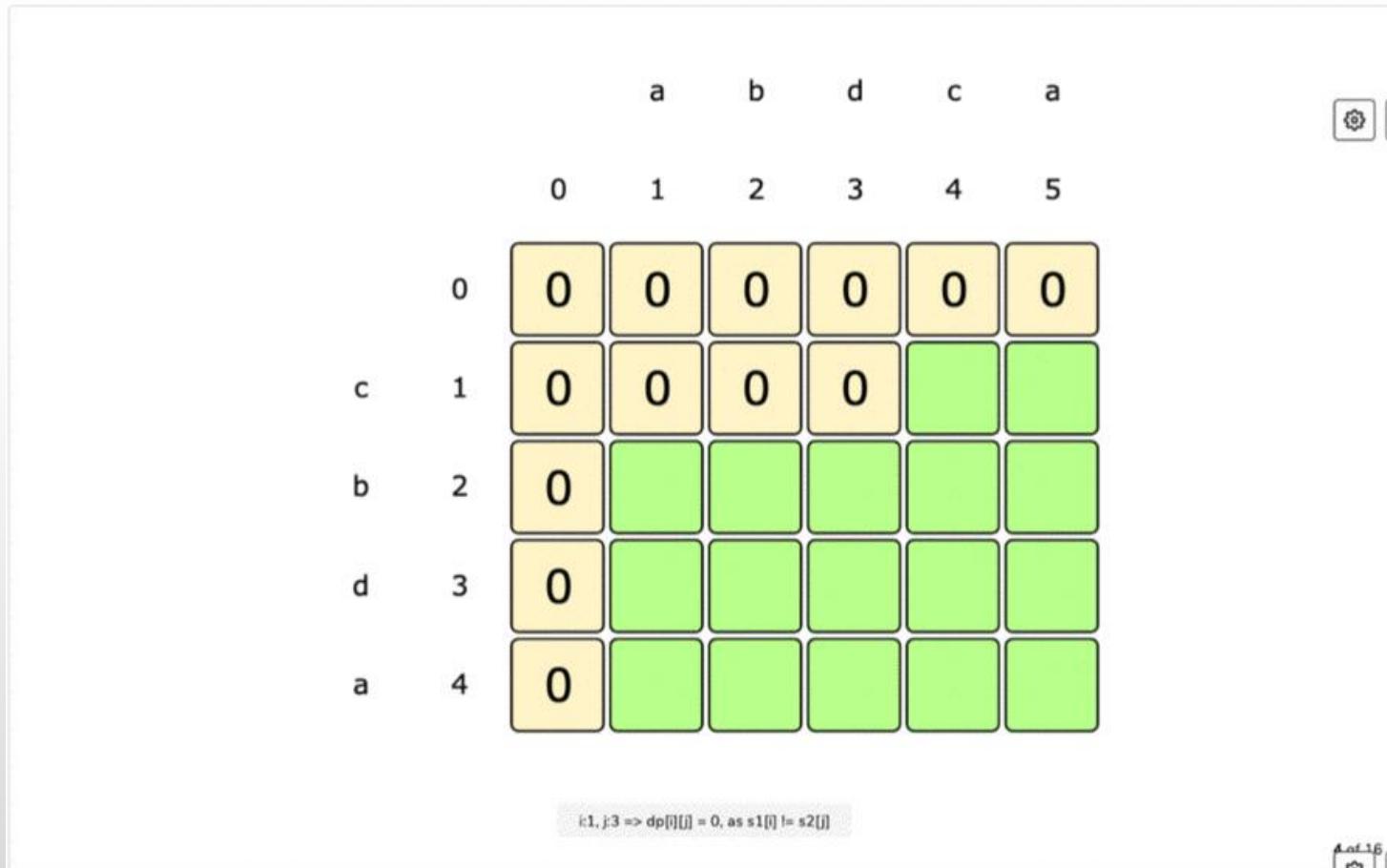


Longest Common Substring



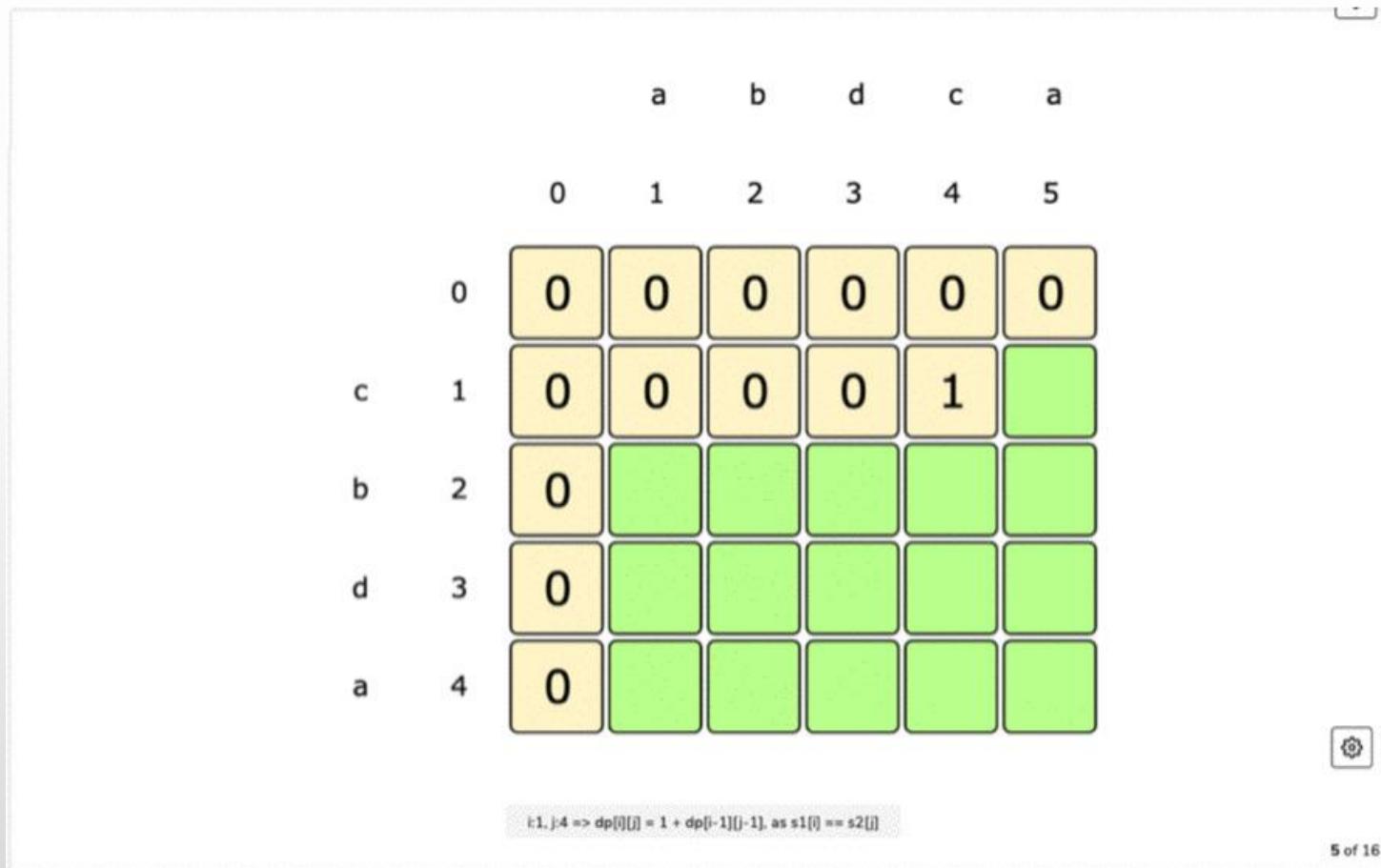


Longest Common Substring



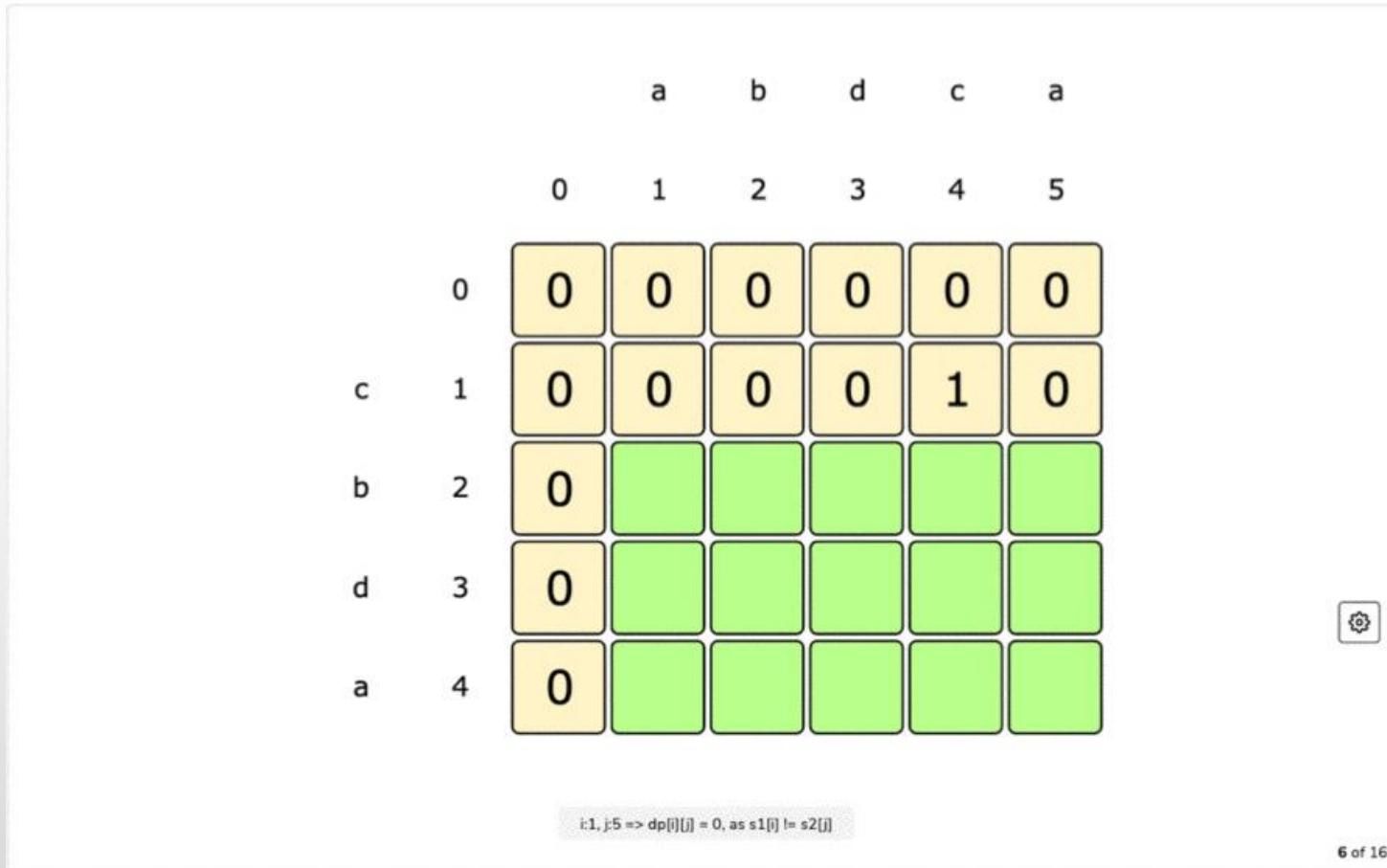


Longest Common Substring



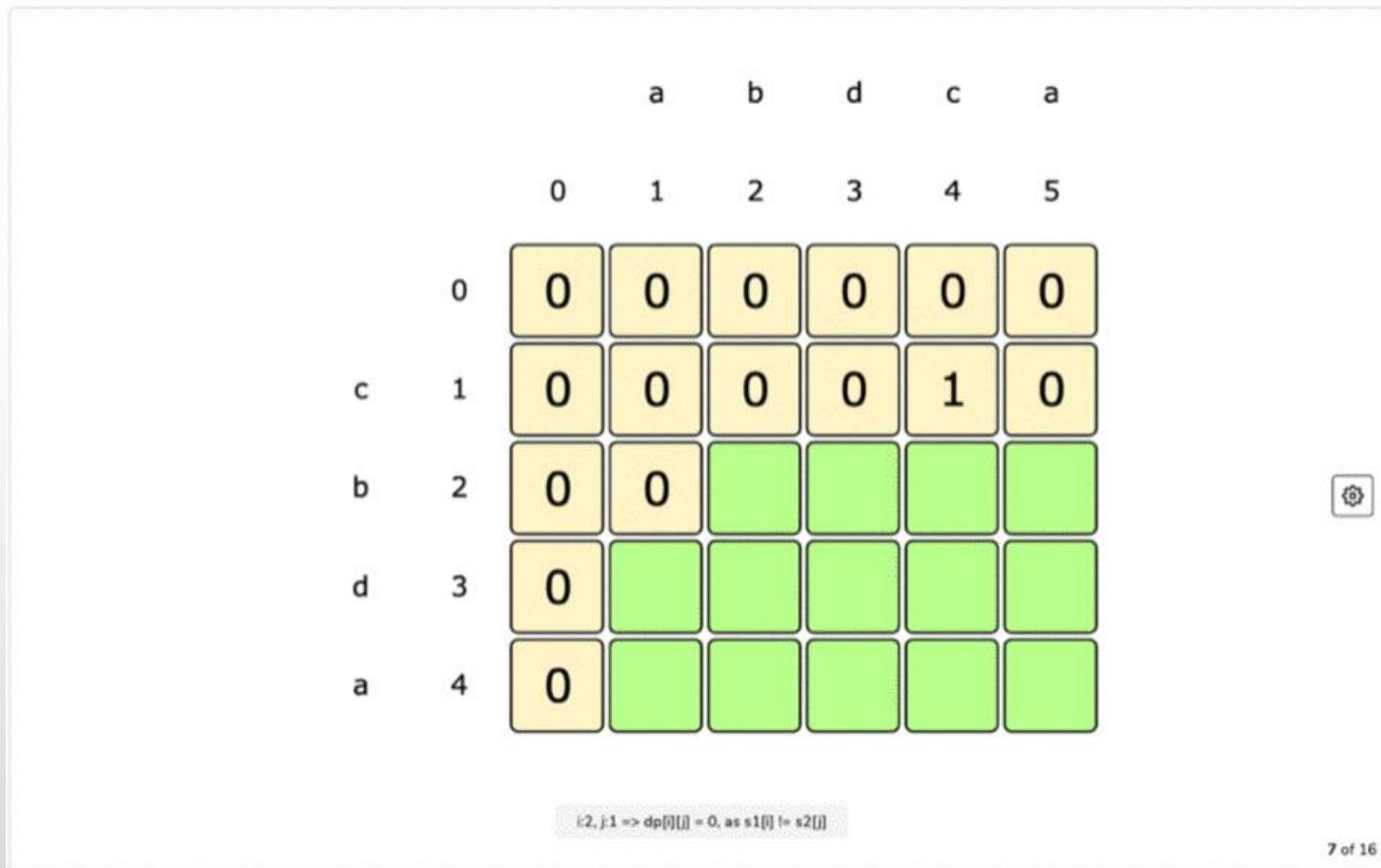


Longest Common Substring



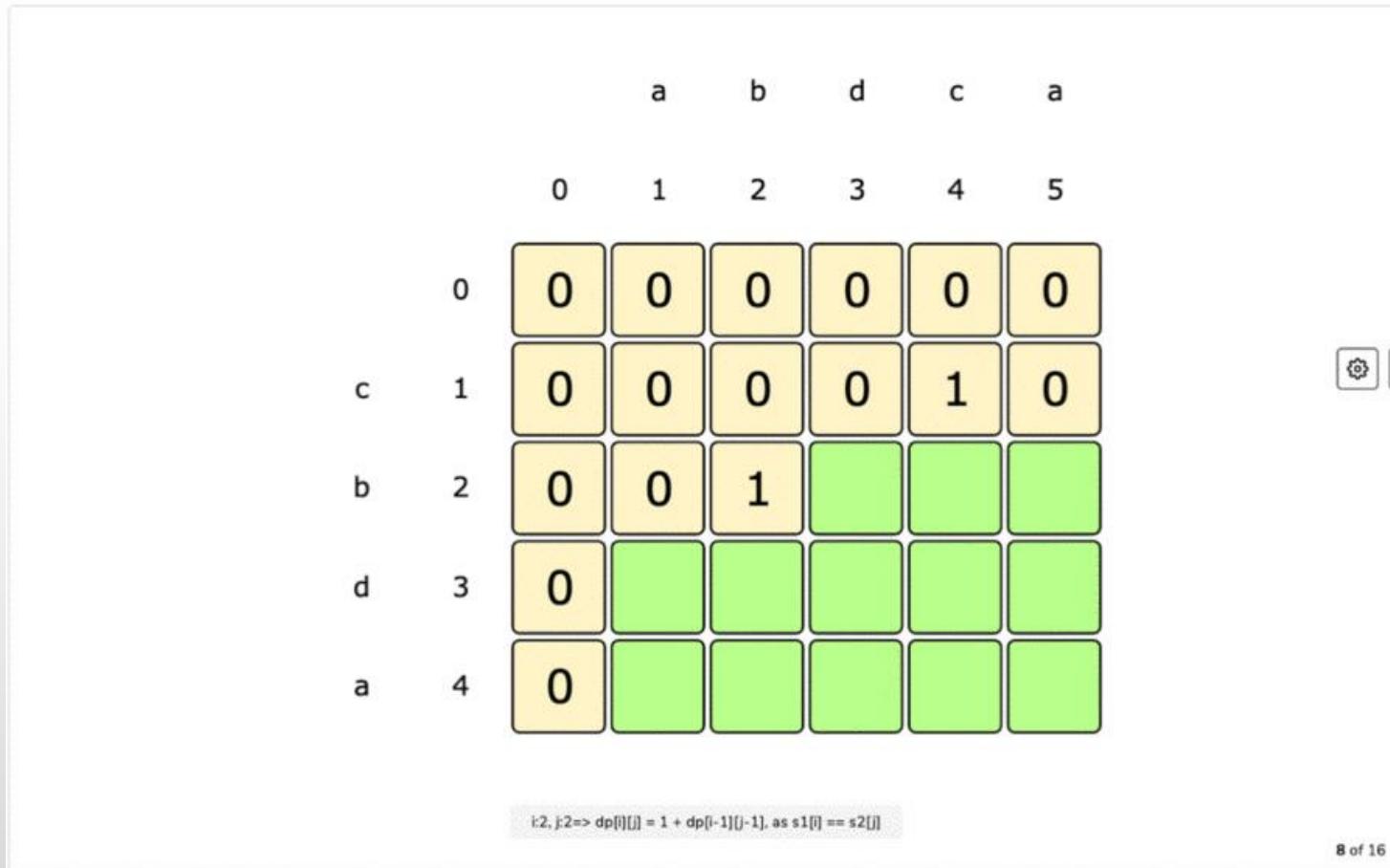


Longest Common Substring



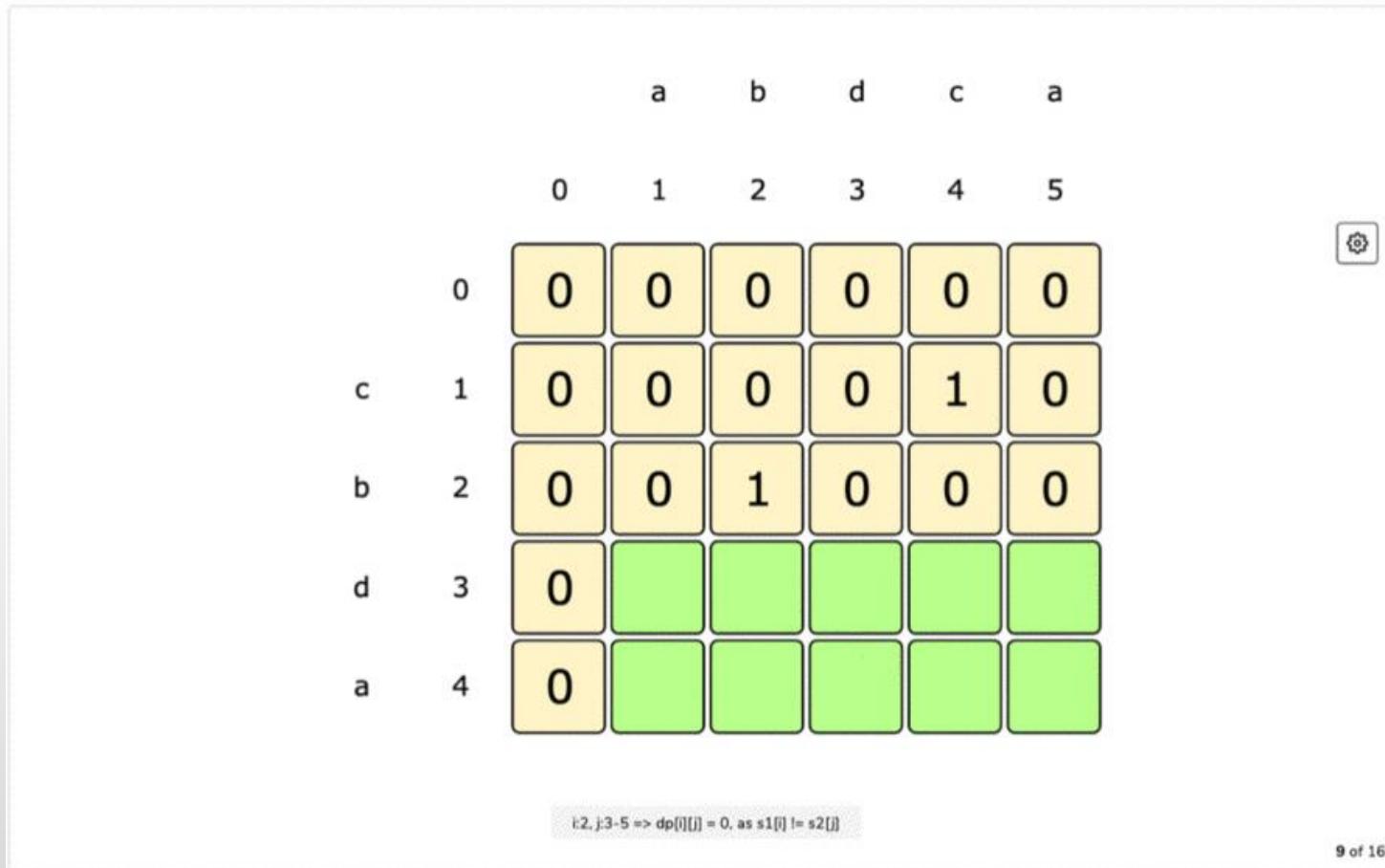


Longest Common Substring



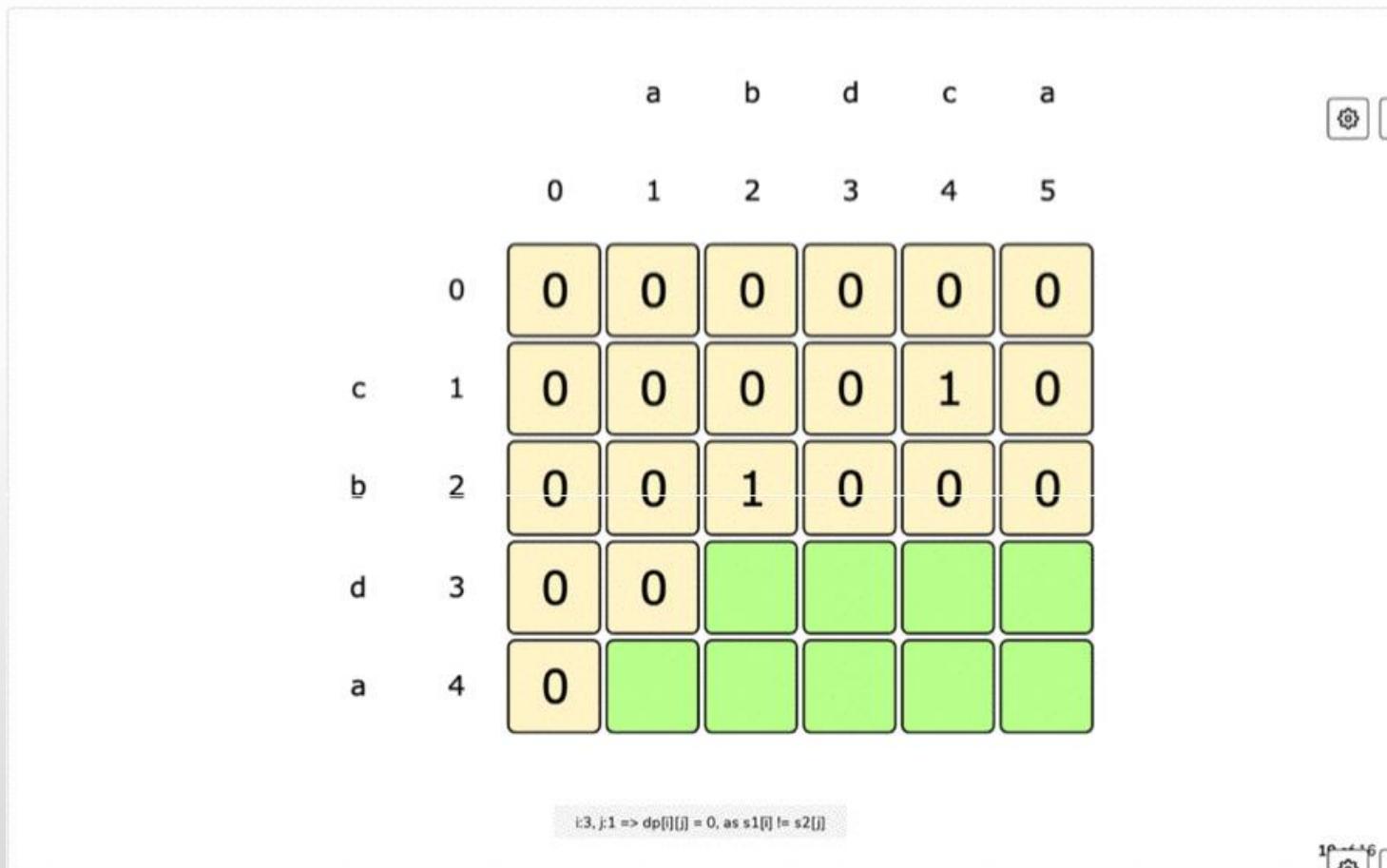


Longest Common Substring



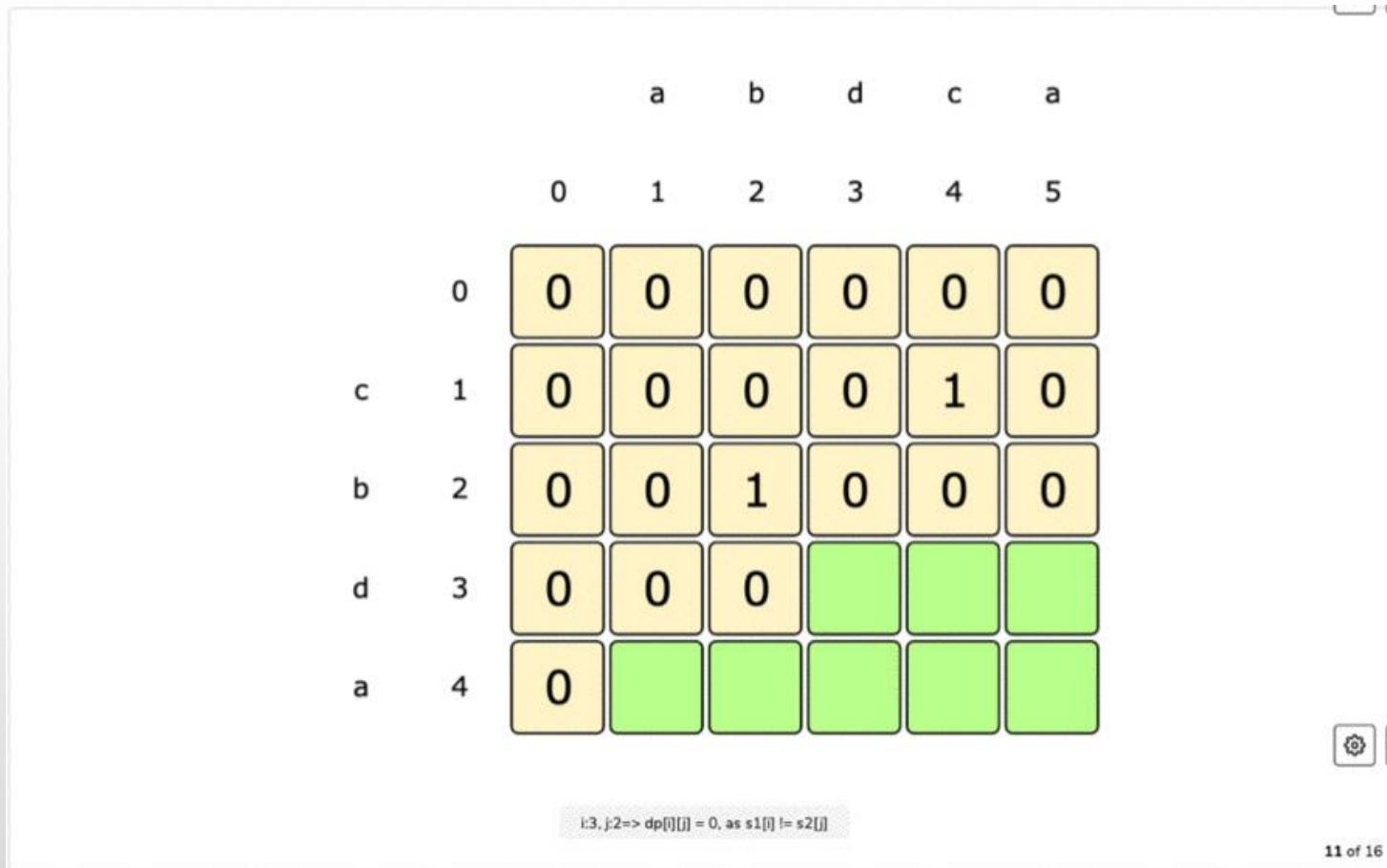


Longest Common Substring



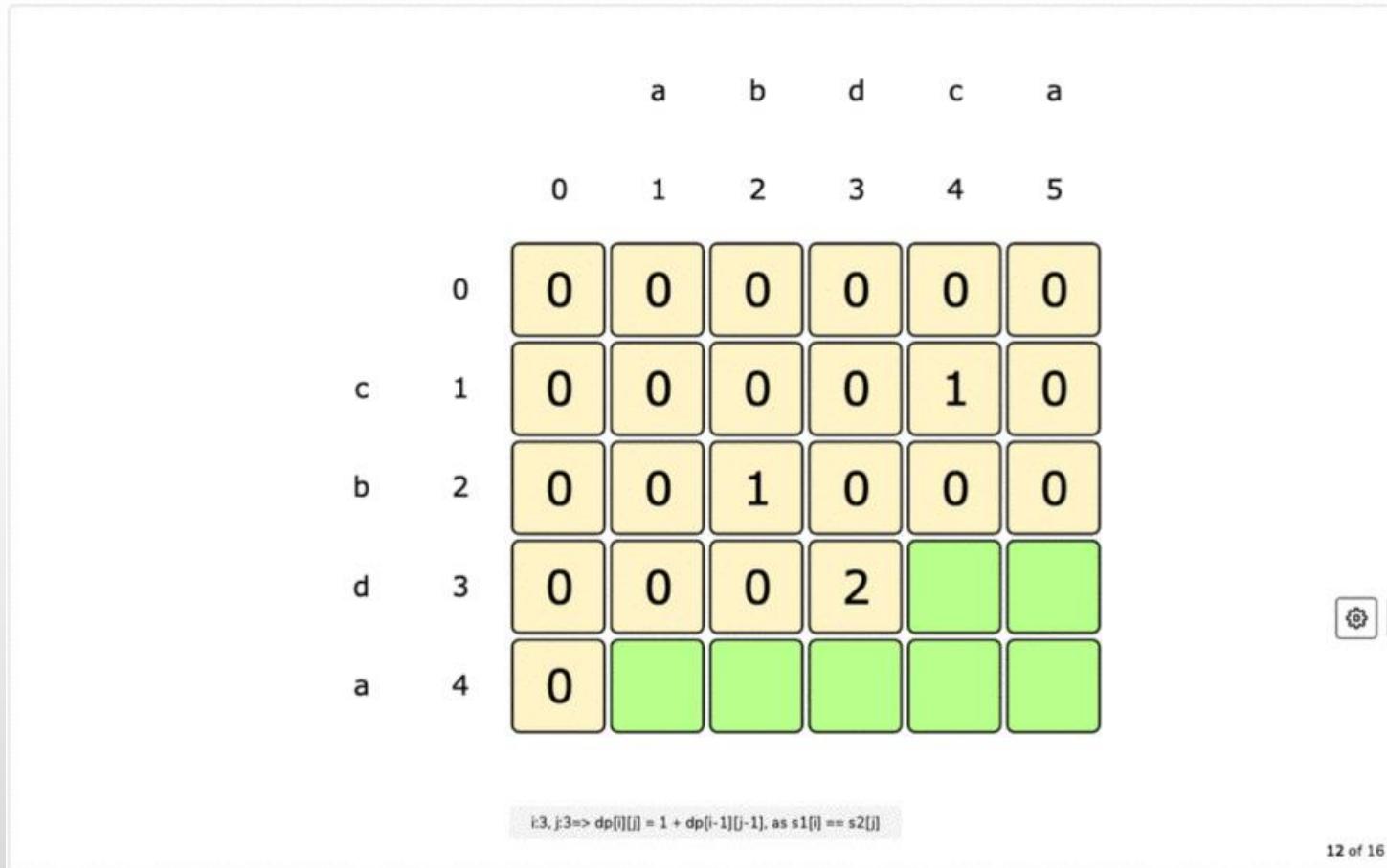


Longest Common Substring



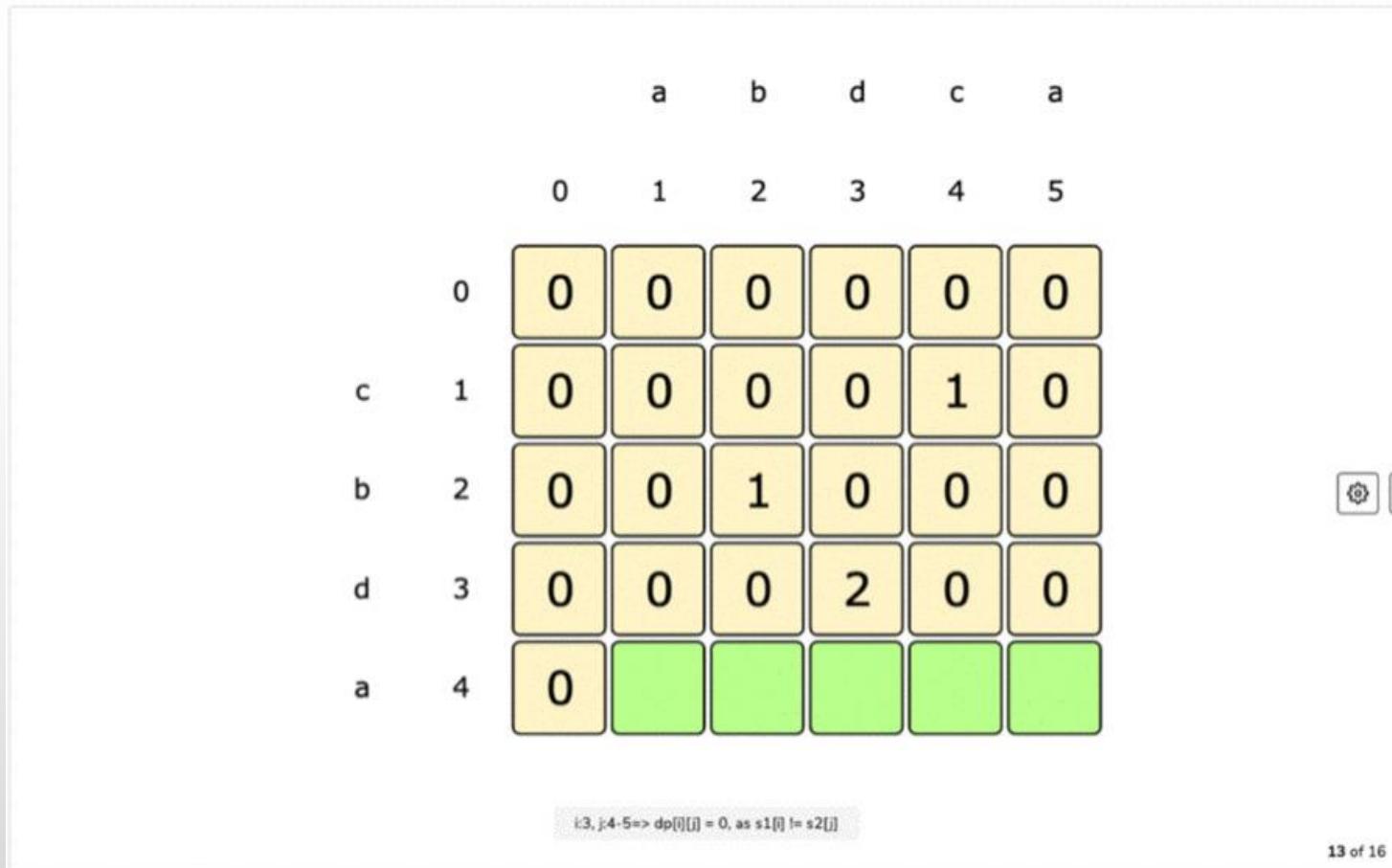


Longest Common Substring



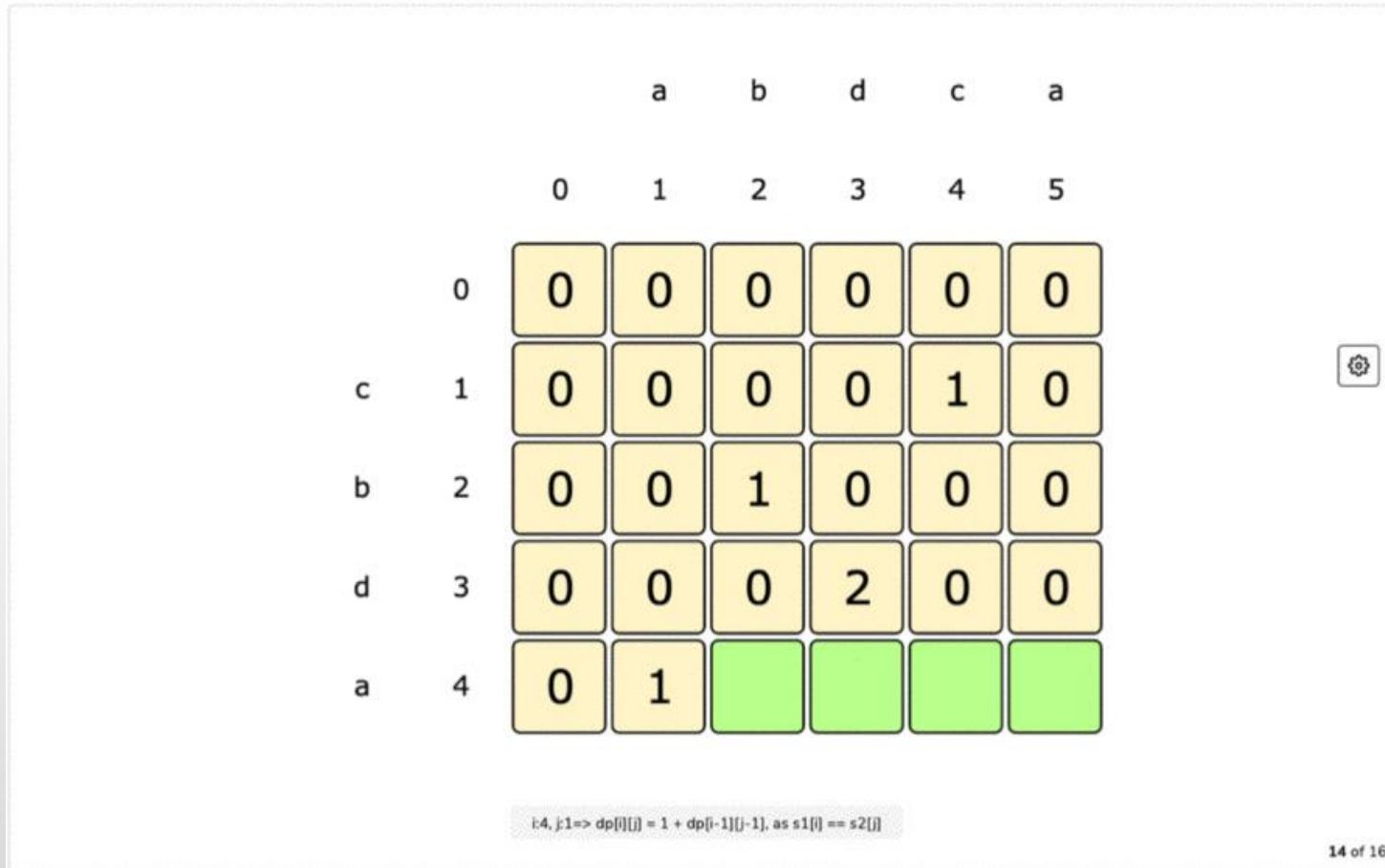


Longest Common Substring





Longest Common Substring





Longest Common Substring

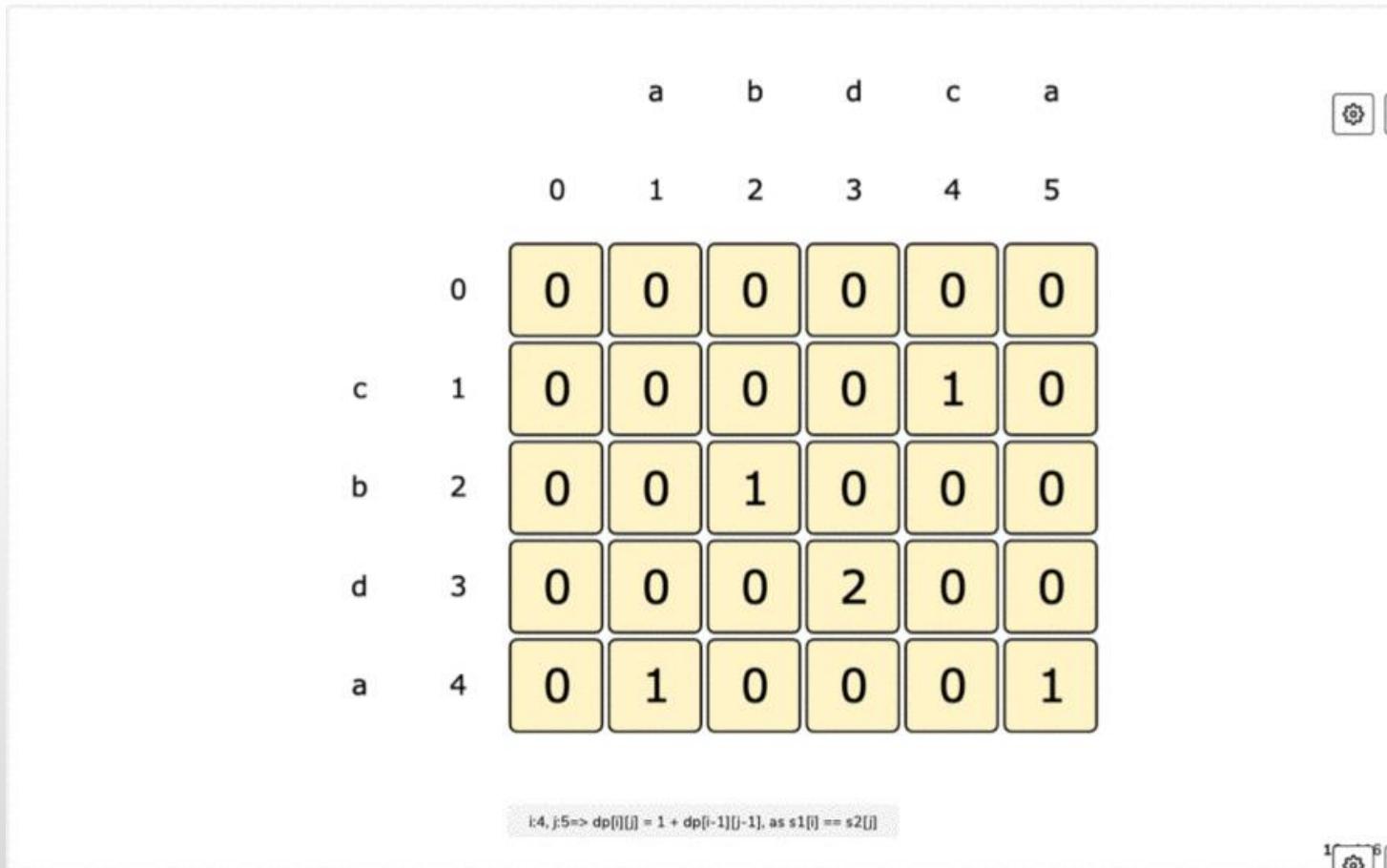
	a	b	d	c	a	
	0	1	2	3	4	5
0	0	0	0	0	0	0
c	1	0	0	0	1	0
b	2	0	0	1	0	0
d	3	0	0	0	2	0
a	4	0	1	0	0	0

i:4, j:2-4=> dp[i][j] = 0, as s1[i] != s2[j]

15 of 16



Longest Common Substring







Longest Common Subsequence

		a	b	d	c	a	
		0	1	2	3	4	5
0		0	0	0	0	0	0
c	1	0					
b	2	0					
d	3	0					
a	4	0					

i:0, j:0-5 and i:0-4, j:0 => dp[i][j] = 0, as we don't have any common subsequence when one of the string is of zero length

1 of 16



Longest Common Subsequence

		a	b	d	c	a	
		0	1	2	3	4	5
0		0	0	0	0	0	0
c		0	0				
b		0					
d		0					
a		0					

i:1, j:1 => $dp[i][j] = \max(dp[i-1][j], dp[i][j-1])$, as $s1[i] \neq s2[j]$

2 of 16



Longest Common Subsequence

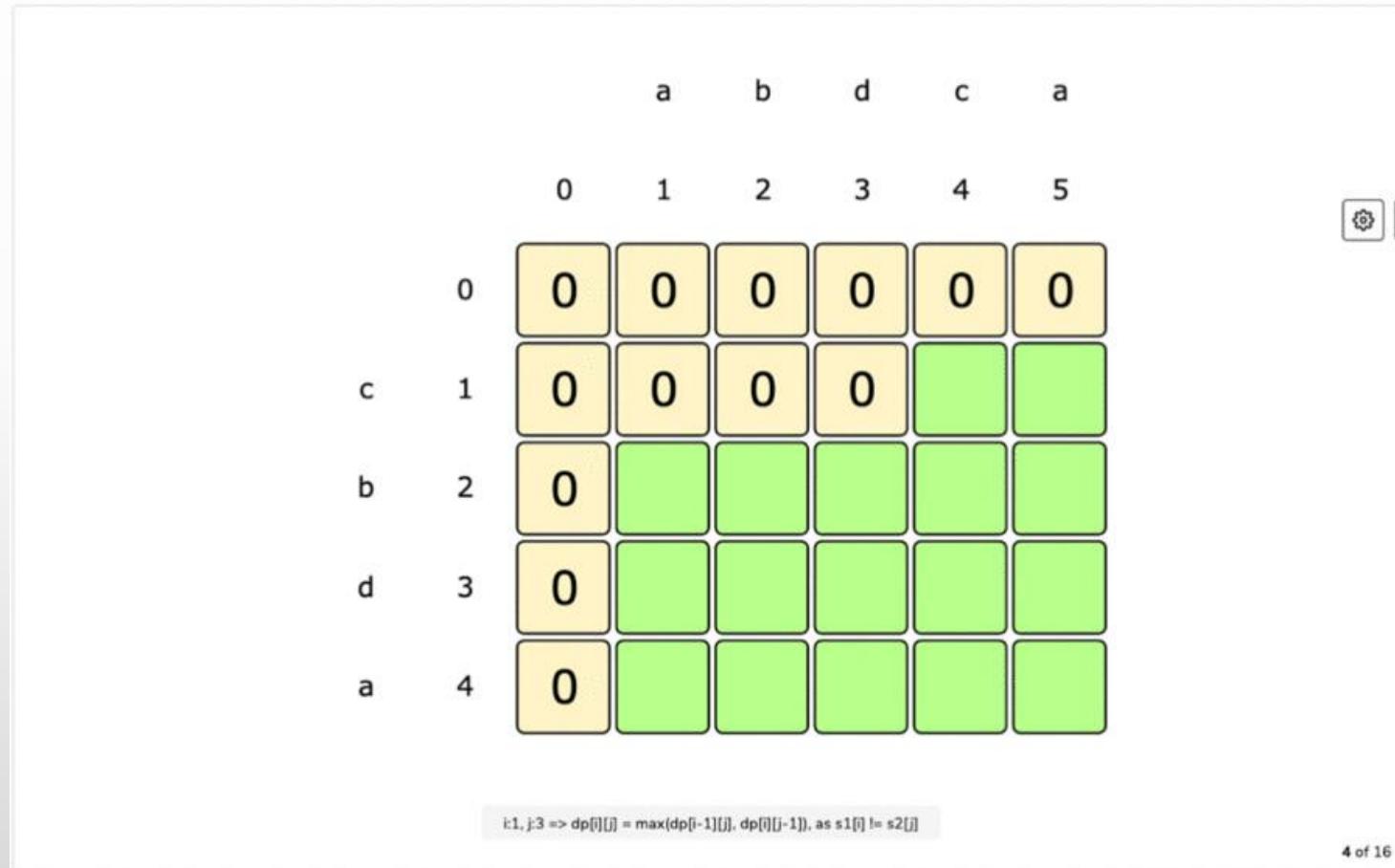
	a	b	d	c	a	
	0	1	2	3	4	5
0	0	0	0	0	0	0
c	1	0	0	0		
b	2	0				
d	3	0				
a	4	0				

$i:1, j:2 \Rightarrow dp[i][j] = \max(dp[i-1][j], dp[i][j-1]), \text{as } s1[i] \neq s2[j]$

3 of 16



Longest Common Subsequence





Longest Common Subsequence

		a	b	d	c	a			
		0	1	2	3	4	5		
0		0	0	0	0	0	0		
c	1	0	0	0	0	1			
b	2	0							
d	3	0							
a	4	0							

i1, j4 => dp[i][j] = 1 + dp[i-1][j-1], as s1[i] == s2[j]



Longest Common Subsequence

		a	b	d	c	a	
		0	1	2	3	4	5
0		0	0	0	0	0	0
c	1	0	0	0	0	1	1
b	2	0					
d	3	0					
a	4	0					

i:1, j:5 => $dp[i][j] = \max(dp[i-1][j], dp[i][j-1])$, as $s1[i] \neq s2[j]$

6 of 16



Longest Common Subsequence

	a	b	d	c	a	
	0	1	2	3	4	5
0	0	0	0	0	0	0
c	1	0	0	0	1	1
b	2	0	0			
d	3	0				
a	4	0				

i:2, j:1 => $dp[i][j] = \max(dp[i-1][j], dp[i][j-1])$, as $s1[i] \neq s2[j]$

7 of 16



Longest Common Subsequence

	a	b	d	c	a	
	0	1	2	3	4	5
0	0	0	0	0	0	0
c	1	0	0	0	1	1
b	2	0	0	1		
d	3	0				
a	4	0				

i:2, j:2=> $dp[i][j] = 1 + dp[i-1][j-1]$, as $s1[i] == s2[j]$

8 of 16



Longest Common Subsequence

		a	b	d	c	a	
		0	1	2	3	4	5
0		0	0	0	0	0	0
c		0	0	0	0	1	1
b		0	0	1	1	1	1
d		0					
a		0					

i:2, j:3-5 => $dp[i][j] = \max(dp[i-1][j], dp[i][j-1])$, as $s1[i] \neq s2[j]$

9 of 16



Longest Common Subsequence

	a	b	d	c	a	
	0	1	2	3	4	5
0	0	0	0	0	0	0
c	1	0	0	0	1	1
b	2	0	0	1	1	1
d	3	0	0			
a	4	0				

i:31, j:1 => $dp[i][j] = \max(dp[i-1][j], dp[i][j-1]),$ as $s1[i] \neq s2[j]$

10 of 16



Longest Common Subsequence

	a	b	d	c	a	
	0	1	2	3	4	5
0	0	0	0	0	0	0
c	1	0	0	0	1	1
b	2	0	0	1	1	1
d	3	0	0	1		
a	4	0				

i:3, j:2 => $dp[i][j] = \max(dp[i-1][j], dp[i][j-1])$, as $s1[i] \neq s2[j]$

1 2 3 4 5 6



Longest Common Subsequence

		a	b	d	c	a	
		0	1	2	3	4	5
0		0	0	0	0	0	0
c	1	0	0	0	0	1	1
b	2	0	0	1	1	1	1
d	3	0	0	1	2		
a	4	0					

i:3, j:3 => $dp[i][j] = 1 + dp[i-1][j-1]$, as $s1[i] == s2[j]$

12 of 16



Longest Common Subsequence

		a	b	d	c	a	
		0	1	2	3	4	5
0		0	0	0	0	0	0
c	1	0	0	0	0	1	1
b	2	0	0	1	1	1	1
d	3	0	0	1	2	2	2
a	4	0					

i:3, j:4-5 => $dp[i][j] = \max(dp[i-1][j], dp[i][j-1])$, as $s1[i] \neq s2[j]$

13 of 16



Longest Common Subsequence

		a	b	d	c	a	
		0	1	2	3	4	5
0		0	0	0	0	0	0
c	1	0	0	0	0	1	1
b	2	0	0	1	1	1	1
d	3	0	0	1	2	2	2
a	4	0	1				

i:4, j:1 => $dp[i][j] = 1 + dp[i-1][j-1]$, as $s1[i] == s2[j]$

14 of 16



Longest Common Subsequence

		a	b	d	c	a	
		0	1	2	3	4	5
0		0	0	0	0	0	0
c	1	0	0	0	0	1	1
b	2	0	0	1	1	1	1
d	3	0	0	1	2	2	2
a	4	0	1	1	2	2	2

i:4, j:2-4 => $dp[i][j] = \max(dp[i-1][j], dp[i][j-1])$, as $s1[i] \neq s2[j]$

15 of 16



Longest Common Subsequence

	a	b	d	c	a	
	0	1	2	3	4	5
0	0	0	0	0	0	0
c	1	0	0	0	1	1
b	2	0	0	1	1	1
d	3	0	0	1	2	2
a	4	0	1	1	2	3

i:4, j:5=> $dp[i][j] = 1 + dp[i-1][j-1]$, as $s1[i] == s2[j]$

16 of 16



SON