

COL 351 : ANALYSIS & DESIGN OF ALGORITHMS

LECTURE 22

DYNAMIC PROGRAMMING III: KNAPSACK (CONTD.)

SEPT 23, 2026

|

ROHIT VAISH

KNAPSACK

KNAPSACK

input : ① n items , each having a

- value v_i (non negative)

- size s_i (non negative and integral)

② capacity C (non negative and integral)

KNAPSACK

- input :
- ① n items , each having a
 - value v_i (non negative)
 - size s_i (non negative and integral)
 - ② capacity C (non negative and integral)

output : a subset $S \subseteq \{1, 2, \dots, n\}$

that maximizes $\sum_{i \in S} v_i$

subject to $\sum_{i \in S} s_i \leq C$

OPTIMAL SUBSTRUCTURE

Let $S :=$ a max value solution of given knapsack instance

Case I : item $n \notin S$

$\Rightarrow S$ must be optimal for the first $(n-1)$ items
and residual capacity C .

Case II : item $n \in S$

$\Rightarrow S \setminus \{n\}$ must be optimal for the first $(n-1)$ items
and residual capacity $C - s_n$.

buffer for item n 

THE SUBPROBLEMS

Subproblems along **two** dimensions

* all possible prefixes of items $\{1, 2, \dots, i\}$

* all possible (integral) residue capacities $x \in \{0, 1, \dots, C\}$

recall C and all
sizes s_i are integral

THE SUBPROBLEMS

Subproblems along **two** dimensions

* all possible prefixes of items $\{1, 2, \dots, i\}$

* all possible (integral) residue capacities $x \in \{0, 1, \dots, C\}$

recall C and all
sizes s_i are integral

Design a recurrence to systematically solve all subproblems.

KNAPSACK RECURRENCE

Define $V_{i,n} :=$ value of best solution that

(a) uses only the first i items, and

(b) has total size $\leq n$

KNAPSACK RECURRENCE

Define $V_{i,n} :=$ value of best solution that

(a) uses only the first i items, and

(b) has total size $\leq n$

For every $i \in \{1, 2, \dots, n\}$ and every $n \in \{0, 1, 2, \dots, C\}$

$$V_{i,n} = \max \left\{ V_{(i-1), n} , V_{(i-1), n-s_i} + v_i \right\}$$

excluding item i

including item i

KNAPSACK RECURRENCE

Define $V_{i,n} :=$ value of best solution that

(a) uses only the first i items, and

(b) has total size $\leq n$

For every $i \in \{1, 2, \dots, n\}$ and every $n \in \{0, 1, 2, \dots, C\}$

$$\begin{aligned} V_{i,n} &= \max \left\{ V_{(i-1), n} , V_{(i-1), n-s_i} + v_i \right\} \text{ if } s_i \leq n \\ &= V_{(i-1), n} \text{ if } s_i > n. \end{aligned}$$

THE ALGORITHM

THE ALGORITHM

input: item values $v_1, \dots, v_n$, item sizes $s_1, \dots, s_n$, capacity C

output: maximum total value of a subset $S \subseteq \{1, 2, \dots, n\}$ with $\sum_{i \in S} s_i \leq C$.

THE ALGORITHM

input: item values $v_1, \dots, v_n$, item sizes $s_1, \dots, s_n$, capacity C

output: maximum total value of a subset $S \subseteq \{1, 2, \dots, n\}$ with $\sum_{i \in S} s_i \leq C$.

// subproblem solutions

$A := (n+1) \times (C+1)$ two-dimensional array

THE ALGORITHM

input : item values $v_1, \dots, v_n$, item sizes $s_1, \dots, s_n$, capacity C

output : maximum total value of a subset $S \subseteq \{1, 2, \dots, n\}$ with $\sum_{i \in S} s_i \leq C$.

// subproblem solutions

$A := (n+1) \times (C+1)$ two-dimensional array

$C+1$

$n+1$

THE ALGORITHM

input : item values $v_1, \dots, v_n$, item sizes $s_1, \dots, s_n$, capacity C

output : maximum total value of a subset $S \subseteq \{1, 2, \dots, n\}$ with $\sum_{i \in S} s_i \leq C$.

// subproblem solutions

$A := (n+1) \times (C+1)$ two-dimensional array

// base case

$A[0, n] = 0$ for $n \in \{0, 1, \dots, C\}$

						$C+1$
$n+1$						

THE ALGORITHM

input : item values $v_1, \dots, v_n$, item sizes $s_1, \dots, s_n$, capacity C

output : maximum total value of a subset $S \subseteq \{1, 2, \dots, n\}$ with $\sum_{i \in S} s_i \leq C$.

// subproblem solutions

$A := (n+1) \times (C+1)$ two-dimensional array

// base case

$A[0, n] = 0$ for $n \in \{0, 1, \dots, C\}$

	$C+1$					
	0	0	0	0	0	0
$n+1$						

THE ALGORITHM

input: item values $v_1, \dots, v_n$, item sizes $s_1, \dots, s_n$, capacity C

output: maximum total value of a subset $S \subseteq \{1, 2, \dots, n\}$ with $\sum_{i \in S} s_i \leq C$.

// subproblem solutions

$A := (n+1) \times (C+1)$ two-dimensional array

// base case

$A[0, n] = 0$ for $n \in \{0, 1, \dots, C\}$

// recurrence

for $i = 1, 2, \dots, n$

 for $x = 0, 1, \dots, C$

$C+1$

0	0	0	0	0	0

$n+1$

THE ALGORITHM

input: item values $v_1, \dots, v_n$, item sizes $s_1, \dots, s_n$, capacity C

output: maximum total value of a subset $S \subseteq \{1, 2, \dots, n\}$ with $\sum_{i \in S} s_i \leq C$.

// subproblem solutions

$A := (n+1) \times (C+1)$ two-dimensional array

// base case

$A[0, n] = 0$ for $n \in \{0, 1, \dots, C\}$

// recurrence

for $i = 1, 2, \dots, n$

 for $x = 0, 1, \dots, C$

$A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$ // $s_i \leq x$

C+1					
0	0	0	0	0	0

n+1

THE ALGORITHM

input: item values $v_1, \dots, v_n$, item sizes $s_1, \dots, s_n$, capacity C

output: maximum total value of a subset $S \subseteq \{1, 2, \dots, n\}$ with $\sum_{i \in S} s_i \leq C$.

// subproblem solutions

$A := (n+1) \times (C+1)$ two-dimensional array

// base case

$A[0, x] = 0$ for $x \in \{0, 1, \dots, C\}$

// recurrence

for $i = 1, 2, \dots, n$

 for $x = 0, 1, \dots, C$

$A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$ // $s_i \leq x$
 or $A[i, x] := A[i-1, x]$ // $s_i > x$

C+1					
0	0	0	0	0	0

n+1

THE ALGORITHM

input: item values $v_1, \dots, v_n$, item sizes $s_1, \dots, s_n$, capacity C

output: maximum total value of a subset $S \subseteq \{1, 2, \dots, n\}$ with $\sum_{i \in S} s_i \leq C$.

// subproblem solutions

$A := (n+1) \times (C+1)$ two-dimensional array

// base case

$A[0, x] = 0$ for $x \in \{0, 1, \dots, C\}$

// recurrence

for $i = 1, 2, \dots, n$

for $x = 0, 1, \dots, C$

$A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$ // $s_i \leq x$
or $A[i, x] := A[i-1, x]$ // $s_i > x$

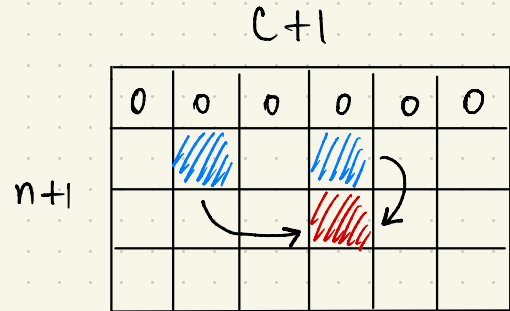
$C+1$

0	0	0	0	0	0
	//		//		
			//		

$n+1$

THE ALGORITHM

Running time : ?

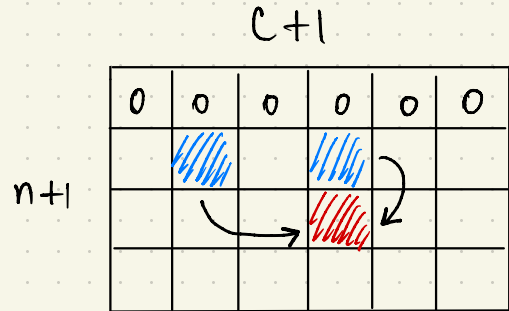


THE ALGORITHM

Running time : $O(n C)$ subproblems

$O(1)$ time per subproblem

$\Rightarrow O(n C)$ time overall



THE ALGORITHM

Running time : $O(nc)$ subproblems

$O(1)$ time per subproblem

$\Rightarrow O(nc)$ time overall

Is this algorithm polynomial time?

THE ALGORITHM

Running time : $O(nc)$ subproblems
 $O(1)$ time per subproblem
 $\Rightarrow O(nc)$ time overall

Is this algorithm polynomial time? NO!

input size : $O(\sum \log v_i + \sum \log s_i + \log c)$

THE ALGORITHM

Running time : $O(nc)$ subproblems
 $O(1)$ time per subproblem
 $\Rightarrow O(nc)$ time overall

Is this algorithm polynomial time? NO!

input size : $O(\sum \log v_i + \sum \log s_i + \log C)$

Pseudopolynomial time

THE ALGORITHM

Correctness : By induction (Exercise)

Reconstruction : By backtracking (Exercise)

EXAMPLE

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

capacity = 6

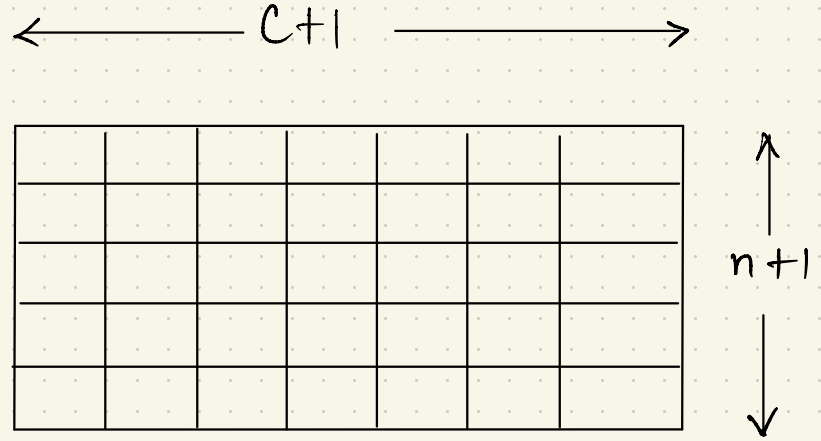
EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

capacity = 6

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3



capacity = 6

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

$\xleftarrow{C+1} \xrightarrow{\hspace{1cm}}$						
$n=0$	1	2	3	4	5	6
$i=0$						
1						
2						
3						
4						

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$							
	$n=0$	1	2	3	4	5	6
$i=0$							
1							
2							
3							
4							

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

// base case

$A[0, n] = 0$ for $n \in \{0, 1, \dots, C\}$

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← $C+1$ →						
	$n=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1							
2							
3							
4							

↑
 $n+1$
↓

capacity = 6

// base case

$A[0, n] = 0$ for $n \in \{0, 1, \dots, C\}$

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

$\xleftarrow{C+1} \xrightarrow{\hspace{1cm}}$						
$n=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0
1						
2						
3						
4						
$\uparrow$ $n+1$ $\downarrow$						

capacity = 6

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	x=0	1	2	3	4	5	6
i=0	0	0	0	0	0	0	0
1							
2							
3							
4							

↑
n+1
↓

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0			
2							
3							
4							

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

```

for  $i = 1, 2, \dots, n$ 
  for  $x = 0, 1, \dots, C$ 
     $A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$  //  $s_i \leq x$ 
    or  $A[i, x] := A[i-1, x]$  //  $s_i > x$ 
    
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	x=0	1	2	3	4	5	6
i=0	0	0	0	0	0	0	0
1	0	0	0	0	?		
2							
3							
4							

↑
n+1
↓

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$

	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0			
2			+3				
3							
4							

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

```

for  $i = 1, 2, \dots, n$ 
  for  $x = 0, 1, \dots, C$ 
     $A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$  //  $s_i \leq x$ 
    or  $A[i, x] := A[i-1, x]$  //  $s_i > x$ 
    
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3		
2							
3							
4							

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

```

for  $i = 1, 2, \dots, n$ 
  for  $x = 0, 1, \dots, C$ 
     $A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$  //  $s_i \leq x$ 
    or  $A[i, x] := A[i-1, x]$  //  $s_i > x$ 
    
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2							
3							
4							

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

```

for  $i = 1, 2, \dots, n$ 
  for  $x = 0, 1, \dots, C$ 
     $A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$  //  $s_i \leq x$ 
    or  $A[i, x] := A[i-1, x]$  //  $s_i > x$ 
    
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2							
3							
4							

↑
n+1
↓

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0				
3							
4							

↑
n+1
↓

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-s_i] + v_i } // s_i ≤ x
    or A[i, x] := A[i-1, x] // s_i > x
  
```


EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$

	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0				
3							
4							

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

```

for  $i = 1, 2, \dots, n$ 
  for  $x = 0, 1, \dots, C$ 
     $A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$  //  $s_i \leq x$ 
    or  $A[i, x] := A[i-1, x]$  //  $s_i > x$ 
    
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2			
3							
4							

↑
n+1
↓

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$

	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2			
3							
4							

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

```

for  $i = 1, 2, \dots, n$ 
  for  $x = 0, 1, \dots, C$ 
     $A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$  //  $s_i \leq x$ 
    or  $A[i, x] := A[i-1, x]$  //  $s_i > x$ 
    
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3		
3							
4							

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

```

for  $i = 1, 2, \dots, n$ 
  for  $x = 0, 1, \dots, C$ 
     $A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$  //  $s_i \leq x$ 
    or  $A[i, x] := A[i-1, x]$  //  $s_i > x$ 
    
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	x=0	1	2	3	4	5	6
i=0	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3							
4							

↑
n+1
↓

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3							
4							

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

```

for  $i = 1, 2, \dots, n$ 
  for  $x = 0, 1, \dots, C$ 
     $A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$  //  $s_i \leq x$ 
    or  $A[i, x] := A[i-1, x]$  //  $s_i > x$ 
    
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	x=0	1	2	3	4	5	6
i=0	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3							
4							

↑
n+1
↓

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	x=0	1	2	3	4	5	6
i=0	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0					
4							

↑
n+1
↓

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```


EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0					
4							

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

```

for  $i = 1, 2, \dots, n$ 
  for  $x = 0, 1, \dots, C$ 
     $A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$  //  $s_i \leq x$ 
    or  $A[i, x] := A[i-1, x]$  //  $s_i > x$ 
    
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4				
4							

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

```

for  $i = 1, 2, \dots, n$ 
  for  $x = 0, 1, \dots, C$ 
     $A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$  //  $s_i \leq x$ 
    or  $A[i, x] := A[i-1, x]$  //  $s_i > x$ 
    
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	x=0	1	2	3	4	5	6
i=0	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4		
4							

↑
n+1
↓

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4		
4					+4		

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

```

for  $i = 1, 2, \dots, n$ 
  for  $x = 0, 1, \dots, C$ 
     $A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$  //  $s_i \leq x$ 
    or  $A[i, x] := A[i-1, x]$  //  $s_i > x$ 
    
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	x=0	1	2	3	4	5	6
i=0	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	
4							

↑
n+1
↓

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x] , A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	3
4						+4	

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

```

for  $i = 1, 2, \dots, n$ 
  for  $x = 0, 1, \dots, C$ 
     $A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$  //  $s_i \leq x$ 
    or  $A[i, x] := A[i-1, x]$  //  $s_i > x$ 
    
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	x=0	1	2	3	4	5	6
i=0	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4							

↑
n+1
↓

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4							

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

```

for  $i = 1, 2, \dots, n$ 
  for  $x = 0, 1, \dots, C$ 
     $A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$  //  $s_i \leq x$ 
    or  $A[i, x] := A[i-1, x]$  //  $s_i > x$ 
    
```


EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4	0	0	4				

↑
n+1
↓

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

		← C+1 →						
		x=0	1	2	3	4	5	6
i=0		0	0	0	0	0	0	0
1		0	0	0	0	3	3	3
2		0	0	0	2	3	3	3
3		0	0	4	4	4	6	7
4		0	0	4				

↑
n+1
↓

+4

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	x=0	1	2	3	4	5	6
i=0	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4	0	0	4	4			

↑
n+1
↓

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	x=0	1	2	3	4	5	6
i=0	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4	0	0	4	4	4		

↑
n+1
↓

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

		$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$						
	$x=0$	1	2	3	4	5	6	
$i=0$	0	0	0	0	0	0	0	
1	0	0	0	0	3	3	3	
2	0	0	0	2	3	3	3	
3	0	0	4	4	4	6	7	
4	0	0	4	4	4			

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4	0	0	4	4	4	8	

↑
n+1
↓

capacity = 6

```

for i = 1, 2, ..., n
  for x = 0, 1, ..., C
    A[i, x] := max { A[i-1, x], A[i-1, x-si] + vi } // si ≤ x
    or A[i, x] := A[i-1, x] // si > x
  
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	$\xleftarrow{C+1} \xrightarrow{\hspace{1cm}}$						
	$x=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4	0	0	4	4	4	8	8

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

```

for  $i = 1, 2, \dots, n$ 
  for  $x = 0, 1, \dots, C$ 
     $A[i, x] := \max \{ A[i-1, x], A[i-1, x-s_i] + v_i \}$  //  $s_i \leq x$ 
    or  $A[i, x] := A[i-1, x]$  //  $s_i > x$ 
    
```

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$							
	$n=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4	0	0	4	4	4	8	8

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

optimal value

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

$\leftarrow C+1 \rightarrow$							
	$n=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4	0	0	4	4	4	8	8

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

How to reconstruct the solution?

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

$\xleftarrow{\quad C+1 \quad} \xrightarrow{\quad}$							
	$n=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4	0	0	4	4	4	8	8

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

Ask : Which branch led us to the current table entry?

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

$\leftarrow C+1 \rightarrow$							
	$n=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4	0	0	4	4	4	8	8

$\uparrow$
 $n+1$
 $\downarrow$

$+4$

capacity = 6

Ask : Which branch led us to the current table entry?

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

$\xleftarrow{C+1} \xrightarrow{\quad}$							
	$n=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4	0	0	4	4	4	8	8

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

Ask : Which branch led us to the current table entry?

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

$\xleftarrow{C+1} \xrightarrow{\hspace{1cm}}$

	$n=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4	0	0	4	4	4	8	8

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

Ask : Which branch led us to the current table entry?

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

← C+1 →

	n=0	1	2	3	4	5	6
i=0	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4	0	0	4	4	4	8	8

↑
n+1
↓

capacity = 6

Ask : Which branch led us to the current table entry?

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

$\xleftarrow{\quad} C+1 \xrightarrow{\quad}$							
	$n=0$	1	2	3	4	5	6
$i=0$	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4	0	0	4	4	4	8	8

$\uparrow$
 $n+1$
 $\downarrow$

capacity = 6

Ask : Which branch led us to the current table entry?

EXAMPLE

item	value	size
1	3	4
2	2	3
3	4	2
4	4	3

	← C+1 →						
	n=0	1	2	3	4	5	6
i=0	0	0	0	0	0	0	0
1	0	0	0	0	3	3	3
2	0	0	0	2	3	3	3
3	0	0	4	4	4	6	7
4	0	0	4	4	4	8	8

↑
n+1
↓

capacity = 6

optimal solution : items 3 and 4 .

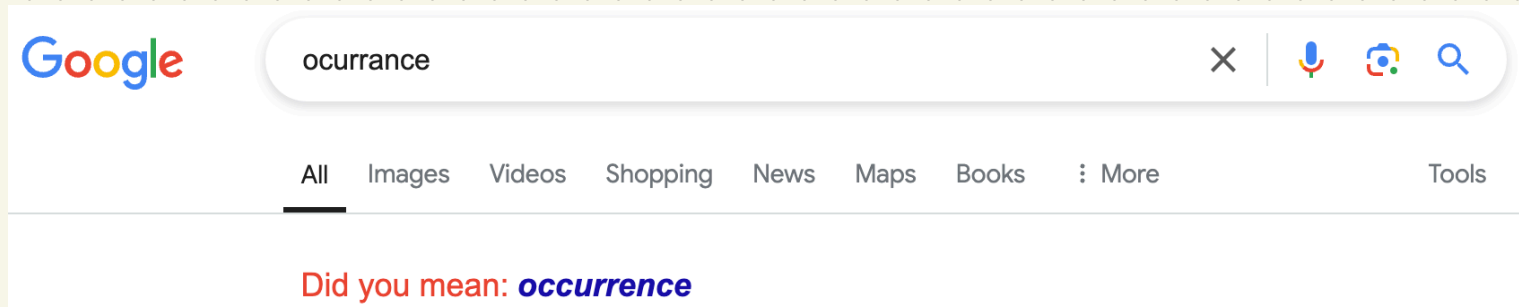
SEQUENCE ALIGNMENT

SEQUENCE ALIGNMENT

How similar are two given strings?

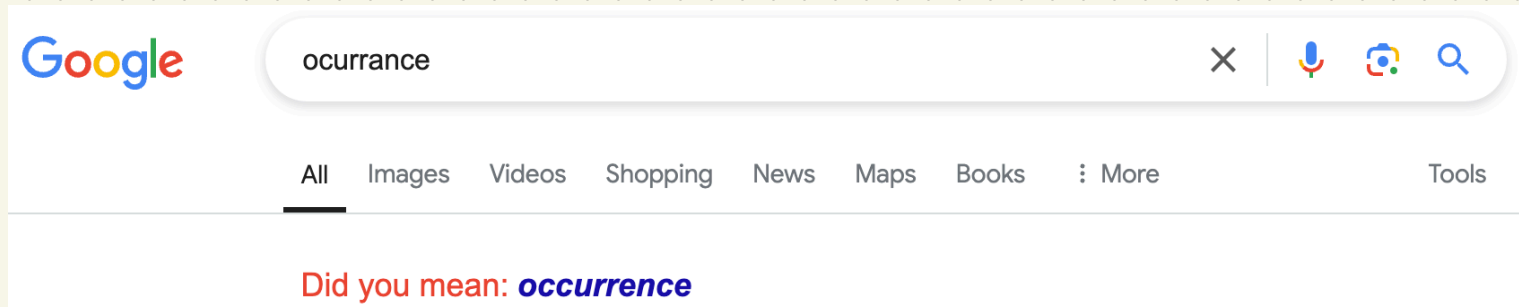
SEQUENCE ALIGNMENT

How similar are two given strings?



SEQUENCE ALIGNMENT

How similar are two given strings?



O _ c u r r a n c e

O c c u r r e n c e

SEQUENCE ALIGNMENT

How similar are two given strings?

ATTGCCCTTACAA TGCTAGGCACCTAGC

ATTGTCCCTTACACTGCTAGGCAGTAGC

SEQUENCE ALIGNMENT

How similar are two given strings?

ATTG CCTTACAA TGCTAGGCACTAGC

ATTG TCCTTACAC TGCTAGGCA GTAGC

Understanding the functions of specific regions of genomes

Understanding phylogenetics