

# Algorithm Problem Solving (APS): Dynamic Programming

---

Niema Moshiri

UC San Diego SPIS 2019

# Example: The 0-1 Knapsack Problem

- Imagine you're in a heist crew en route to your next job



# Example: The 0-1 Knapsack Problem

- Imagine you're in a heist crew en route to your next job
- Your target is a truck carrying electronic goods



# Example: The 0-1 Knapsack Problem

- Imagine you're in a heist crew en route to your next job
- Your target is a truck carrying electronic goods
- The modified Honda Civic you're driving has a weight limit

# Example: The 0-1 Knapsack Problem

- Imagine you're in a heist crew en route to your next job
- Your target is a truck carrying electronic goods
- The modified Honda Civic you're driving has a weight limit
  - You know how much each item costs and weighs

# Example: The 0-1 Knapsack Problem

- Imagine you're in a heist crew en route to your next job
- Your target is a truck carrying electronic goods
- The modified Honda Civic you're driving has a weight limit
  - You know how much each item costs and weighs
  - Which items do you steal to maximize your profits?

# Example: The 0-1 Knapsack Problem

- **Input:** A list of  $n$  items, where the  $i$ -th item has weight  $\mathbf{w}_i$  and value  $\mathbf{v}_i$ ,  
and a capacity  $\mathbf{x}$

# Example: The 0-1 Knapsack Problem

- **Input:** A list of  $n$  items, where the  $i$ -th item has weight  $w_i$  and value  $v_i$ , and a capacity  $x$
- **Output:** A set of items such that the sum of their weights is below  $x$  and the sum of their values is **maximized**



## Example: The 0-1 Knapsack Problem

$w$ :	3	8	10	6
$v$ :	\$5	\$5	\$13	\$6

$$x = 20$$

Ex

Can we use a Greedy Algorithm?

$w$ :	3	8	10	6
$v$ :	\$5	\$5	\$13	\$6

$$x = 20$$

Ex

Can we use a Greedy Algorithm?

$w$ :	3	8	10	6
$v$ :	\$5	\$5	\$13	\$6

$$x = 20$$

Nope! Not guaranteed to be optimal!

# Simpler Example: Fibonacci Sequence

# Simpler Example: Fibonacci Sequence

- The first number is 0

0

# Simpler Example: Fibonacci Sequence

- The first number is 0
- The second number is 1

0, 1

# Simpler Example: Fibonacci Sequence

- The first number is 0
- The second number is 1
- Each subsequent number is the sum of the two before it

0, 1

# Simpler Example: Fibonacci Sequence

- The first number is 0
- The second number is 1
- Each subsequent number is the sum of the two before it

0, 1



# Simpler Example: Fibonacci Sequence

- The first number is 0
- The second number is 1
- Each subsequent number is the sum of the two before it

0, 1, 1

# Simpler Example: Fibonacci Sequence

- The first number is 0
- The second number is 1
- Each subsequent number is the sum of the two before it

0, 1, 1

# Simpler Example: Fibonacci Sequence

- The first number is 0
- The second number is 1
- Each subsequent number is the sum of the two before it

0, 1, 1, 2

# Simpler Example: Fibonacci Sequence

- The first number is 0
- The second number is 1
- Each subsequent number is the sum of the two before it

0, 1, 1, 2

# Simpler Example: Fibonacci Sequence

- The first number is 0
- The second number is 1
- Each subsequent number is the sum of the two before it

0, 1, 1, 2, 3

# Simpler Example: Fibonacci Sequence

- The first number is 0
- The second number is 1
- Each subsequent number is the sum of the two before it

0, 1, 1, **2**, **3**

# Simpler Example: Fibonacci Sequence

- The first number is 0
- The second number is 1
- Each subsequent number is the sum of the two before it

0, 1, 1, 2, 3, 5

# Simpler Example: Fibonacci Sequence

- The first number is 0
- The second number is 1
- Each subsequent number is the sum of the two before it

0, 1, 1, 2, **3**, **5**



# Simpler Example: Fibonacci Sequence

- The first number is 0
- The second number is 1
- Each subsequent number is the sum of the two before it

0, 1, 1, 2, 3, 5, 8

# Simpler Example: Fibonacci Sequence

- The first number is 0
- The second number is 1
- Each subsequent number is the sum of the two before it

0, 1, 1, 2, 3, 5, 8

# Simpler Example: Fibonacci Sequence

- The first number is 0
- The second number is 1
- Each subsequent number is the sum of the two before it

0, 1, 1, 2, 3, 5, 8, **13**

# Simpler Example: Fibonacci Sequence

- The first number is 0
- The second number is 1
- Each subsequent number is the sum of the two before it

0, 1, 1, 2, 3, 5, 8, 13, ...

# Simpler Example: Fibonacci Sequence

- **Input:** An integer  $n$

# Simpler Example: Fibonacci Sequence

- **Input:** An integer  $n$
- **Output:** The  $n$ -th number of the Fibonacci sequence (0-based indexing)

# Simpler Example: Fibonacci Sequence

- **Input:** An integer  $n$
- **Output:** The  $n$ -th number of the Fibonacci sequence (0-based indexing)

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n - 1) + F(n - 2) & \text{if } n > 1 \end{cases}$$

## Simpler Example: Fibonacci Sequence

- Write a recursive Python function!
- **Output:** The  $n$ -th number of the Fibonacci sequence (0-based indexing)

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n - 1) + F(n - 2) & \text{if } n > 1 \end{cases}$$



# Recursive Fibonacci

```
def fib_rec(n):  
    if n == 0:  
        return 0  
  
    elif n == 1:  
        return 1  
  
    else:  
        return fib_rec(n-1) + fib_rec(n-2)
```

# Recursive Fibonacci

```
def fib_rec(n):  
    if n == 0:  
        return 0  
  
    elif n == 1:  
        return 1  
  
    else:  
        return fib_rec(n-1) + fib_rec(n-2)
```

Yay! 😊

# Recursive Fibonacci

```
def fib_rec(n):  
    if n == 0:  
        return 0  
  
    elif n == 1:  
        return 1  
  
    else:  
        return fib_rec(n-1) + fib_rec(n-2)
```

Hold up... 🤔

# Recursive Fibonacci: SLOW!!!

rec\_fib(5)

```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```

# Recursive Fibonacci: SLOW!!!

rec\_fib(5)

```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```

# Recursive Fibonacci: SLOW!!!

rec\_fib(5)

```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```

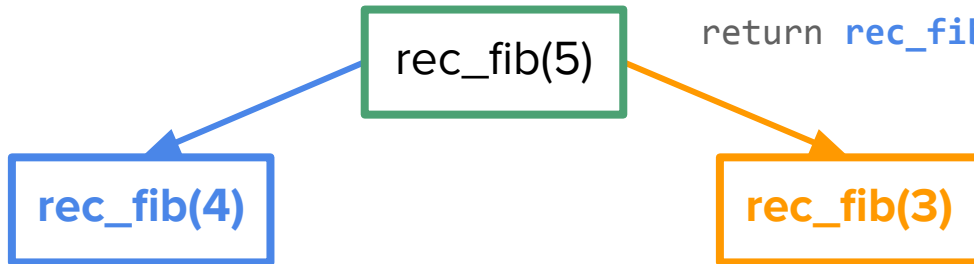
# Recursive Fibonacci: SLOW!!!

rec\_fib(5)

```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```

# Recursive Fibonacci: SLOW!!!

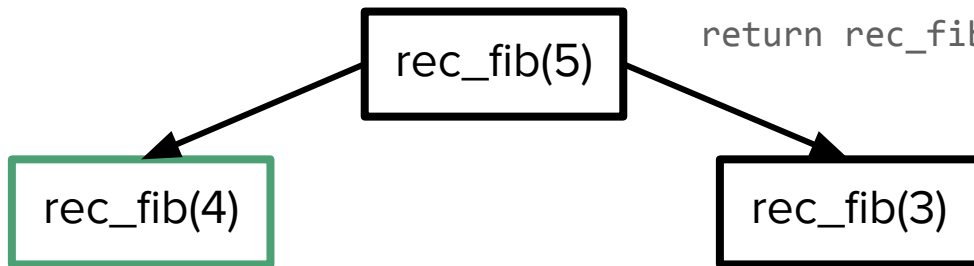
```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```





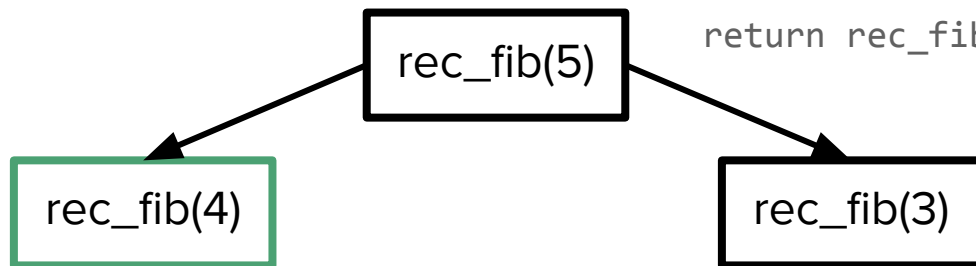
# Recursive Fibonacci: SLOW!!!

```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```



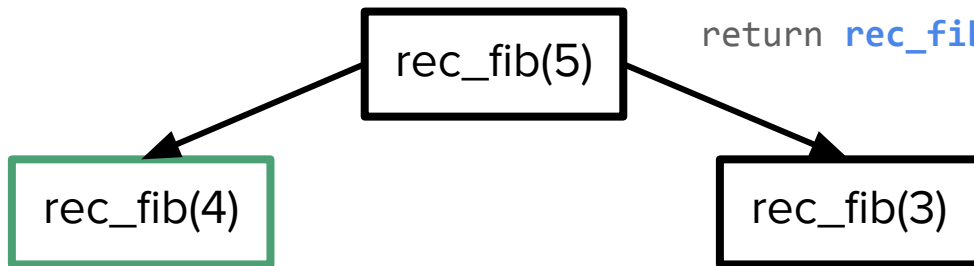
# Recursive Fibonacci: SLOW!!!

```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```



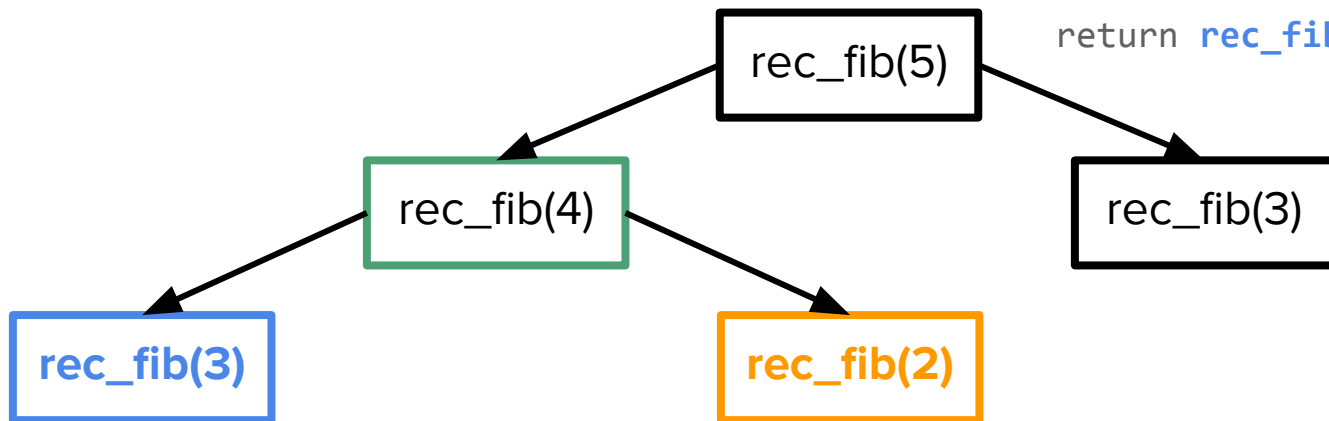
# Recursive Fibonacci: SLOW!!!

```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```



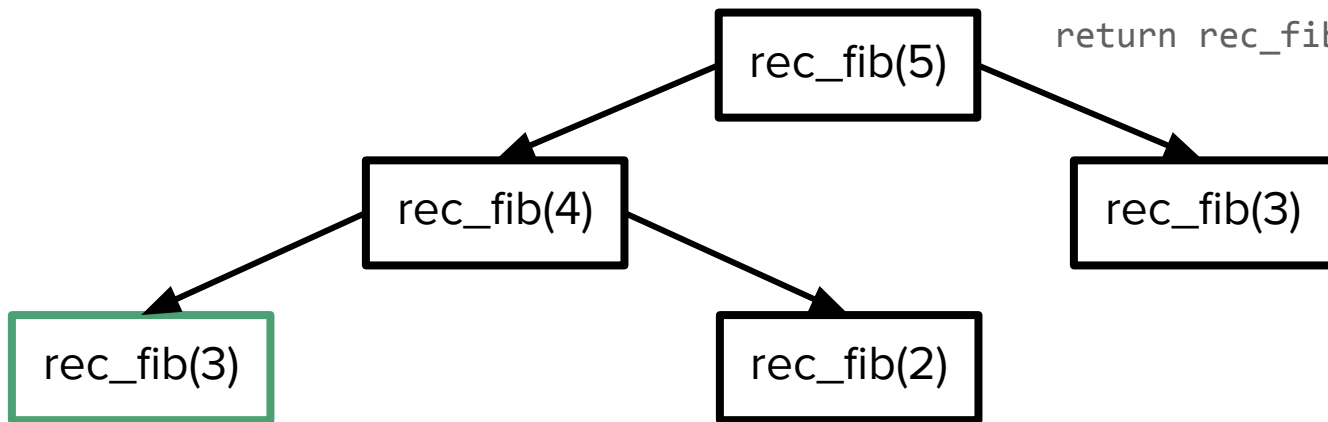
# Recursive Fibonacci: SLOW!!!

```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```



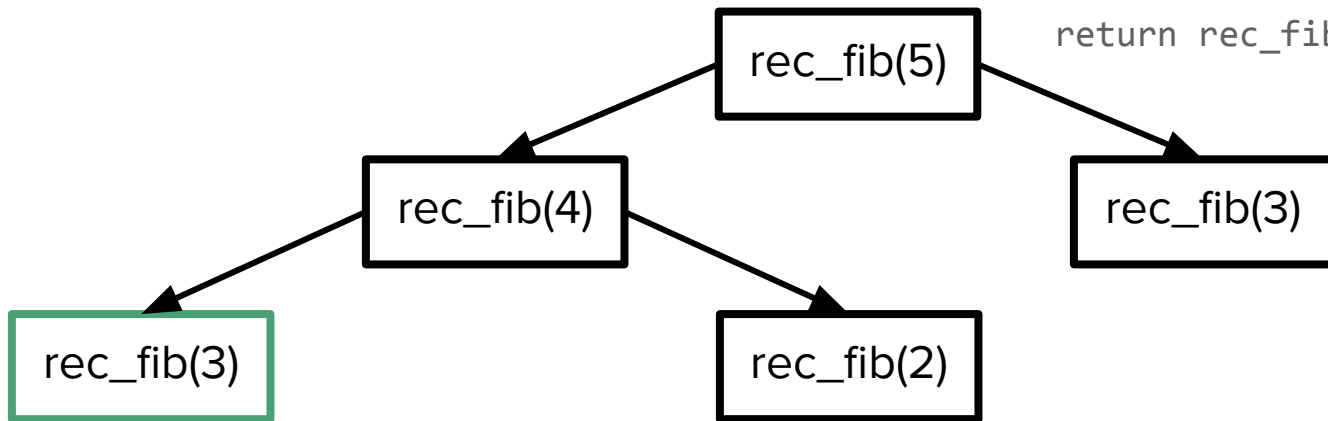
# Recursive Fibonacci: SLOW!!!

```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```



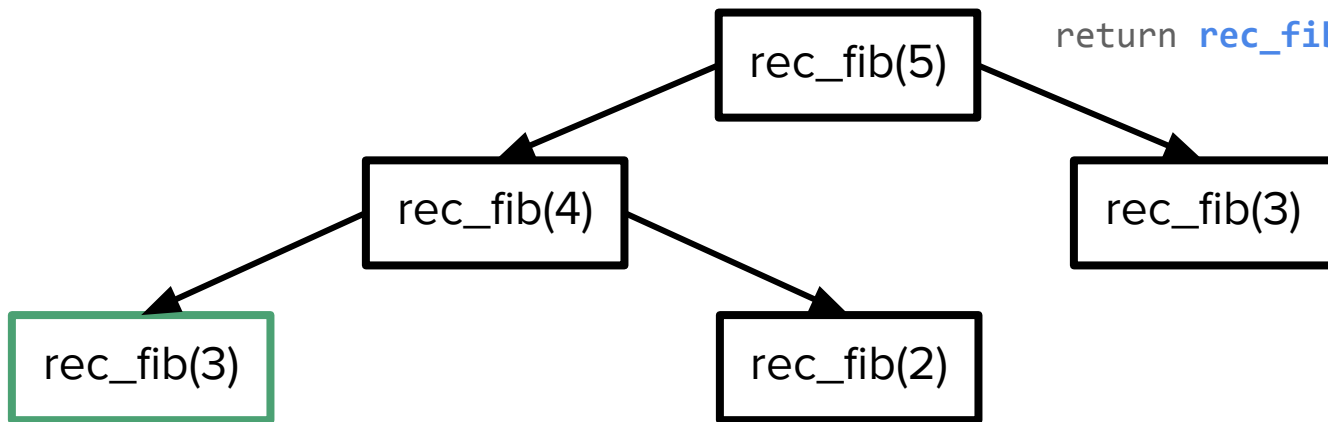
# Recursive Fibonacci: SLOW!!!

```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```



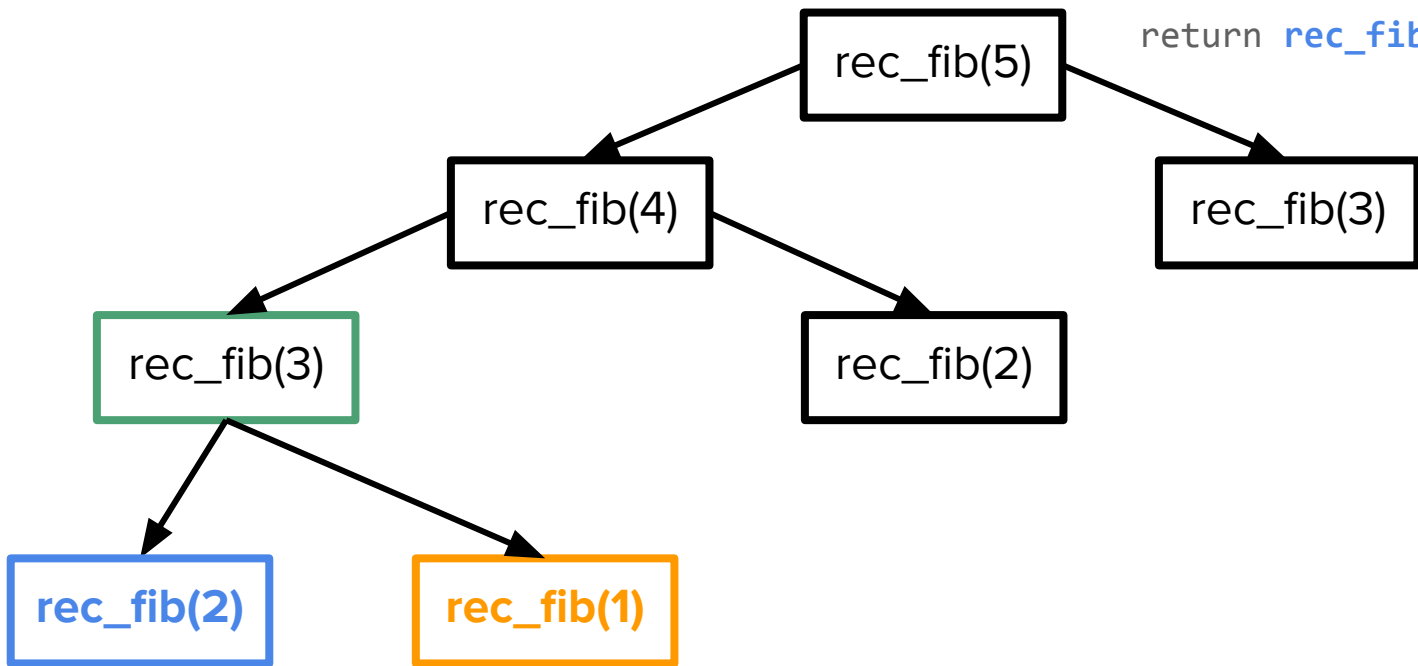
# Recursive Fibonacci: SLOW!!!

```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```



# Recursive Fibonacci: SLOW!!!

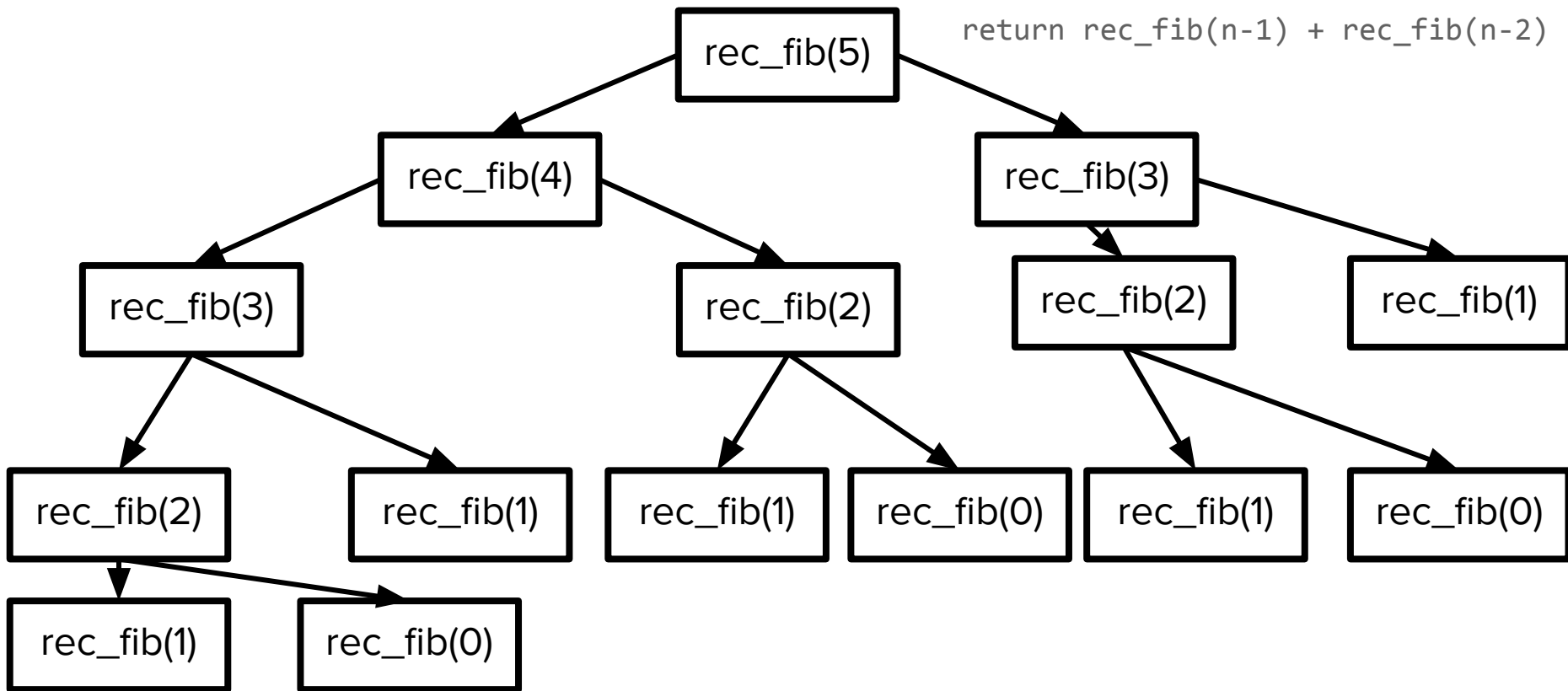
```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```





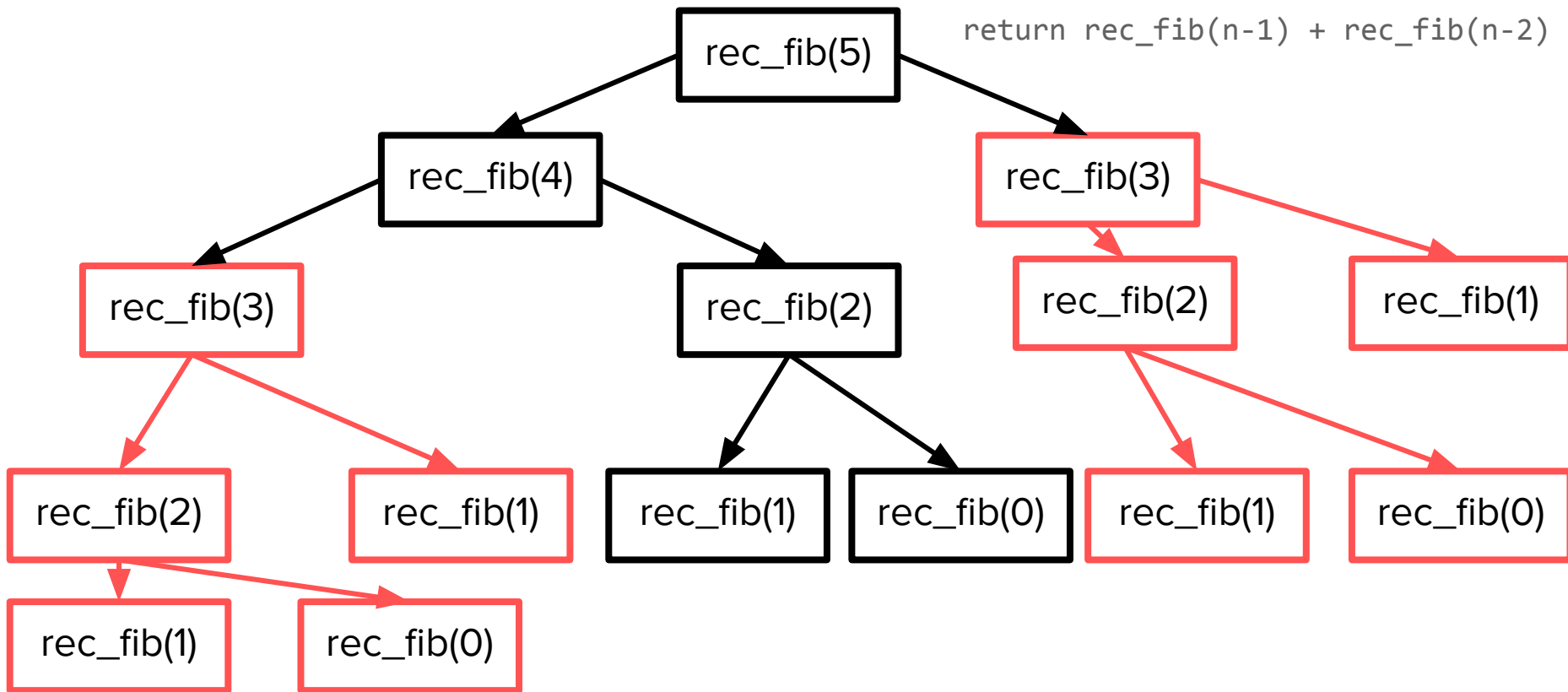
# Recursive Fibonacci: SLOW!!!

```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```



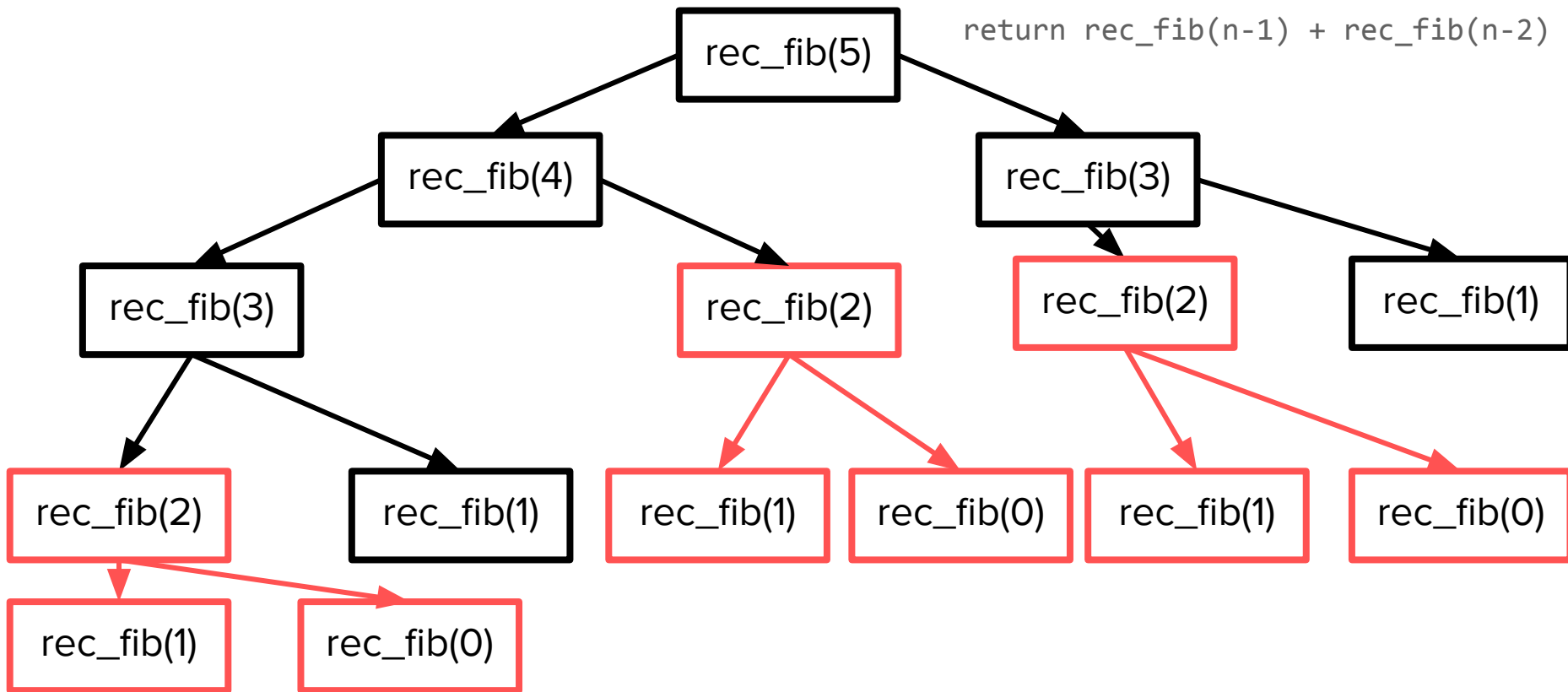
# Recursive Fibonacci: SLOW!!!

```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```



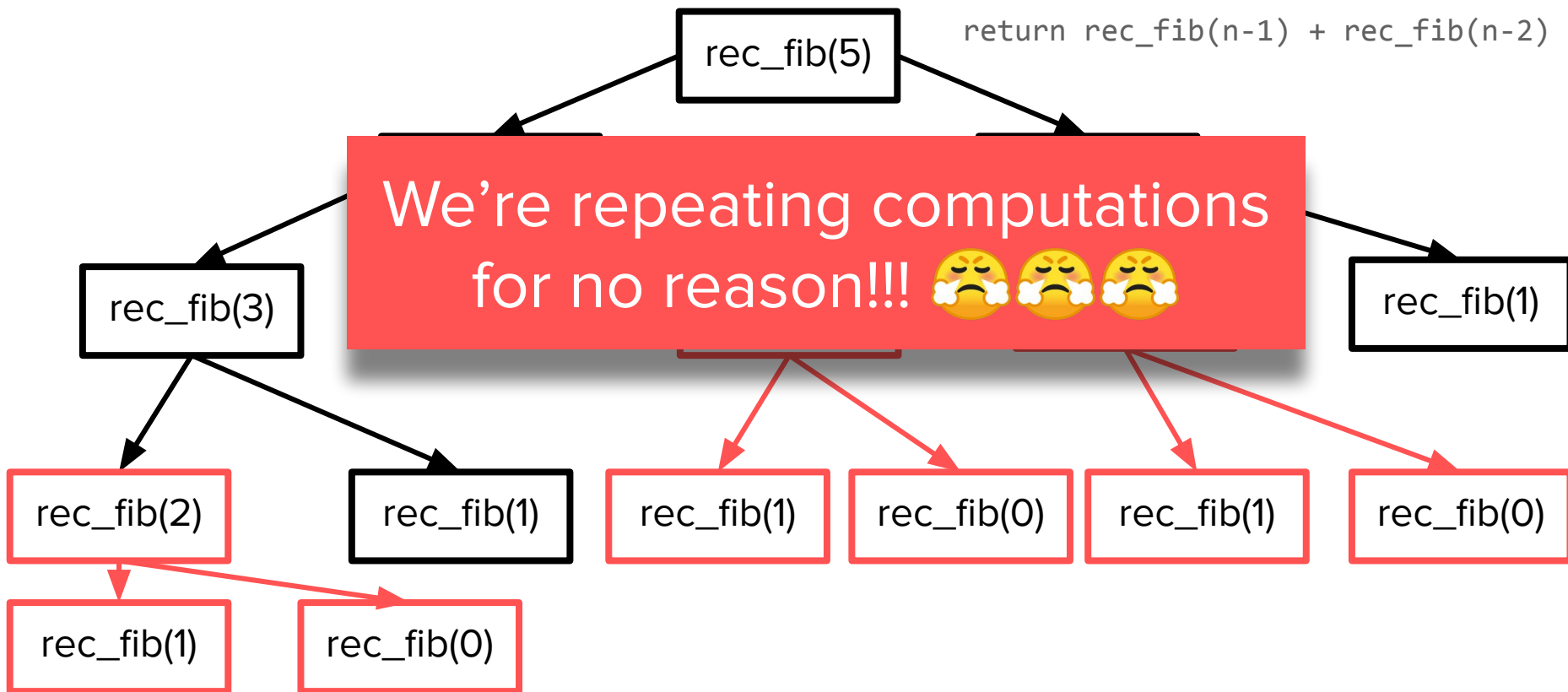
# Recursive Fibonacci: SLOW!!!

```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```



# Recursive Fibonacci: SLOW!!!

```
def fib_rec(n):  
    if n in {0,1}:  
        return n  
    return rec_fib(n-1) + rec_fib(n-2)
```



## 2 Fast 2 Fibonacci

- We saw that, in computing `rec_fib(5)`, we ended up calling `rec_fib(3)` twice and `rec_fib(2)` two times

## 2 Fast 2 Fibonacci

- We saw that, in computing `rec_fib(5)`, we ended up calling `rec_fib(3)` twice and `rec_fib(2)` two times
  - The values of `rec_fib(3)` and `rec_fib(2)` never change, so once we compute them the first time, why not just save them somewhere?

## 2 Fast 2 Fibonacci

- We saw that, in computing `rec_fib(5)`, we ended up calling `rec_fib(3)` twice and `rec_fib(2)` two times
  - The values of `rec_fib(3)` and `rec_fib(2)` never change, so once we compute them the first time, why not just save them somewhere?
- **Memoization:** Saving the results of expensive functions somewhere and returning the saved result when the same inputs occur again

# Fast Fib(e)

```
fast_fib(5)
```



# Fast Fib(e)

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

fast\_fib(5)

<i>n</i> :	0	1	2	3	4	5
fib( <i>n</i> ):						

# Fast Fib(e)

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

fast\_fib(5)

$n$ :	0	1	2	3	4	5
fib( $n$ ):	0					

# Fast Fib(e)

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

fast\_fib(5)

$n$ :	0	1	2	3	4	5
fib( $n$ ):	0	1				

# Fast Fib(e)

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

fast\_fib(5)

$n$ :	0	1	2	3	4	5
fib( $n$ ):	0	1	1			

# Fast Fib(e)

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

fast\_fib(5)

$n$ :	0	1	2	3	4	5
fib( $n$ ):	0	1	1	2		

# Fast Fib(e)

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

fast\_fib(5)

$n$ :	0	1	2	3	4	5
fib( $n$ ):	0	1	1	2	3	

# Fast Fib(e)

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

fast\_fib(5)

$n$ :	0	1	2	3	4	5
fib( $n$ ):	0	1	1	2	3	5

# Fast Fib(e)

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

fast\_fib(5)

$n$ :	0	1	2	3	4	5
fib( $n$ ):	0	1	1	2	3	5



# Dynamic Programming

# Dynamic Programming

1. **Define** the solution **recursively** from top-down

# Dynamic Programming

1. **Define** the solution **recursively** from top-down

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

# Dynamic Programming

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

1. **Define** the solution **recursively** from top-down
2. **Create** a **memoization table** to store the results

# Dynamic Programming

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

1. **Define** the solution **recursively** from top-down
2. **Create** a **memoization table** to store the results

$n$ :	0	1	2	3	4	5
$\text{fib}(n)$ :						

# Dynamic Programming

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

1. **Define** the solution **recursively** from top-down
2. **Create** a **memoization table** to store the results
3. **Fill** the **memoization table** from bottom-up

$n$ :	0	1	2	3	4	5
$\text{fib}(n)$ :	0					

# Dynamic Programming

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

1. **Define** the solution **recursively** from top-down
2. **Create** a **memoization table** to store the results
3. **Fill** the **memoization table** from bottom-up

$n$ :	0	1	2	3	4	5
fib( $n$ ):	0	1				

# Dynamic Programming

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

1. **Define** the solution **recursively** from top-down
2. **Create** a **memoization table** to store the results
3. **Fill** the **memoization table** from bottom-up

$n$ :	0	1	2	3	4	5
fib( $n$ ):	0	1	1			



# Dynamic Programming

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

1. **Define** the solution **recursively** from top-down
2. **Create** a **memoization table** to store the results
3. **Fill** the **memoization table** from bottom-up

$n$ :	0	1	2	3	4	5
fib( $n$ ):	0	1	1	2		

# Dynamic Programming

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

1. **Define** the solution **recursively** from top-down
2. **Create** a **memoization table** to store the results
3. **Fill** the **memoization table** from bottom-up

$n$ :	0	1	2	3	4	5
fib( $n$ ):	0	1	1	2	3	

# Dynamic Programming

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n > 1 \end{cases}$$

1. **Define** the solution **recursively** from top-down
2. **Create** a **memoization table** to store the results
3. **Fill** the **memoization table** from bottom-up

$n$ :	0	1	2	3	4	5
fib( $n$ ):	0	1	1	2	3	5

# Example: The 0-1 Knapsack Problem

- **Input:** A list of  $n$  items, where the  $i$ -th item has weight  $w_i$  and value  $v_i$ , and a capacity  $x$
- **Output:** A set of items such that the sum of their weights is below  $x$  and the sum of their values is **maximized**

## Example: The 0-1 Knapsack Problem

- Let  $P(i, j)$  denote the maximum possible profit of a selection of the first  $i$  items that have a total weight of at most  $j$

## Example: The 0-1 Knapsack Problem

- Let  $P(i, j)$  denote the maximum possible profit of a selection of the first  $i$  items that have a total weight of at most  $j$
- When we haven't chosen any items ( $i = 0$ ), the profit must be 0

## Example: The 0-1 Knapsack Problem

- Let  $P(i, j)$  denote the maximum possible profit of a selection of the first  $i$  items that have a total weight of at most  $j$
- When we haven't chosen any items ( $i = 0$ ), the profit must be 0
  - $P(0, j) = 0$  for all  $0 \leq j \leq x$

# Example: The 0-1 Knapsack Problem

- Let  $P(i, j)$  denote the maximum possible profit of a selection of the first  $i$  items that have a total weight of at most  $j$
- When we haven't chosen any items ( $i = 0$ ), the profit must be 0
  - $P(0, j) = 0$  for all  $0 \leq j \leq x$
- Negative weights ( $j < 0$ ) are impossible, so they should be disallowed



# Example: The 0-1 Knapsack Problem

- Let  $P(i, j)$  denote the maximum possible profit of a selection of the first  $i$  items that have a total weight of at most  $j$
- When we haven't chosen any items ( $i = 0$ ), the profit must be 0
  - $P(0, j) = 0$  for all  $0 \leq j \leq x$
- Negative weights ( $j < 0$ ) are impossible, so they should be disallowed
  - $P(i, j) = -\infty$  for all  $j < 0$

## Example: The 0-1 Knapsack Problem

- For all remaining  $1 \leq i \leq n$  and  $0 \leq j \leq x$ , we could either:

## Example: The 0-1 Knapsack Problem

- For all remaining  $1 \leq i \leq n$  and  $0 \leq j \leq x$ , we could either:
  - Add item  $i$  to our pack, so our profit would be  $v_i + P(i-1, j-w_i)$

# Example: The 0-1 Knapsack Problem

- For all remaining  $1 \leq i \leq n$  and  $0 \leq j \leq x$ , we could either:
  - Add item  $i$  to our pack, so our profit would be  $v_i + P(i-1, j-w_i)$ 
    - We get the value of item  $i$  ( $v_i$ ) added to the best scenario of looking at only the first  $i-1$  items with  $j-w_i$  total weight

# Example: The 0-1 Knapsack Problem

- For all remaining  $1 \leq i \leq n$  and  $0 \leq j \leq x$ , we could either:
  - Add item  $i$  to our pack, so our profit would be  $v_i + P(i-1, j-w_i)$ 
    - We get the value of item  $i$  ( $v_i$ ) added to the best scenario of looking at only the first  $i-1$  items with  $j-w_i$  total weight
  - Don't add item  $i$  to our pack, so our profit would be  $P(i-1, j)$

# Example: The 0-1 Knapsack Problem

- For all remaining  $1 \leq i \leq n$  and  $0 \leq j \leq x$ , we could either:
  - Add item  $i$  to our pack, so our profit would be  $v_i + P(i-1, j-w_i)$ 
    - We get the value of item  $i$  ( $v_i$ ) added to the best scenario of looking at only the first  $i-1$  items with  $j-w_i$  total weight
  - Don't add item  $i$  to our pack, so our profit would be  $P(i-1, j)$ 
    - No value added, so it's just our best scenario of looking at the first  $i-1$  items with the same ( $j$ ) total weight

# Example: The 0-1 Knapsack Problem

- For all remaining  $1 \leq i \leq n$  and  $0 \leq j \leq x$ , we could either:
  - Add item  $i$  to our pack, so our profit would be  $v_i + P(i-1, j-w_i)$ 
    - We get the value of item  $i$  ( $v_i$ ) added to the best scenario of looking at only the first  $i-1$  items with  $j-w_i$  total weight
  - Don't add item  $i$  to our pack, so our profit would be  $P(i-1, j)$ 
    - No value added, so it's just our best scenario of looking at the first  $i-1$  items with the same ( $j$ ) total weight
  - Take the **maximum** of these two possible options

## Example: The 0-1 Knapsack Problem

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3

8

10

6

$v$ :

\$5

\$5

\$13

\$6

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$$x = 20$$

***W:***

**$v$ :**

if  $i = 0$

if  $j < 0$

$$\text{if } 1 \leq i \leq n, 0 \leq j \leq x$$

**$j$  (total weight)**

 $i$  (number of items seen)

$x = 20$ 

$W:$

$V:$

**$j$  (total weight)**

[illegible]

$$x = 20$$

***W:***

0

[illegible]

$$x = 20$$

***W:***

**$v$ :**

 $i$  (number of items seen)

$$x = 20$$

***W:***

**V:**

 $i$  (number of items seen)

# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3	8	10	6
---	---	----	---

$v$ :

\$5	\$5	\$13	\$6
-----	-----	------	-----

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)

$i$ (number of items seen)		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	0	0	0	0	0	0	0	0									0	0	0	0	0	0
	1	?																				
	2																					
	3																					
	4																					

Add: \$5 +  $-\infty$

# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3	8	10	6
---	---	----	---

$v$ :

\$5	\$5	\$13	\$6
-----	-----	------	-----

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)

$i$ (number of items seen)		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	0	0	0	0	0	0	0	0									0	0	0	0	0	0
1	?																					
2																						
3																						
4																						

Add: \$5 +  $-\infty$

Don't Add: \$0



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3

8

10

6

$v$ :

\$5

\$5

\$13

\$6

3	8	10	6
\$5	\$5	\$13	\$6

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)

$i$ (number of items seen)		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0									0	0	0	0	0	0
1	0																					
2																						
3																						
4																						

Add: \$5 +  $-\infty$

Don't Add: \$0

$$x = 20$$

***W:***

**V:**

[illegible]

$$x = 20$$

***W:***

if  $i = 0$

if  $j < 0$

$$\text{if } 1 \leq i \leq n, 0 \leq j \leq x$$

**$j$  (total weight)**

 $i$  (number of items seen)

$$x = 20$$

***W:***

if  $i = 0$

if  $j < 0$

$$\text{if } 1 \leq i \leq n, 0 \leq j \leq x$$

**$j$  (total weight)**

 $i$  (number of items seen)

$$x = 20$$

***W:***

**V:**

if  $i = 0$

if  $j < 0$

$$\text{if } 1 \leq i \leq n, 0 \leq j \leq x$$

**$j$  (total weight)**

 $i$  (number of items seen)

$$x = 20$$

***W:***

**v:**

if  $i = 0$

if  $j < 0$

$$\text{if } 1 \leq i \leq n, 0 \leq j \leq x$$

**$j$  (total weight)**

 $i$  (number of items seen)

# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3	8	10	6
---	---	----	---

$v$ :

\$5	\$5	\$13	\$6
-----	-----	------	-----

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)

$i$ (number of items seen)		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	5																	
	2	0	0	0	5																	
	3	0	0	0	5																	
	4	0	0	0	5																	

# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3	8	10	6
---	---	----	---

$v$ :

\$5	\$5	\$13	\$6
-----	-----	------	-----

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)

$i$ (number of items seen)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	5	5																
2	0	0	0	5	5																
3	0	0	0	5	5																
4	0	0	0	5	5																



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3	8	10	6
---	---	----	---

$v$ :

\$5	\$5	\$13	\$6
-----	-----	------	-----

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)

$i$ (number of items seen)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	5	5	5															
2	0	0	0	5	5	5															
3	0	0	0	5	5	5															
4	0	0	0	5	5	5															

# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3

8

10

6

$v$ :

\$5

\$5

\$13

\$6

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	5	5	5	5														
2	0	0	0	5	5	5	5														
3	0	0	0	5	5	5	5														
4	0	0	0	5	5	5															

# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3

8

10

6

$v$ :

\$5

\$5

\$13

\$6

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	5	5	5	5														
2	0	0	0	5	5	5	5														
3	0	0	0	5	5	5	5														
4	0	0	0	5	5	5	6														

# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

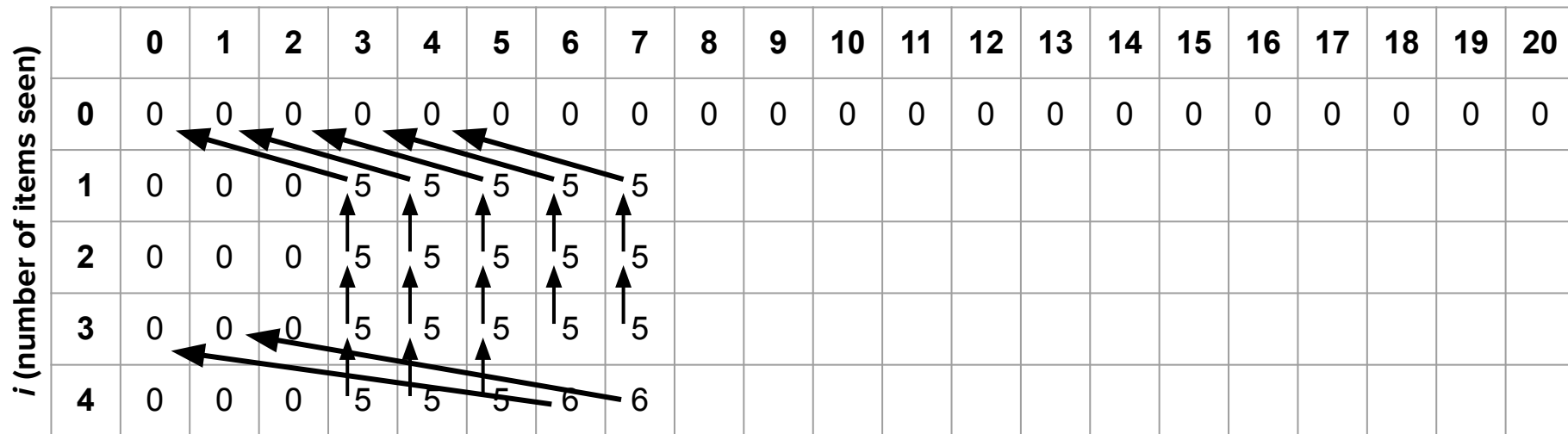
3	8	10	6
---	---	----	---

$v$ :

\$5	\$5	\$13	\$6
-----	-----	------	-----

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3

8

10

6

$v$ :

\$5

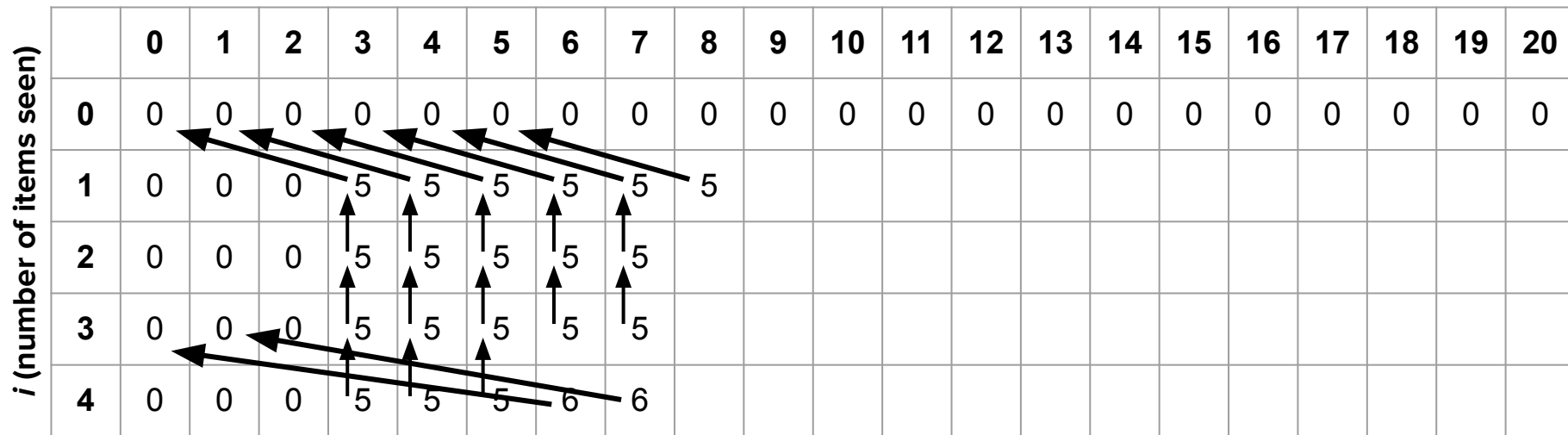
\$5

\$13

\$6

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

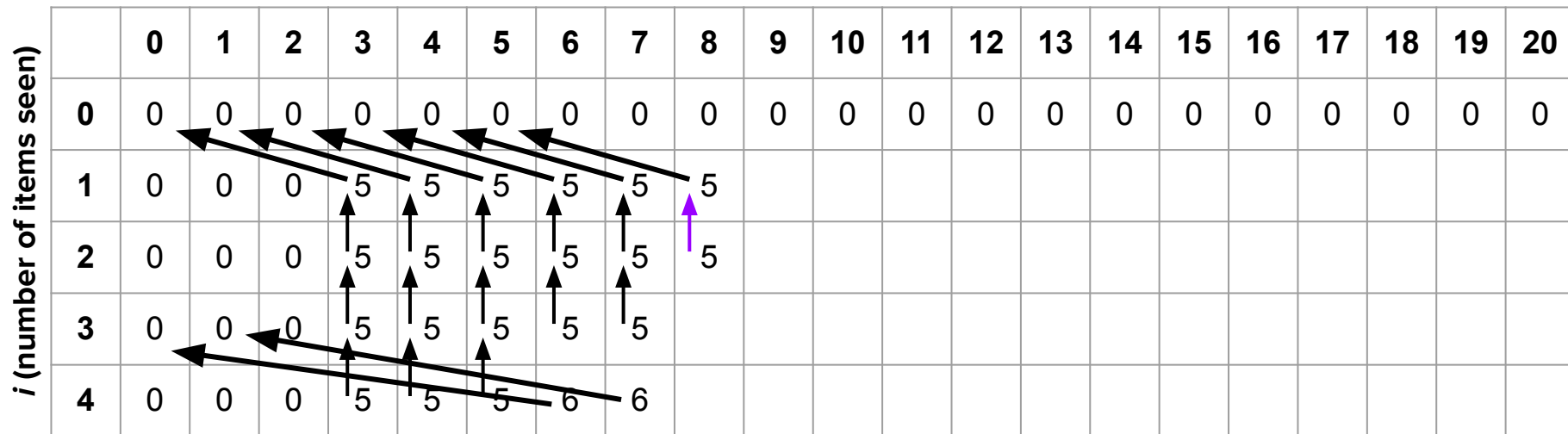
3	8	10	6
---	---	----	---

$v$ :

\$5	\$5	\$13	\$6
-----	-----	------	-----

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i-1, j-w_i), P(i-1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)



# Example: The 0-1 Knapsack Problem

$x = 20$

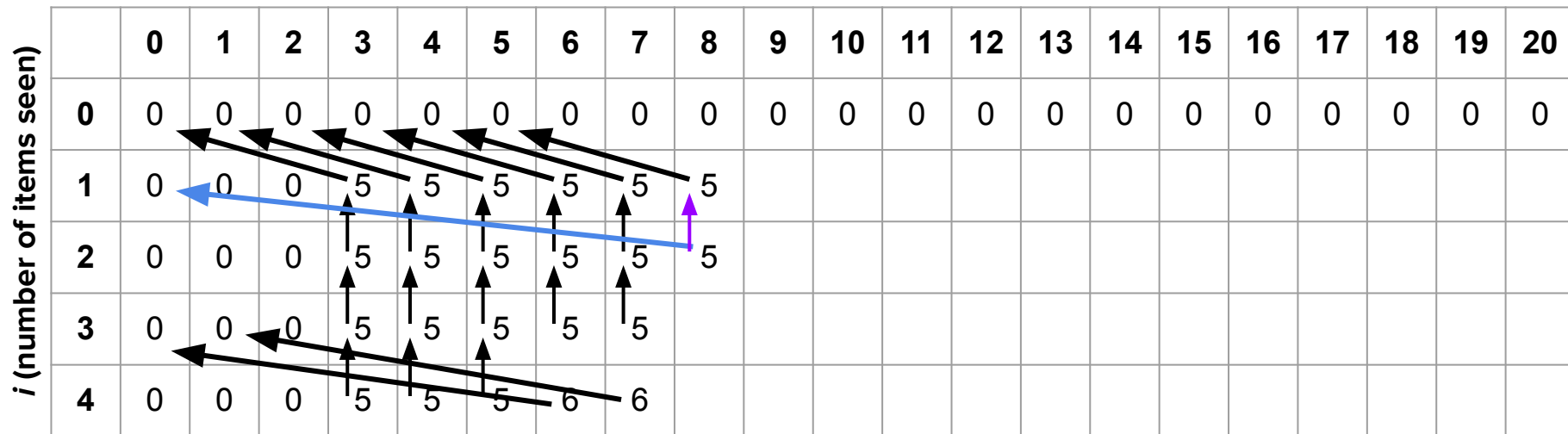
$w$ :

3	8	10	6
---	---	----	---

$v$ :

\$5	\$5	\$13	\$6
-----	-----	------	-----

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{ v_i + P(i-1, j-w_i), P(i-1, j) \} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$



# Example: The 0-1 Knapsack Problem

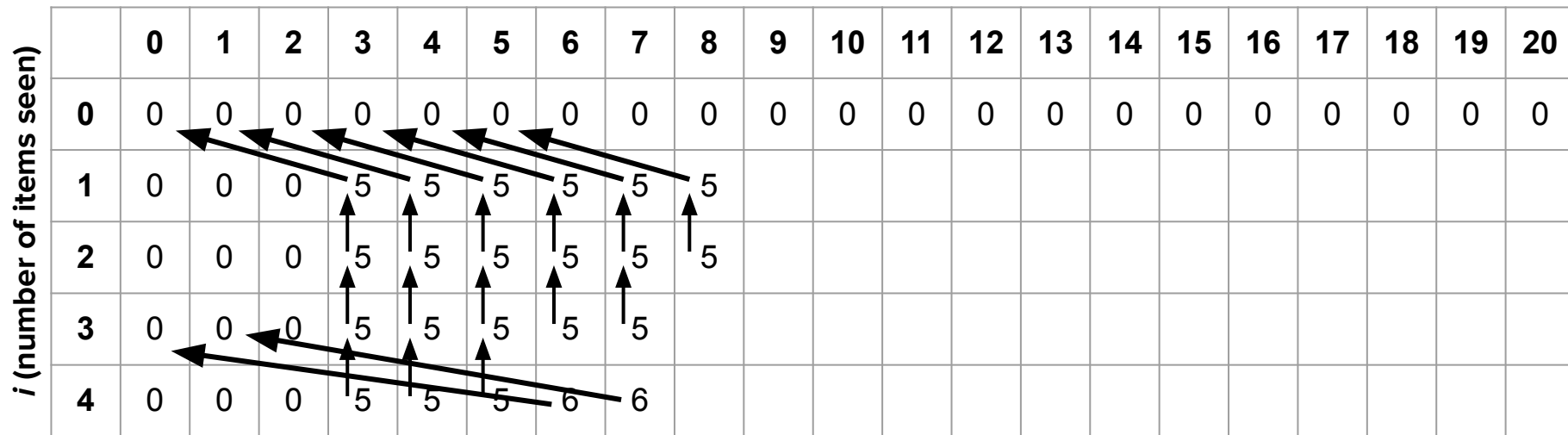
$x = 20$

$w$ :

3	8	10	6
\$5	\$5	\$13	\$6

Break ties arbitrarily

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i-1, j-w_i), P(i-1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$





# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3

8

10

6

$v$ :

\$5

\$5

\$13

\$6

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	5	5	5	5	5	5												
2	0	0	0	5	5	5	5	5	5												
3	0	0	0	5	5	5	5	5	5												
4	0	0	0	5	5	5	6	6	6												





$x = 20$ 

***W:***

**V:**

if  $i = 0$

if  $j < 0$

$$\text{if } 1 \leq i \leq n, 0 \leq j \leq x$$



$$x = 20$$

***W:***

**V:**

if  $i = 0$

if  $j < 0$

$$\text{if } 1 \leq i \leq n, 0 \leq j \leq x$$



$$x = 20$$

***W:***

**V:**

if  $i = 0$

if  $j < 0$

$$\text{if } 1 \leq i \leq n, 0 \leq j \leq x$$



$$x = 20$$

***W:***

**V:**

if  $i = 0$

if  $j < 0$

$$\text{if } 1 \leq i \leq n, 0 \leq j \leq x$$



$$x = 20$$

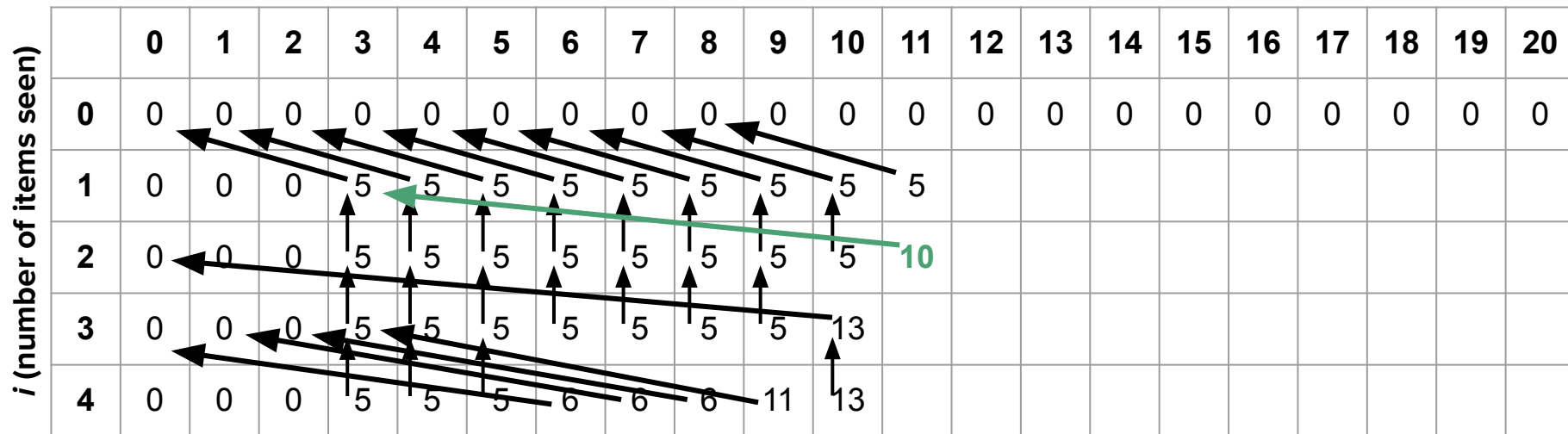
**W:**

 $V:$ 

if  $i = 0$

if  $j < 0$

$$\text{if } 1 \leq i \leq n, 0 \leq j \leq x$$





$$x = 20$$

***W:***

**$v$ :**

if  $i = 0$

if  $j < 0$

$$\text{if } 1 \leq i \leq n, 0 \leq j \leq x$$



$$x = 20$$

***W:***

**$v$ :**

if  $i = 0$

if  $j < 0$

$$\text{if } 1 \leq i \leq n, 0 \leq j \leq x$$



$$x = 20$$

***W:***

***V:***

if  $i = 0$

if  $j < 0$

$$\text{if } 1 \leq i \leq n, 0 \leq j \leq x$$



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

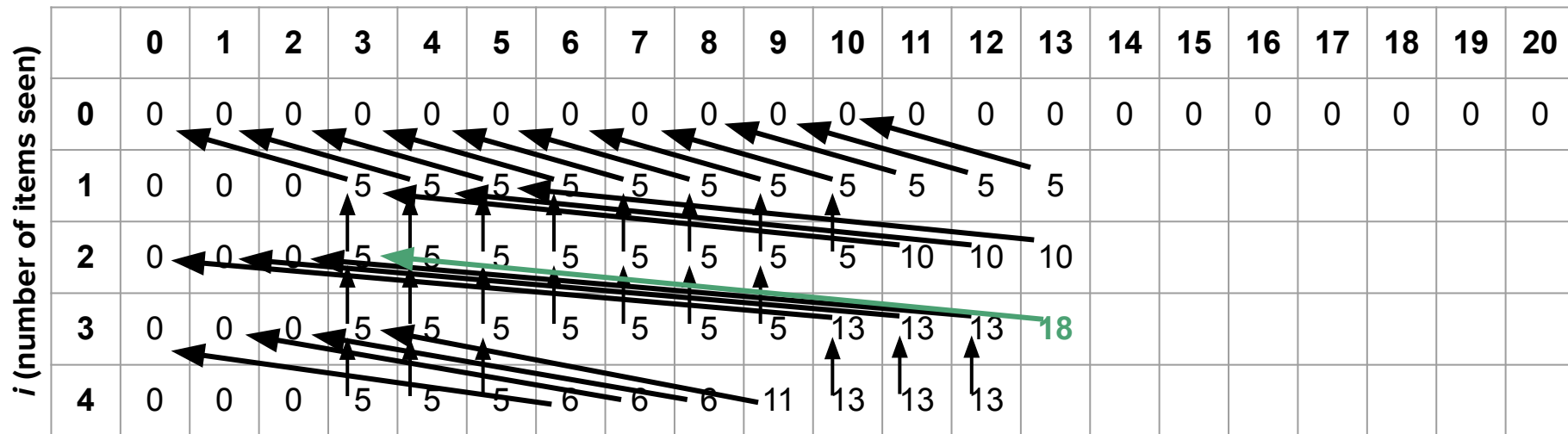
3	8	10	6
---	---	----	---

$v$ :

\$5	\$5	\$13	\$6
-----	-----	------	-----

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i-1, j-w_i), P(i-1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3

8

10

6

$v$ :

\$5

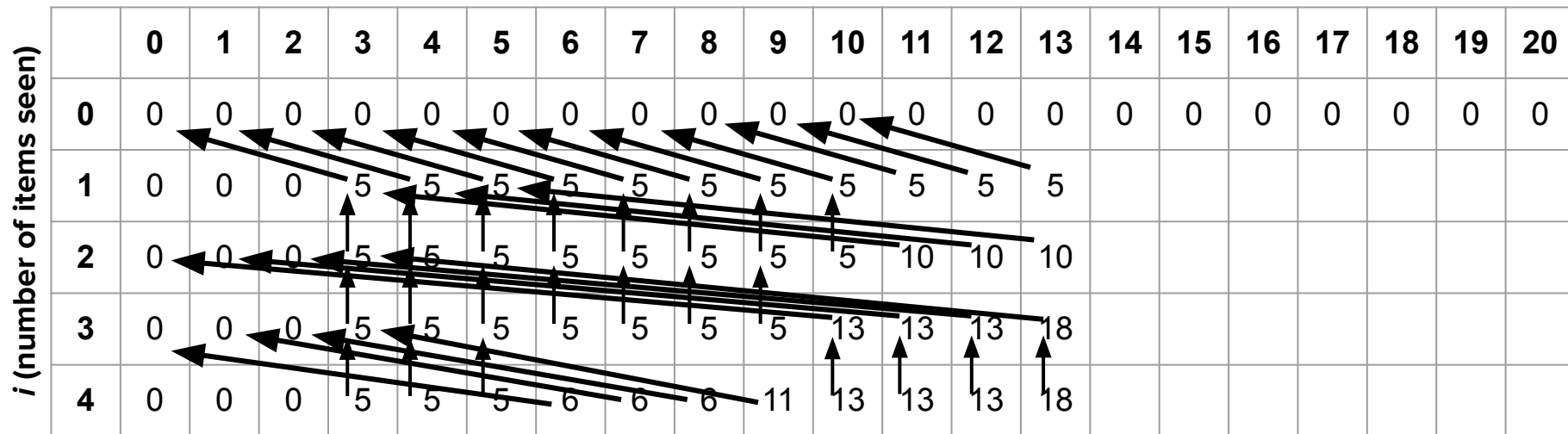
\$5

\$13

\$6

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i-1, j-w_i), P(i-1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3

8

10

6

$v$ :

\$5

\$5

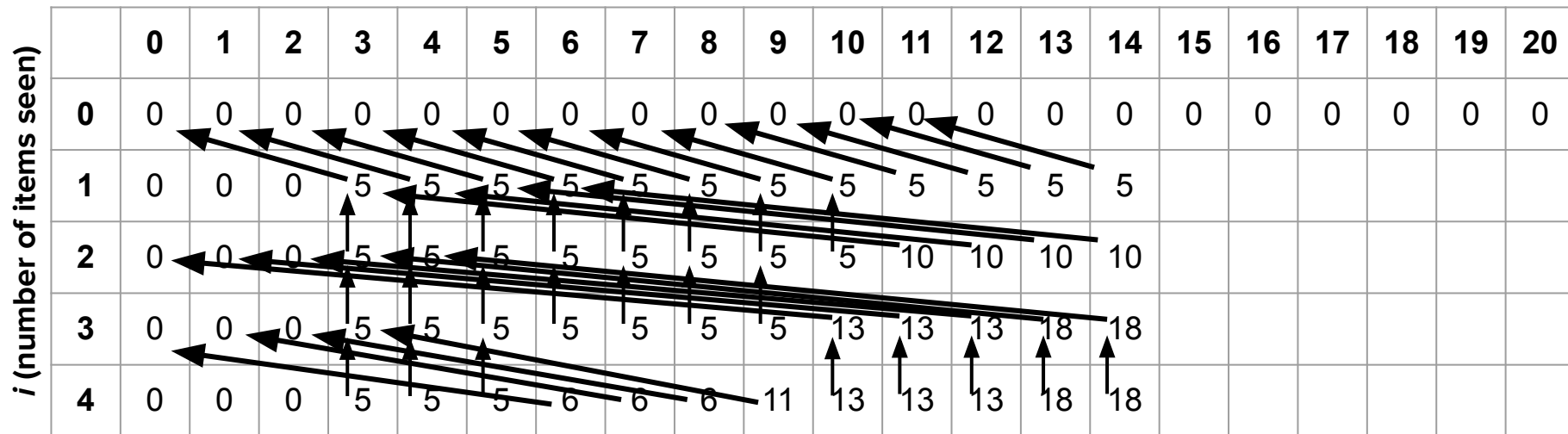
\$13

\$6

$w$ :	3	8	10	6
$v$ :	\$5	\$5	\$13	\$6

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3

8

10

6

$v$ :

\$5

\$5

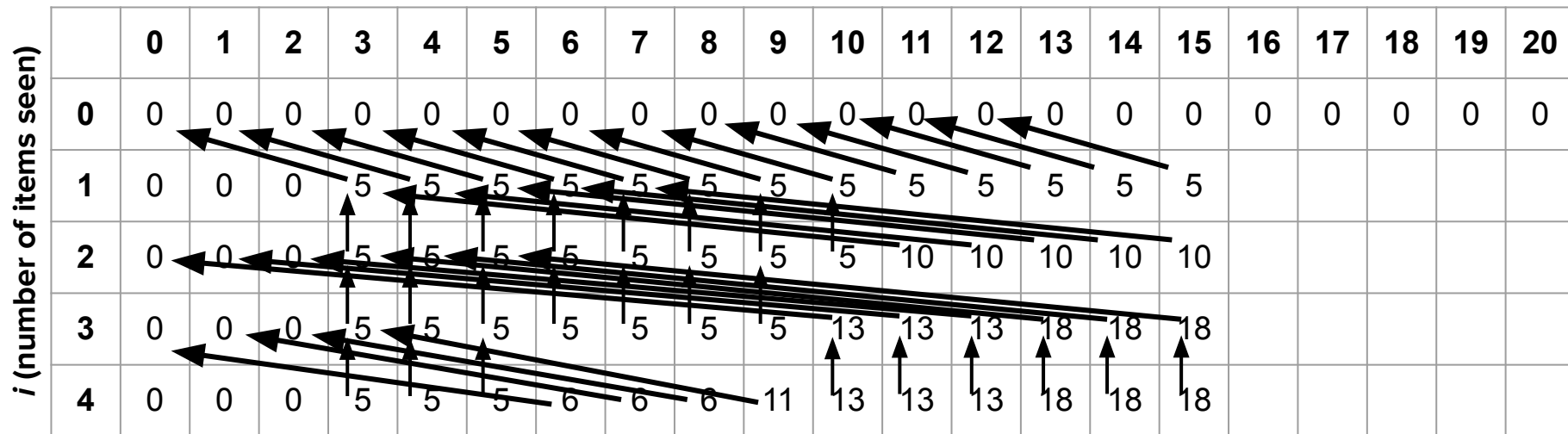
\$13

\$6

	3	8	10	6
	\$5	\$5	\$13	\$6

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i-1, j-w_i), P(i-1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3

8

10

6

$v$ :

\$5

\$5

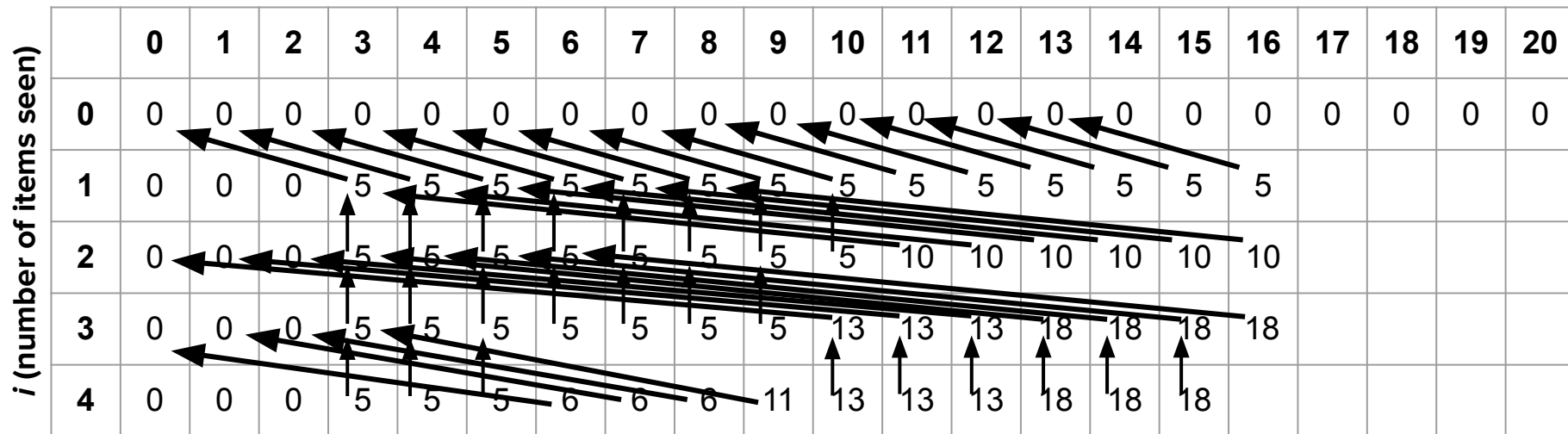
\$13

\$6

if  $i = 0$

if  $j < 0$

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$





# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3

8

10

6

$v$ :

\$5

\$5

\$13

\$6

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3

8

10

6

$v$ :

\$5

\$5

\$13

\$6

$w$ :	3	8	10	6
$v$ :	\$5	\$5	\$13	\$6

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3	8	10	6
---	---	----	---

$v$ :

\$5	\$5	\$13	\$6
-----	-----	------	-----

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3

8

10

6

$v$ :

\$5

\$5

\$13

\$6

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

3

8

10

6

$v$ :

\$5

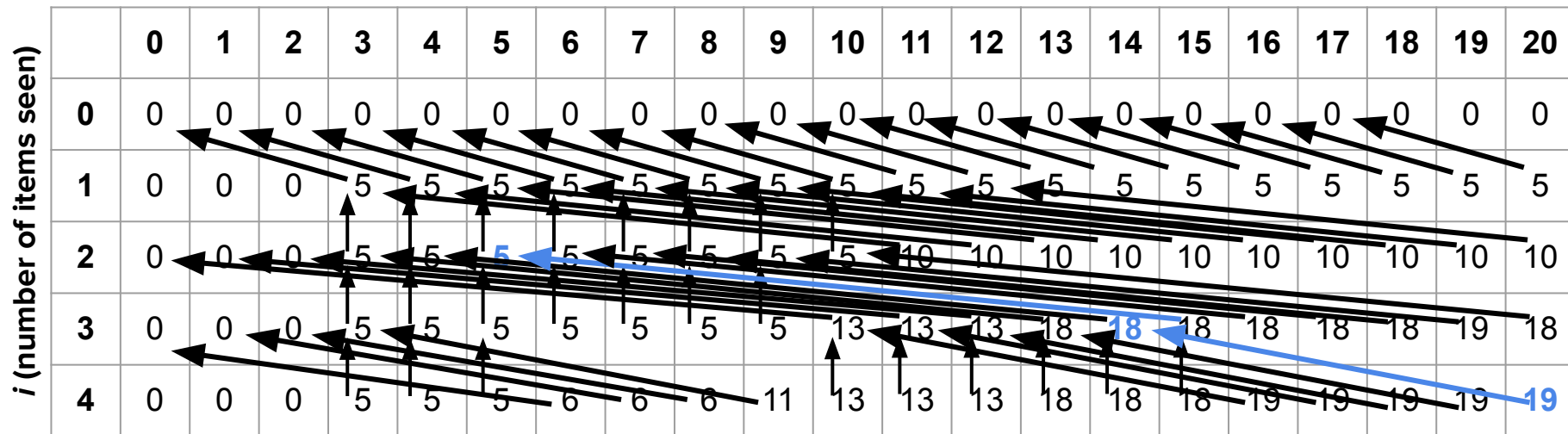
\$5

\$13

\$6

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :

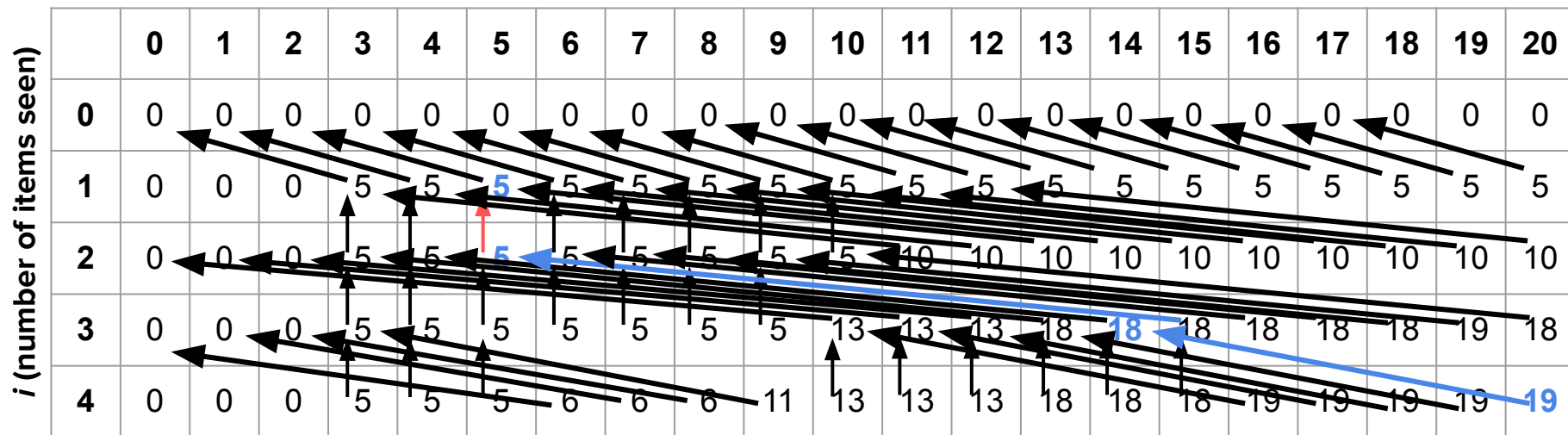
3	8	10	6
---	---	----	---

$v$ :

\$5	\$5	\$13	\$6
-----	-----	------	-----

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i - 1, j - w_i), P(i - 1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)



# Example: The 0-1 Knapsack Problem

$x = 20$

$w$ :	3	8	10	6
$v$ :	\$5	\$5	\$13	\$6

$$P(i, j) = \begin{cases} 0 & \text{if } i = 0 \\ -\infty & \text{if } j < 0 \\ \max \{v_i + P(i-1, j-w_i), P(i-1, j)\} & \text{if } 1 \leq i \leq n, 0 \leq j \leq x \end{cases}$$

$j$  (total weight)

