OntarioTech Science

Kourosh Davoudi
kourosh@ontariotechu.ca

Lecture 6: Dynamic Programming II

CSCI 3070U: Design and Analysis of Algorithms

Learning Outcomes

- Dynamic Programming (DP):
  - Case Study:
    - Longest Common Subsequence (LCS)
    - Case Study: 0/1 Knapsack

Case Study: Longest Common Subsequence (LCS)

- What is the problem?
  - Given two sequences x[1..m] and y[1..n], find the longest subsequence which occurs in both

- Example: x = <A B C B D A B >, y = <B D C A B A>
  - <B C A B> and <B D> are both subsequences of both x and y
  - A subsequence doesn't have to be consecutive

LNC = < B C A B> or <B D A B> or ...

Length = 4

Case Study: Longest Common Subsequence (LCS)

- Brute-force solution:
  - For every subsequence of x, check if it's a subsequence of y?
  - 2^m subsequences of x to check (why?)
  - Each subsequence takes $\Theta(n)$ time to check

Time = $\Theta(n2^m)$

LCS Optimal Substructure

- Notations:

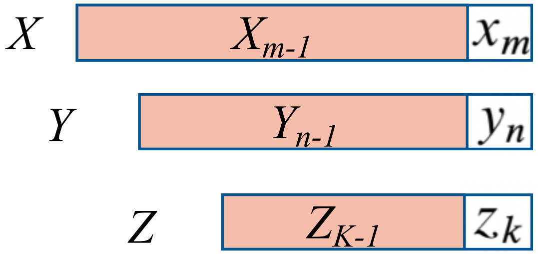
$$
\begin{aligned}
X_i &= \text{prefix } \langle x_1, \ldots, x_i \rangle \\
Y_i &= \text{prefix } \langle y_1, \ldots, y_i \rangle
\end{aligned}
$$

Let $Z = \langle z_1, \ldots, z_k \rangle$ be any LCS of X and Y

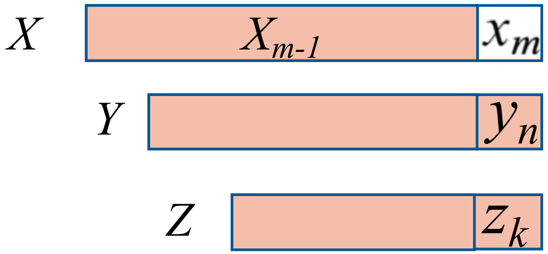
- How we can find the LCS length recursively?

LCS Optimal Substructure

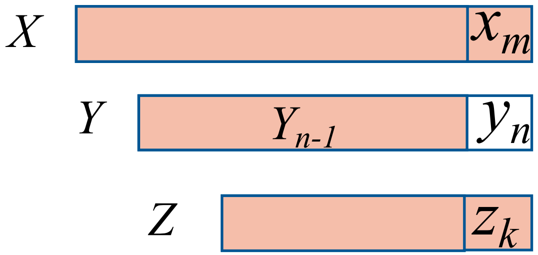
1. If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} is an LCS of X_{m-1} and Y_{n-1}.



2. If $x_m \neq y_n$, then $z_k \neq x_m \Rightarrow Z$ is an LCS of X_{m-1} and Y



3. If $x_m \neq y_n$, then $z_k \neq y_n \Rightarrow Z$ is an LCS of X and Y_{n-1}



LCS Optimal Substructure

- Notations:

$$\begin{aligned} X_i &= \text{prefix } \langle x_1, \ldots, x_i \rangle \\ Y_i &= \text{prefix } \langle y_1, \ldots, y_i \rangle \end{aligned}$$

Let $Z = \langle z_1, \ldots, z_k \rangle$ be any LCS of X and Y

1. If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} is an LCS of X_{m-1} and Y_{n-1}.
2. If $x_m \neq y_n$, then $z_k \neq x_m \Rightarrow Z$ is an LCS of X_{m-1} and Y
3. If $x_m \neq y_n$, then $z_k \neq y_n \Rightarrow Z$ is an LCS of X and Y_{n-1}

LCS Recursion Formula

- Define: $c[i, j] = \text{length of LCS of } X_i \text{ and } Y_j$

Case 1

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0\,, \\ c[i-1, j-1] + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(c[i-1, j], c[i, j-1]) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

Case 2 and 3

LCS-Length Algorithm

LCS-LENGTH(X, Y, m, n)

let $c[0\,..\,m, 0\,..\,n]$ be new tables
for $i = 1$ **to** m
    $c[i, 0] = 0$
for $j = 0$ **to** n
    $c[0, j] = 0$
for $i = 1$ **to** m
    for $j = 1$ **to** n
        if $x_i == y_j$
            $c[i, j] = c[i-1, j-1] + 1$
        else if $c[i-1, j] \geq c[i, j-1]$
            $c[i, j] = c[i-1, j]$
        else $c[i, j] = c[i, j-1]$
return $c[m, n]$

$$c[i, j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0\,, \\ c[i-1, j-1] + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\ \max(c[i-1, j], c[i, j-1]) & \text{if } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

Time: $\Theta(mn)$

LCS Example (0)

i		j: 0	1	2	3	4	5
		y_j	**B**	**D**	**C**	**A**	**B**
0	x_i						
1	**A**						
2	**B**						
3	**C**						
4	**B**						

X = ABCB; $m = |X| = 4$

Y = BDCAB; $n = |Y| = 5$

Allocate array $c[0..4,0..5]$

LCS Example (1)

i		j	0	1	2	3	4	5
			y_j	**B**	**D**	**C**	**A**	**B**
0	x_i		**0**	**0**	**0**	**0**	**0**	**0**
1	**A**		**0**					
2	**B**		**0**					
3	**C**		**0**					
4	**B**		**0**					

ABCB
BDCAB

for $i = 1$ **to** m
$\quad c[i, 0] = 0$
for $j = 0$ **to** n
$\quad c[0, j] = 0$

LCS Example (2)

i		j	0	1	2	3	4	5
		y_j		B	D	C	A	B
0	x_i		0	0	0	0	0	0
1	A		0	0				
2	B		0					
3	C		0					
4	B		0					

ABCB
BDCAB

if $x_i == y_j$
$c[i, j] = c[i-1, j-1] + 1$
else if $c[i-1, j] \geq c[i, j-1]$
$c[i, j] = c[i-1, j]$
else $c[i, j] = c[i, j-1]$

LCS Example (3)

	j	0	1	2	3	4	5
i		y_j	B	D	C	A	B
0	x_i	0	0	0	0	0	0
1	A	0	0	0	0		
2	B	0					
3	C	0					
4	B	0					

ABCB
BDCAB

if $x_i == y_j$
$c[i, j] = c[i-1, j-1] + 1$
else if $c[i-1, j] \geq c[i, j-1]$
$c[i, j] = c[i-1, j]$
else $c[i, j] = c[i, j-1]$

LCS Example (4)

	j	0	1	2	3	4	5
i		y_j	**B**	**D**	**C**	**A**	**B**
0	x_i	**0**	**0**	**0**	**0**	**0**	**0**
1	**A**	**0**	**0**	**0**	**0**	**1**	
2	**B**	**0**					
3	**C**	**0**					
4	**B**	**0**					

ABCB
BDCAB

if $x_i == y_j$
$c[i, j] = c[i-1, j-1] + 1$
else if $c[i-1, j] \geq c[i, j-1]$
$c[i, j] = c[i-1, j]$
else $c[i, j] = c[i, j-1]$

LCS Example (5)

i	j	0	1	2	3	4	5
		y_j	B	D	C	A	B
0	x_i	**0**	**0**	**0**	**0**	**0**	**0**
1	A	**0**	**0**	**0**	**0**	**1** →	**1**
2	B	**0**					
3	C	**0**					
4	B	**0**					

ABCB
BDCAB

if $x_i == y_j$
$c[i, j] = c[i-1, j-1] + 1$
else if $c[i-1, j] \geq c[i, j-1]$
$c[i, j] = c[i-1, j]$
else $c[i, j] = c[i, j-1]$

LCS Example (6)

	j	0	1	2	3	4	5
i		y_j	B	D	C	A	B
0	x_i	0	0	0	0	0	0
1	A	0	0	0	0	1	1
2	B	0	1				
3	C	0					
4	B	0					

ABCB
BDCAB

if $x_i == y_j$
$c[i, j] = c[i-1, j-1] + 1$
else if $c[i-1, j] \geq c[i, j-1]$
$c[i, j] = c[i-1, j]$
else $c[i, j] = c[i, j-1]$

LCS Example (7)

i		j	0	1	2	3	4	5
		y_j		B	D	C	A	B
0	x_i		0	0	0	0	0	0
1	A		0	0	0	0	1	1
2	B		0	1 →	1 →	1		
3	C		0					
4	B		0					

ABCB
BDCAB

if $x_i == y_j$
$c[i, j] = c[i-1, j-1] + 1$
else if $c[i-1, j] \geq c[i, j-1]$
$c[i, j] = c[i-1, j]$
else $c[i, j] = c[i, j-1]$

LCS Example (8)

i		j	0	1	2	3	4	5
			y_j	**B**	**D**	**C**	**A**	**B**
0	x_i		**0**	**0**	**0**	**0**	**0**	**0**
1	**A**		**0**	**0**	**0**	**0**	**1**	**1**
2	**B**		**0**	**1**	**1**	**1**	**1**	
3	**C**		**0**					
4	**B**		**0**					

ABCB
BDCAB

if $x_i == y_j$
$c[i, j] = c[i-1, j-1] + 1$
else if $c[i-1, j] \geq c[i, j-1]$
$c[i, j] = c[i-1, j]$
else $c[i, j] = c[i, j-1]$

LCS Example (9)

i		j	0	1	2	3	4	5
			y_j	B	D	C	A	B
0	x_i		0	0	0	0	0	0
1	A		0	0	0	0	1	1
2	B		0	1	1	1	1	2
3	C		0					
4	B		0					

ABCB
BDCAB

$$\boxed{\begin{aligned}&\textbf{if } x_i == y_j \\ &\qquad c[i,j] = c[i-1,j-1]+1\end{aligned}}$$

$$\begin{aligned}&\textbf{else if } c[i-1,j] \ge c[i,j-1] \\ &\qquad\qquad c[i,j] = c[i-1,j] \\ &\textbf{else } c[i,j] = c[i,j-1]\end{aligned}$$

LCS Example (10)

i		j	0	1	2	3	4	5
			y_j	B	D	C	A	B
0	x_i		0	0	0	0	0	0
1	A		0	0	0	0	1	1
2	B		0	1	1	1	1	2
3	C		0	1	1			
4	B		0					

ABCB
BDCAB

if $x_i == y_j$
$c[i, j] = c[i-1, j-1] + 1$
else if $c[i-1, j] \geq c[i, j-1]$
$c[i, j] = c[i-1, j]$
else $c[i, j] = c[i, j-1]$

LCS Example (11)

j		0	1	2	3	4	5
i		y_j	B	D	C	A	B
0	x_i	0	0	0	0	0	0
1	A	0	0	0	0	1	1
2	B	0	1	1	1	1	2
3	C	0	1	1	2		
4	B	0					

ABCB
BDCAB

$$\boxed{\begin{aligned}&\textbf{if } x_i == y_j\\ &\quad c[i,j] = c[i-1,j-1]+1\end{aligned}}$$

$$\begin{aligned}&\textbf{else if } c[i-1,j] \geq c[i,j-1]\\ &\qquad c[i,j] = c[i-1,j]\\ &\textbf{else } c[i,j] = c[i,j-1]\end{aligned}$$

LCS Example (12)

	j	0	1	2	3	4	5
i		y_j	**B**	**D**	**C**	**A**	**B**
0	x_i	**0**	**0**	**0**	**0**	**0**	**0**
1	**A**	**0**	**0**	**0**	**0**	**1**	**1**
2	**B**	**0**	**1**	**1**	**1**	**1**	**2**
3	**C**	**0**	**1**	**1**	**2**	**2**	
4	**B**	**0**					

ABCB
BDCAB

if $x_i == y_j$
$c[i, j] = c[i-1, j-1] + 1$
else if $c[i-1, j] \geq c[i, j-1]$
$c[i, j] = c[i-1, j]$
else $c[i, j] = c[i, j-1]$

LCS Example (13)

	j	0	1	2	3	4	5
i		y_j	B	D	C	A	B
0	x_i	0	0	0	0	0	0
1	A	0	0	0	0	1	1
2	B	0	1	1	1	1	2
3	C	0	1	1	2	2	2
4	B	0					

ABCB
BDCAB

if $x_i == y_j$
$c[i, j] = c[i-1, j-1] + 1$
$\boxed{\textbf{else if } c[i-1, j] \geq c[i, j-1] \quad c[i, j] = c[i-1, j]}$
else $c[i, j] = c[i, j-1]$

LCS Example (14)

i		j	0	1	2	3	4	5
		y_j		B	D	C	A	B
0	x_i		0	0	0	0	0	0
1	A		0	0	0	0	1	1
2	B		0	1	1	1	1	2
3	C		0	1	1	2	2	2
4	B		0	1				

ABCB
BDCAB

if $x_i == y_j$
$c[i, j] = c[i-1, j-1] + 1$
else if $c[i-1, j] \geq c[i, j-1]$
$c[i, j] = c[i-1, j]$
else $c[i, j] = c[i, j-1]$

LCS Example (15)

i		j: 0	1	2	3	4	5
		y_j	B	D	C	A	B
0	x_i	0	0	0	0	0	0
1	A	0	0	0	0	1	1
2	B	0	1	1	1	1	2
3	C	0	1	1	2	2	2
4	B	0	1	1	2	2	

ABCB

BDCAB

if $x_i == y_j$

$c[i, j] = c[i-1, j-1] + 1$

$\boxed{\textbf{else if } c[i-1, j] \geq c[i, j-1] \quad c[i, j] = c[i-1, j]}$

else $c[i, j] = c[i, j-1]$

LCS Example (16)

	j	0	1	2	3	4	5
i		y_j	B	D	C	A	B
0	x_i	0	0	0	0	0	0
1	A	0	0	0	0	1	1
2	B	0	1	1	1	1	2
3	C	0	1	1	2	2	2
4	B	0	1	1	2	2	3

ABCB
BDCAB

if $x_i == y_j$
$c[i, j] = c[i-1, j-1] + 1$
else if $c[i-1, j] \geq c[i, j-1]$
$c[i, j] = c[i-1, j]$
else $c[i, j] = c[i, j-1]$

LCS Algorithm

```
LCS-LENGTH(X, Y, m, n)
    let c[0..m, 0..n] be new tables
    for i = 1 to m
        c[i, 0] = 0
    for j = 0 to n
        c[0, j] = 0
    for i = 1 to m
        for j = 1 to n
            if x_i == y_j
                c[i, j] = c[i - 1, j - 1] + 1
            else if c[i - 1, j] ≥ c[i, j - 1]
                    c[i, j] = c[i - 1, j]
            else c[i, j] = c[i, j - 1]
    return c[m, n]
```

Time:

$$\Theta(mn)$$

Case Study: Knapsack Algorithm

- Problem:
  - We have a knapsack with capacity W, and a number of item, where each item has a weight, and a value
  - Objective: select the items with maximum total value and putting them in knapsack

- Variations:
  - 0/1: We can decide to select / NOT to select (no division)
  - Fractional: We can divide items and take a part of them, for part of the value

0/1 - Knapsack Problem:

- Problem Formulation:
  - There are n items available
  - The knapsack can store W total weight
  - Each i'th item has value b_i and weight w_i
  - ***Goal***: Find the maximum value that we put in Knapsack

- Example:
  - There are four available items (n=4)
    - W_1=2kg, b_1=\$12
    - W_2=4kg, b_2=\$24
    - W_3=5kg, b_3=\$28
    - W_4=8kg, b_4=\$41
  - The knapsack can hold 11 kg of items

What is the maximum value that we can put in Knapsack?

0/1 - Knapsack problem

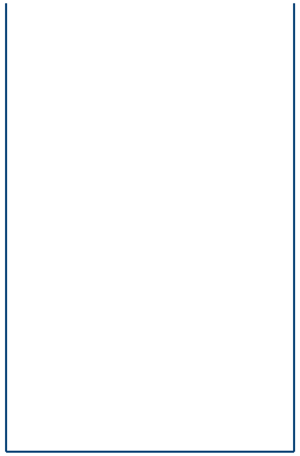
- Brute Force Algorithm:
  - Try all combinations of the n items
  - Find the maximum value of the combinations
    - Number of combinations?

$$O(2^n)$$

- Solution:
  - Dynamic Programming (optimum)
  - Branch and Bound (optimum)
  - Greedy (not optimum)

0/1 - Knapsack problem

Idea: choose the items with highest values

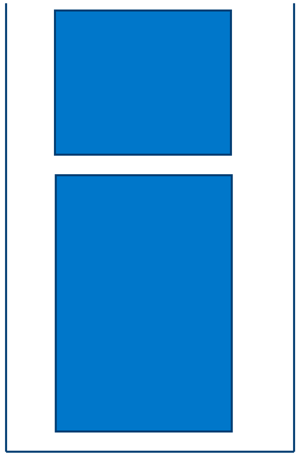


Max weight: W = 15

Items	*Weight* w_i	*Benefit value* b_i
1	2	3
2	3	4
3	4	5
4	5	8
5	9	10

0/1 - Knapsack problem

Idea: choose the items with highest values



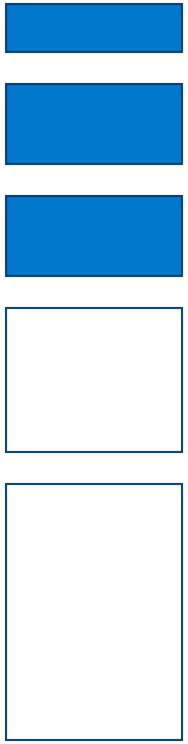
B = 10 + 8 = 18

Max weight: W = 15

Best answer: B = 8 + 5 + 4 + 3 = 20

Items = {5 , 4 , 3 , 2}

Items



Weight w_i	Benefit value b_i
2	3
3	4
4	5
5	8
9	10

Knapsack Optimum Subproblem

- The items are labeled 1..n
- Let S_k be an optimal solution for the set items labeled as 1, 2, .. k
- General Idea: The best subset of S_k that has maximum total weight w is:

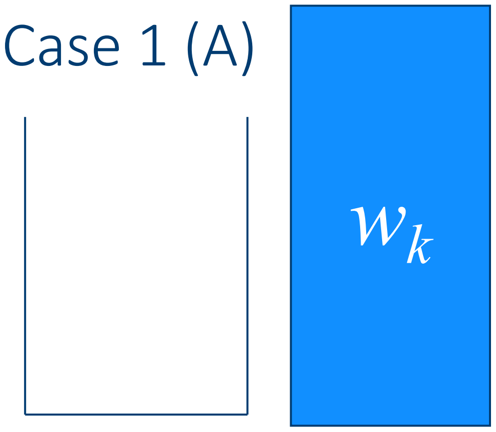
1. The best subset of S_{k-1} that has total weight w
   - A) We cannot select
   - B) We can but we do not select

   OR

2. The best subset of S_{k-1} that has maximum total weight w-w_k plus the value of item k

Knapsack Optimum Subproblem

Case 1 (A)

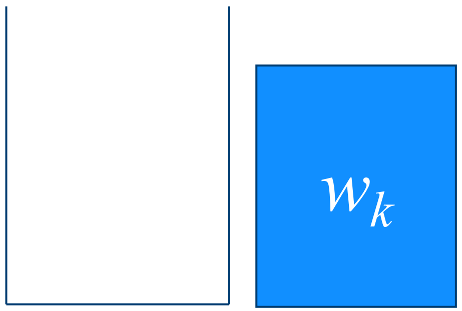


w

$w_k > w$

$B[k, w] = B[k-1, w]$

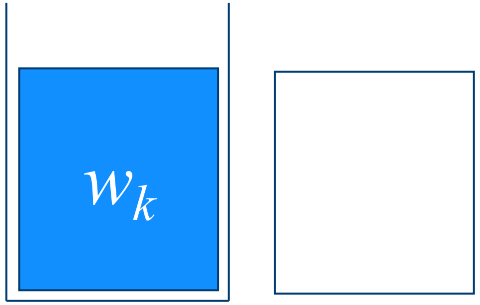
Case 1 (B)



w

$w_k \leq w$

Case 2



w - w_k

$w_k \leq w$

$B[k, w] = \max\{B[k-1, w], b_k + B[k-1, w-w_k]\}$

Assume $B[k,\ w]$ is a best value for the set S_k

0/1 - Knapsack Optimum Subproblem

- Recursive Formulation :
  - Assume that $B[k, w]$ is a best value for the set S_k whose sum of item weights is less than w

$$B[k,w] = \begin{cases} B[k-1,w] & \text{if } w_k > w \\ \max\{B[k-1,w], B[k-1,w-w_k]+b_k\} & \text{else} \end{cases}$$

  - *Case* $w_k > w$:Item k can't be part of the solution, since if it was, the total weight would be $> w$, which is unacceptable
  - Case $w_k \le w$: Then the item k **can** be in the solution,

we choose the case with greater value

0/1 - Knapsack Algorithm

KNAPSSACK-0/1($w_{1..n}, b_{1:n}, W$)

for $w = 0$ to W $\quad \Theta(W)$
    $B[0, w] = 0$
for $i = 0$ to n $\quad \Theta(n)$
    $B[i, 0] = 0$
for $i = 0$ to n
    for $w = 0$ to W
        if $w_i \leq w$
            if $b_i + B[i-1, w-w_i] > B[i-1, w]$
                $B[i, w] = b_i + B[i-1, w-w_i]$
            else
                $B[i, w] = B[i-1, w]$
        else
            $B[i, w] = B[i-1, w]$
return $B[n, w]$

$\Theta(nW)$

Time Complexity:

$\Theta(nW)$

To know the items that make this maximum value, an addition to this algorithm is necessary

0/1 - Knapsack Example

w \ i	0	1	2	3	4
0					
1					
2					
3					
4					
5					

n = 4 (# of elements)

W = 5 (max weight)

Elements (weight, benefit):

(2,3), (3,4), (4,5), (5,6)

$$B[k,w] = \begin{cases} B[k-1,w] & \text{if } w_k > w \\ \max\{B[k-1,w], B[k-1,w-w_k]+b_k\} & \text{else} \end{cases}$$

From: https://www.cs.virginia.edu/~luebke/cs332/index.html

0/1 - Knapsack Example

w \ i	0	1	2	3	4
0	0				
1	0				
2	0				
3	0				
4	0				
5	0				

1 **for** $w = 0$ to W

2 $\quad B[0, w] = 0$

0/1 - Knapsack Example

w \ i	0	1	2	3	4
0	0	0	0	0	0
1	0				
2	0				
3	0				
4	0				
5	0				

for $i = 0$ to n

$\qquad B[i, 0] = 0$

0/1 - Knapsack Example

Items:

$\boxed{1: (2,3)}$

2: (3,4)

3: (4,5)

4: (5,6)

w \ i	0	1	2	3	4
0	0	0	0	0	0
1	0 →	0			
2	0				
3	0				
4	0				
5	0				

$i = 1$

$b_i = 3$

$w_i = 2$

$w = 1$

$w - w_i = -1$

if $w_i \leq w$

  if $b_i + B[i-1, w-w_i] > B[i-1, w]$

    $B[i,w] = b_i + B[i-1, w-w_i]$

  else

    $B[i,w] = B[i-1, w]$

else

  $\boxed{B[i,w] = B[i-1, w]}$

0/1 - Knapsack Example

Items:

1: (2,3)

2: (3,4)

3: (4,5)

4: (5,6)

w \ i	0	1	2	3	4
0	0	0	0	0	0
1	0	0			
2	0	**3**			
3	0				
4	0				
5	0				

$i=1$

$b_i=3$

$w_i=2$

$w=2$

$w-w_i=0$

if $w_i \le w$

  if $b_i + B[i-1, w-w_i] > B[i-1, w]$

    $\boxed{B[i,w] = b_i + B[i-1, w-w_i]}$

  else

    $B[i,w] = B[i-1,w]$

else

  $B[i,w] = B[i-1,w]$

0/1 - Knapsack Example

Items:

$\boxed{1: (2,3)}$

2: (3,4)

3: (4,5)

4: (5,6)

w \ i	0	1	2	3	4
0	0	0	0	0	0
1	0	0			
2	0	3			
3	0	3			
4	0				
5	0				

i =1

b_i=3

w_i=2

w=3

w-w_i=1

if $w_i \le w$

 if $b_i + B[i-1, w-w_i] > B[i-1, w]$

  $\boxed{B[i,w] = b_i + B[i-1, w-w_i]}$

 else

  $B[i,w] = B[i-1, w]$

else

 $B[i,w] = B[i-1, w]$

0/1 - Knapsack Example

Items:

$\boxed{1: (2,3)}$
2: (3,4)
3: (4,5)
4: (5,6)

w \ i	0	1	2	3	4
0	0	0	0	0	0
1	0	0			
2	0	3			
3	0	3			
4	0	**3**			
5	0				

$i=1$

$b_i=3$

$w_i=2$

$w=4$

$w\text{-}w_i=2$

if $w_i \leq w$
  if $b_i + B[i-1, w-w_i] > B[i-1, w]$
    $\boxed{B[i,w] = b_i + B[i-1, w-w_i]}$
  else
    $B[i,w] = B[i-1,w]$
else $B[i,w] = B[i-1,w]$

0/1 - Knapsack Example

Items:

1: (2,3)

2: (3,4)

3: (4,5)

4: (5,6)

w \ i	0	1	2	3	4
0	0	0	0	0	0
1	0	0			
2	0	3			
3	0	3			
4	0	3			
5	0	3			

i=1

b_i=3

w_i=2

w=5

w-w_i=3

if $w_i \le w$
  if $b_i + B[i-1, w-w_i] > B[i-1, w]$
    $\boxed{B[i, w] = b_i + B[i-1, w-w_i]}$
  else
    $B[i, w] = B[i-1, w]$
else
  $B[i, w] = B[i-1, w]$

0/1 - Knapsack Example

Items:

1: (2,3)
2: (3,4)

3: (4,5)
4: (5,6)

$w \backslash i$	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0		
2	0	3			
3	0	3			
4	0	3			
5	0	3			

$i=2$
$b_i=4$
$w_i=3$
$w=1$
$w-w_i=-2$

if $w_i \leq w$
  if $b_i + B[i-1, w-w_i] > B[i-1, w]$
    $B[i, w] = b_i + B[i-1, w-w_i]$
  else
    $B[i, w] = B[i-1, w]$
else $\boxed{B[i, w] = B[i-1, w]}$

0/1 - Knapsack Example

Items:

1: (2,3)
2: (3,4)

3: (4,5)

4: (5,6)

w \ i	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0		
2	0	3	3		
3	0	3			
4	0	3			
5	0	3			

$i=2$

$b_i=4$

$w_i=3$

$w=2$

$w-w_i=-1$

if $w_i \leq w$

  if $b_i + B[i-1, w-w_i] > B[i-1, w]$

    $B[i,w] = b_i + B[i-1, w-w_i]$

  else

    $B[i,w] = B[i-1, w]$

else $\boxed{B[i,w] = B[i-1, w]}$

0/1 - Knapsack Example

Items:

1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

w \ i	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0		
2	0	3	3		
3	0	3	4		
4	0	3			
5	0	3			

$i=2$

$b_i=4$

$w_i=3$

$w=3$

$w-w_i=0$

if $w_i \le w$
  if $b_i + B[i-1, w-w_i] > B[i-1, w]$
    $\boxed{B[i,w] = b_i + B[i-1, w-w_i]}$
  else
    $B[i,w] = B[i-1, w]$
else
  $B[i,w] = B[i-1, w]$

0/1 - Knapsack Example

Items:

1: (2,3)
2: (3,4)

3: (4,5)

4: (5,6)

w \ i	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0		
2	0	3	3		
3	0	3	4		
4	0	3	**4**		
5	0	3			

$i=2$

$b_i=4$

$w_i=3$

$w=4$

$w\text{-}w_i=1$

if $w_i \leq w$

  if $b_i + B[i-1, w-w_i] > B[i-1, w]$

    $\boxed{B[i,w] = b_i + B[i-1, w-w_i]}$

  else

    $B[i,w] = B[i-1, w]$

else $B[i,w] = B[i-1, w]$

0/1 - Knapsack Example

Items:

1: (2,3)
2: (3,4)

3: (4,5)

4: (5,6)

w \ i	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0		
2	0	3	3		
3	0	3	4		
4	0	3	4		
5	0	3	7		

$i=2$

$b_i=4$

$w_i=3$

$w=5$

$w-w_i=2$

if $w_i \le w$
  if $b_i + B[i-1, w-w_i] > B[i-1, w]$
    $\boxed{B[i,w] = b_i + B[i-1, w-w_i]}$
  else
    $B[i,w] = B[i-1,w]$
else $B[i,w] = B[i-1,w]$

0/1 - Knapsack Example

Items:

1: (2,3)
2: (3,4)
3: (4,5)

4: (5,6)

w \ i	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	**0**	
2	0	3	3	**3**	
3	0	3	4	**4**	
4	0	3	4		
5	0	3	7		

i=3

b_i=5

w_i=4

w=1..3

$$
\begin{array}{l}
\textbf{if } w_i \le w \\
\quad \textbf{if } b_i + B[i-1, w-w_i] > B[i-1, w] \\
\quad \quad B[i,w] = b_i + B[i-1, w-w_i] \\
\quad \textbf{else} \\
\quad \quad B[i,w] = B[i-1, w] \\
\textbf{else} \\
\quad \boxed{B[i,w] = B[i-1, w]}
\end{array}
$$

0/1 - Knapsack Example

Items:

1: (2,3)
2: (3,4)
3: (4,5)

4: (5,6)

w \ i	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	
2	0	3	3	3	
3	0	3	4	4	
4	0	3	4	5	
5	0	3	7		

$i = 3$

$b_i = 5$

$w_i = 4$

$w = 4$

$w - w_i = 0$

if $w_i \le w$
    if $b_i + B[i-1, w-w_i] > B[i-1, w]$
        $\boxed{B[i,w] = b_i + B[i-1, w-w_i]}$
    else
        $B[i,w] = B[i-1, w]$
else
    $B[i,w] = B[i-1, w]$

0/1 - Knapsack Example

Items:

1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

w \ i	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	
2	0	3	3	3	
3	0	3	4	4	
4	0	3	4	5	
5	0	3	7 →	7	

i=3
b_i=5
w_i=4
w=5
w-w_i=1

if $w_i \le w$
  if $b_i + B[i-1, w-w_i] > B[i-1, w]$
    $B[i, w] = b_i + B[i-1, w-w_i]$
  else
    $\boxed{B[i, w] = B[i-1, w]}$
else
  $B[i, w] = B[i-1, w]$

0/1 - Knapsack Example

w \ i	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0 →	**0**
2	0	3	3	3 →	**3**
3	0	3	4	4 →	**4**
4	0	3	4	5 →	**5**
5	0	3	7	7	

$i=3$
$b_i=5$
$w_i=4$
$w=1..4$

Items:

1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

if $w_i \leq w$
  if $b_i + B[i-1, w-w_i] > B[i-1, w]$
    $B[i, w] = b_i + B[i-1, w-w_i]$
  else
    $B[i, w] = B[i-1, w]$
else $\boxed{B[i, w] = B[i-1, w]}$

0/1 - Knapsack Example

w \ i	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	3	3	3	3
3	0	3	4	4	4
4	0	3	4	5	5
5	0	3	7	7	7

i=3
b_i=5
w_i=4
w=5

Items:

1: (2,3)
2: (3,4)
3: (4,5)
4: (5,6)

if $w_i \le w$
  if $b_i + B[i-1, w-w_i] > B[i-1, w]$
    $B[i, w] = b_i + B[i-1, w-w_i]$
  else
    $\boxed{B[i, w] = B[i-1, w]}$
else
  $B[i, w] = B[i-1, w]$

Wrap-up

- We learned more advance DP techniques using two more case studies:
  - Longest Common Subsequence (LCS)
  - 0/1 - Knapsack Problem