CSI 2103: Data Structures

HW Assignment 2

Yonsei University
Spring 2022

Seong Jae Hwang

Array and Doubly Linked List

- Aims:
  - Study two data structures: array and doubly linked list (DDL)
  - See how a digital image can be stored in such data structures
  - Implement algorithms that transform the images by manipulating the data structures
- Two parts:
  - Part 1: Toy example of converting a 2D array into 2D doubly linked list
  - Part 2: Resizing image

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

Simple Node Class

```
class Node:
    """ Node class
            up
             |
    left - data - right
             |
           down
    """

    def __init__(self, data):
        self.data = data
        self.right = None
        self.left = None
        self.up = None
        self.down = None
```

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

Image as a 2D array

0	0	0	0	255	0
100	100	100	0	255	0
100	100	100	0	255	0
100	100	100	0	255	0
0	0	0	0	255	0

image: np.ndarray (numpy n-dimensional array)
image_matrix.shape[0] shows the "height" (5 in above example)
image_matrix.shape[1] shows the "width" (6 in above example)

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

Part 1

- In this simple toy example, you will convert a 2D array to a 2D doubly linked list. Several functions have to be implemented at this stage:
  - constructDoublyLinkedListRecursiveStep
  - constructDoublyLinkedListLoop

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

2D Array (Matrix) with numpy.ndarray

arr: numpy.ndarray
arr.shape[0]: first dimension size ("height")
arr.shape[1]: second dimension size ("width")
arr[y][x]: element at row "y" and column "x"

		column			
		x = 0	x = 1	x = 2	x = 3
row	y = 0	5	4	7	10
	y = 1	6	12	1	9
	y = 2	8	2	3	11

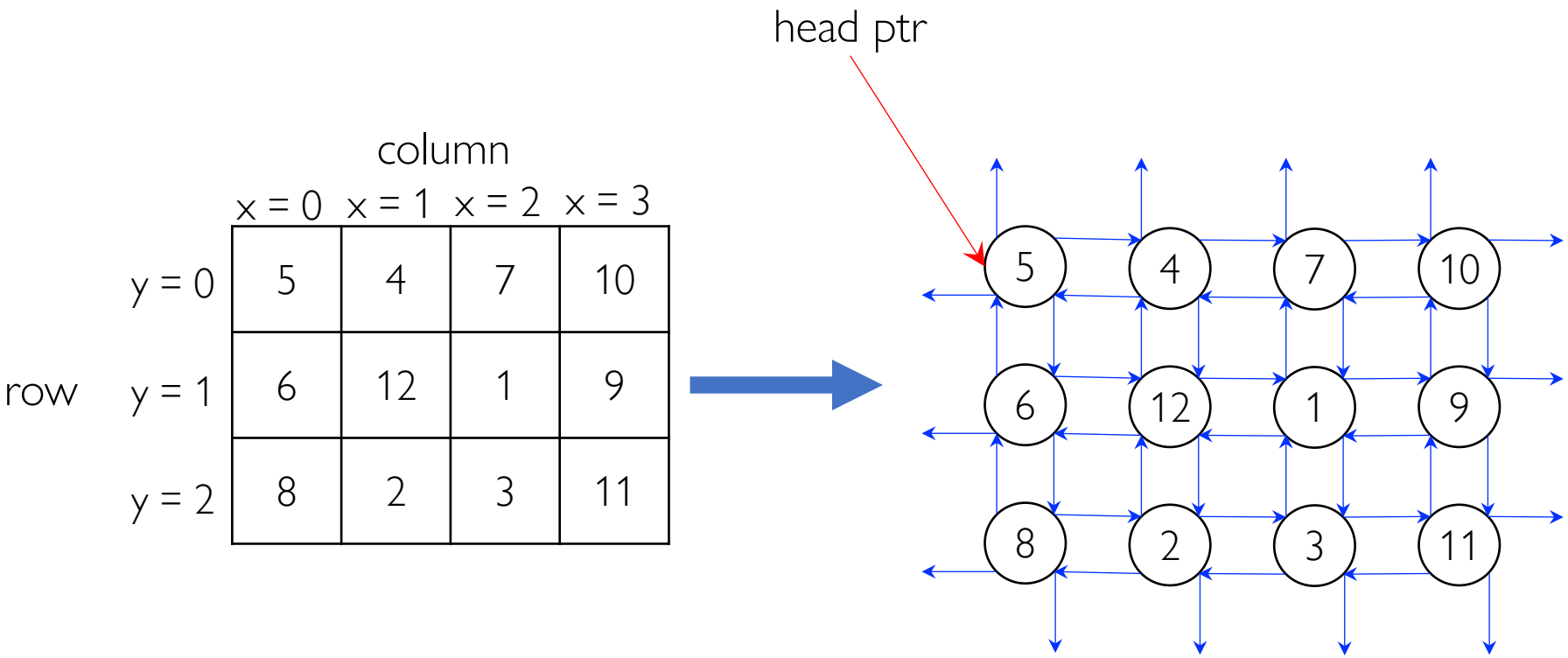
```
print(arr.shape[0])     # 3
print(arr.shape[1])     # 4
print(arr[0][0])        # 5
print(arr[2][3])        # 1
print(arr[2,3])         # also 1
```

Randomly generated 3x3 2D array

```
>> N = 3
>> arr = np.random.randint(0, 10, size=(N, N))
>> print(arr)
[[5  4  7]
 [9  6  3]
 [7  6  9]]

>> print(type(arr))
<class 'numpy.ndarray'>
```

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are

2D Array to 2D Doubly Linked List



dll_rec = constructDoublyLinkedListRecursion(arr)
dll_loop = constructDoublyLinkedListLoop(arr)

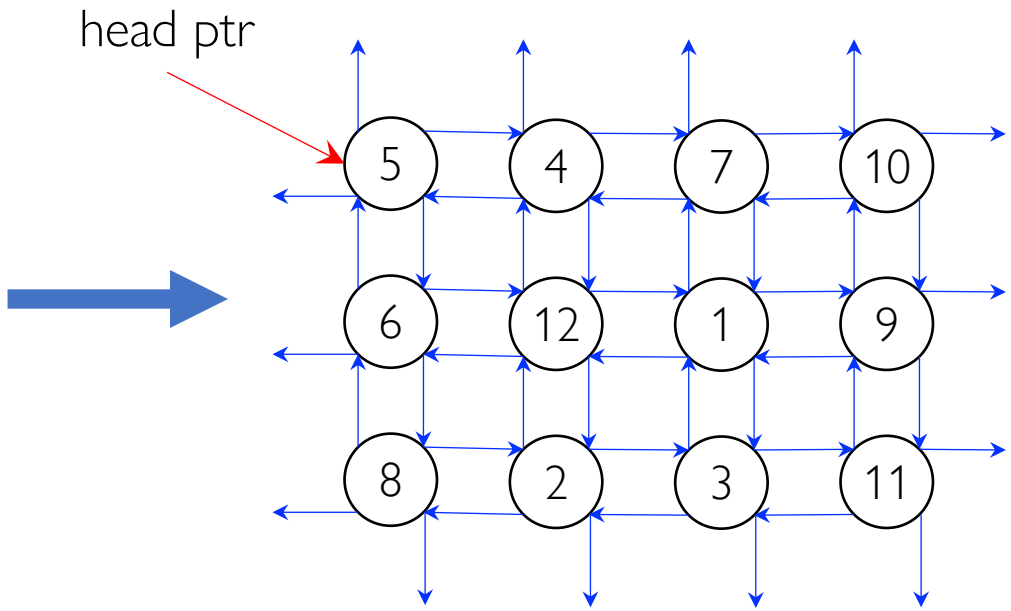
This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDoublyLinkedListLoop

- input: arr (2D array to turn into a 2D DLL)
- output: top_left_ptr (head pointer to the top left node of the 2D DLL)

column

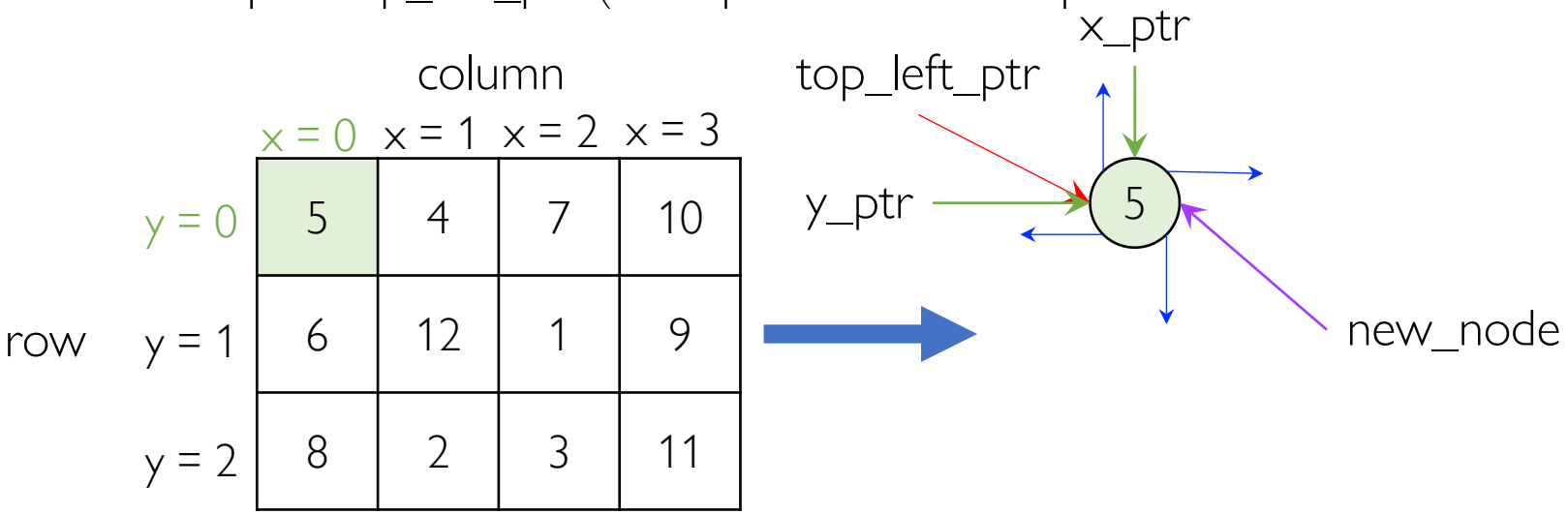
row	x = 0	x = 1	x = 2	x = 3
y = 0	5	4	7	10
y = 1	6	12	1	9
y = 2	8	2	3	11



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDoublyLinkedListLoop

- input: arr (2D array to turn into a 2D DLL)
- output: top_left_ptr (head pointer to the top left node of the 2D DLL)

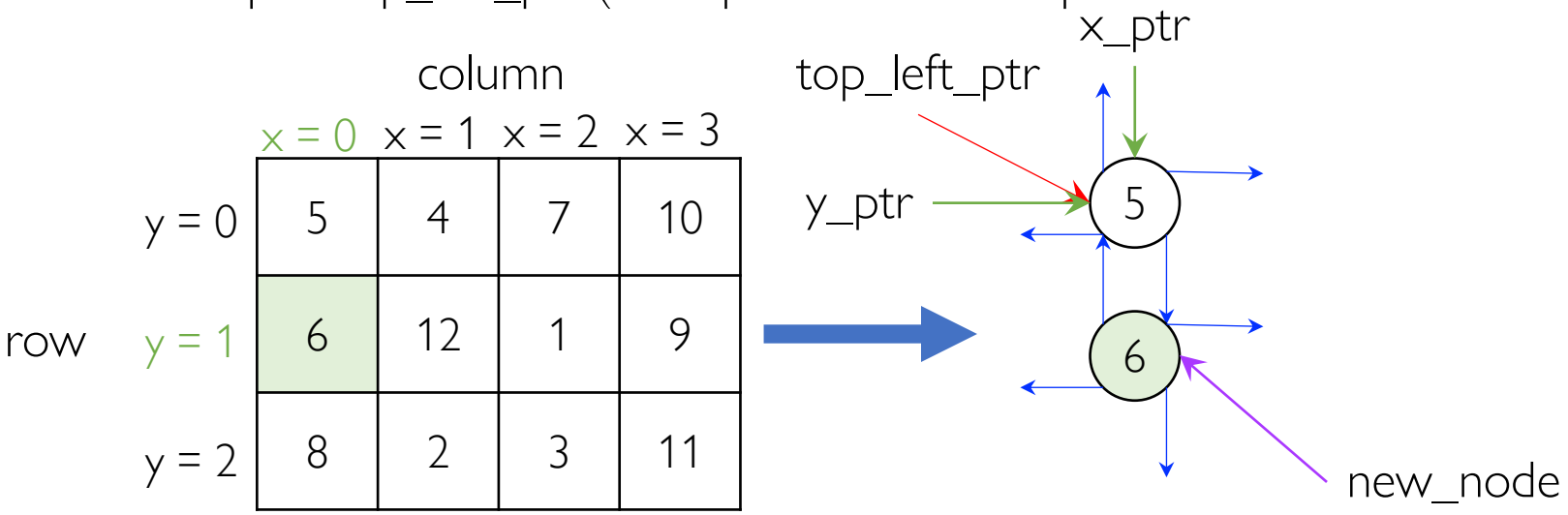


new_node = Node(arr[y,x])

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDoublyLinkedListLoop

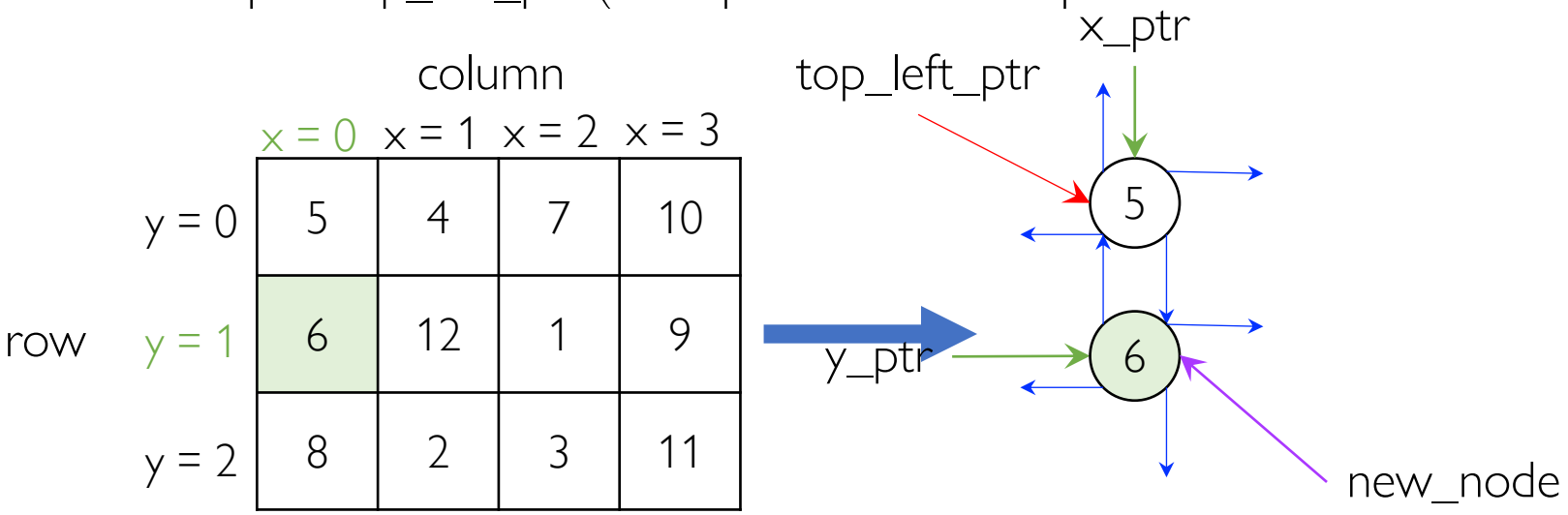
- input: arr (2D array to turn into a 2D DLL)
- output: top_left_ptr (head pointer to the top left node of the 2D DLL)



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDoublyLinkedListLoop

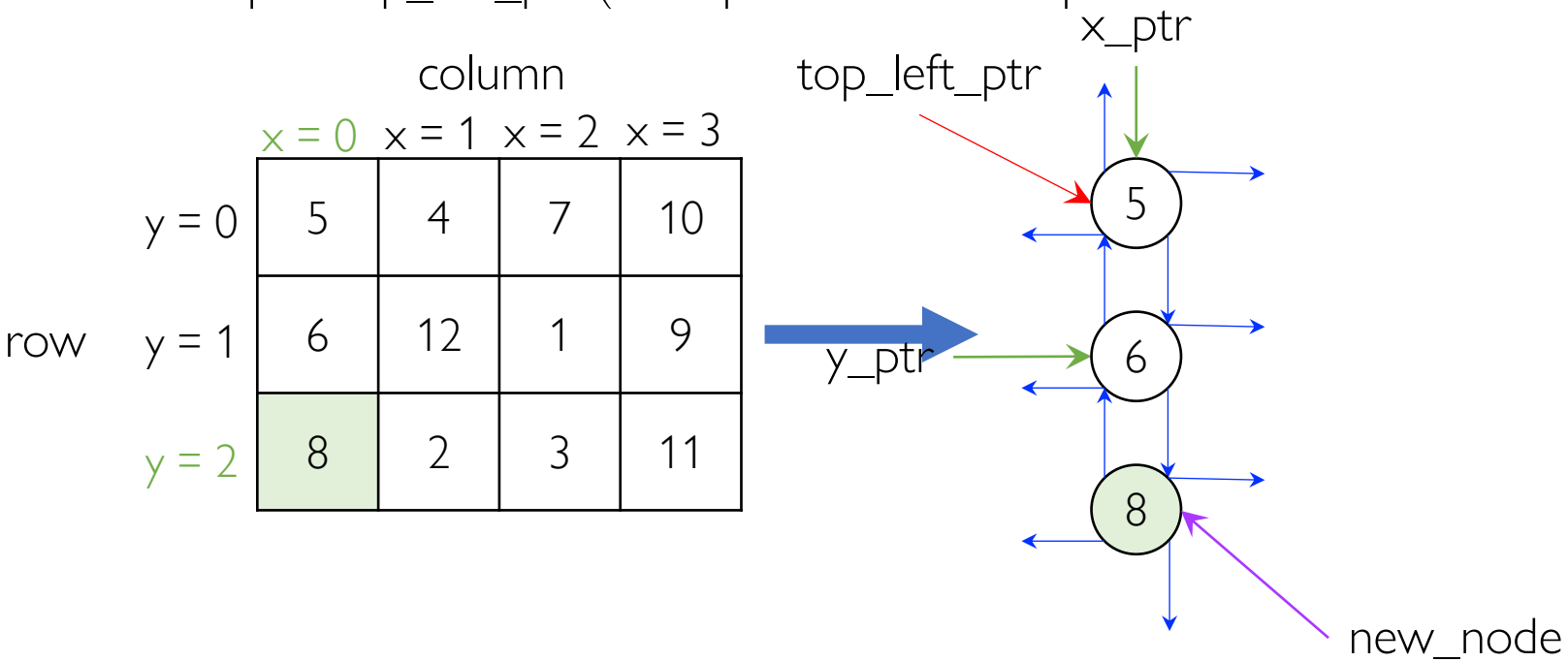
- input: arr (2D array to turn into a 2D DLL)
- output: top_left_ptr (head pointer to the top left node of the 2D DLL)



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDoublyLinkedListLoop

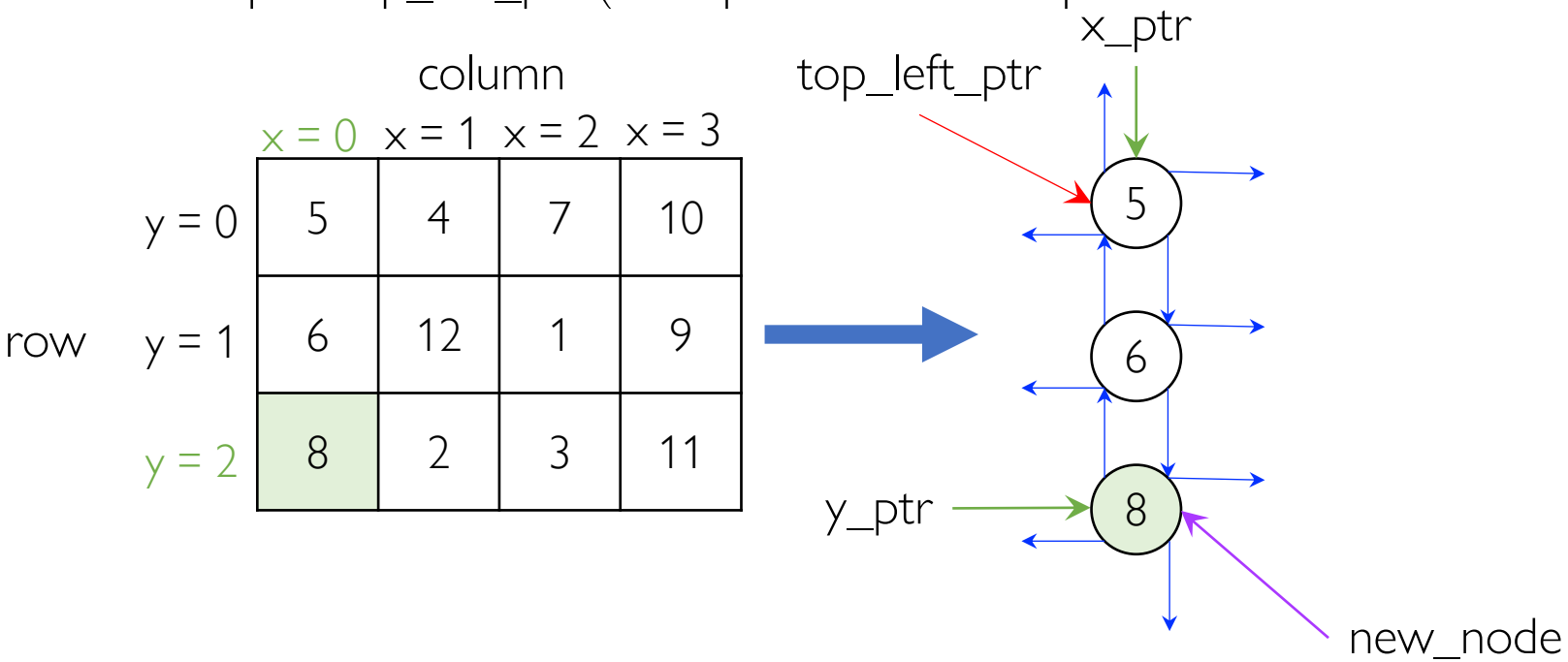
- input: arr (2D array to turn into a 2D DLL)
- output: top_left_ptr (head pointer to the top left node of the 2D DLL)



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDoublyLinkedListLoop

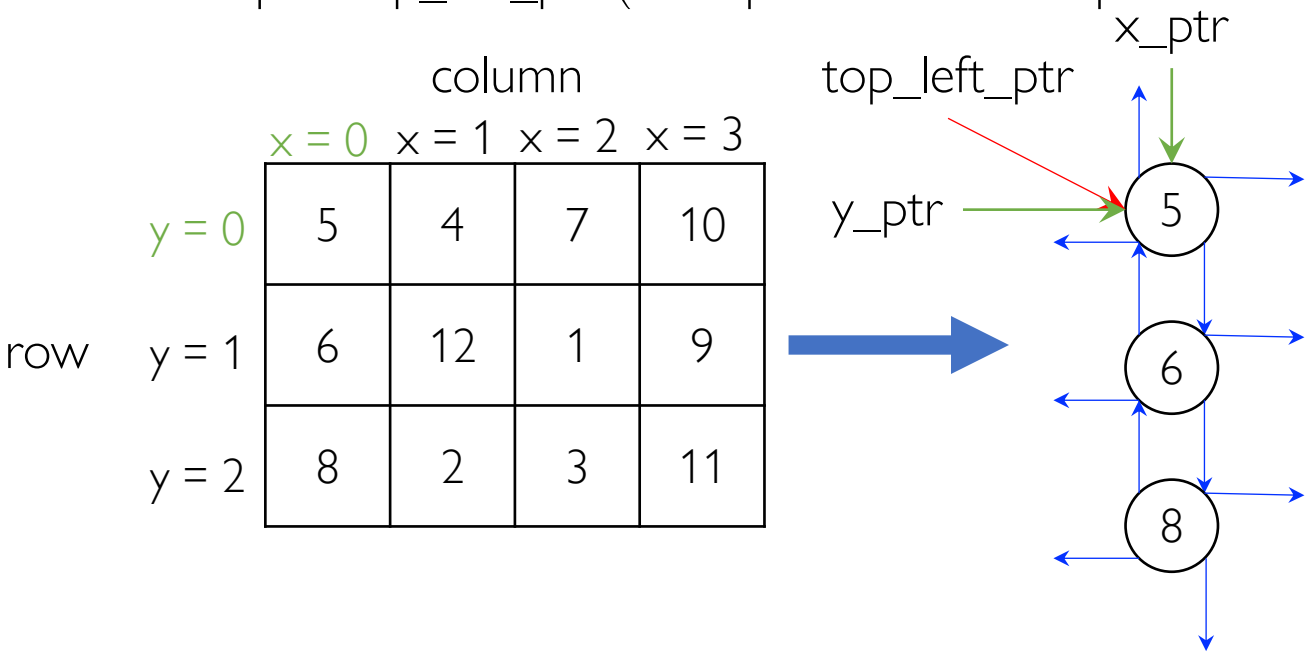
- input: arr (2D array to turn into a 2D DLL)
- output: top_left_ptr (head pointer to the top left node of the 2D DLL)



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms are legally responsible for any violation of intellectual property laws.

constructDoublyLinkedListLoop

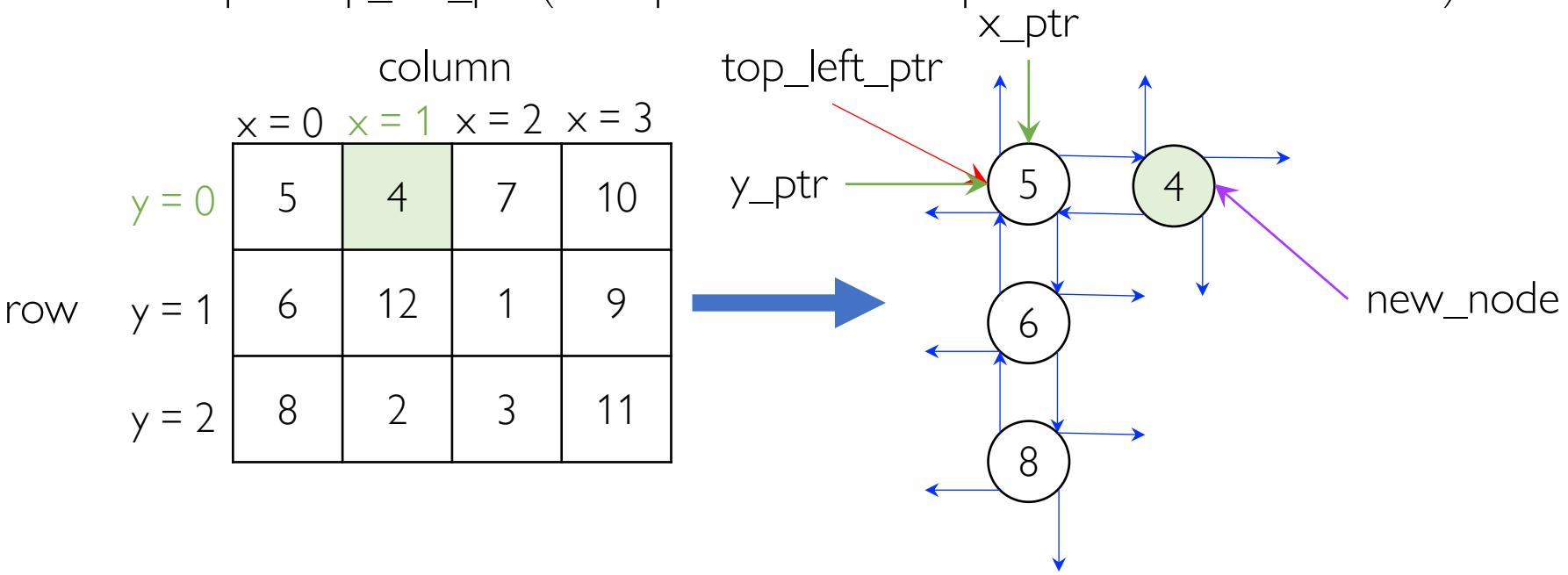
- input: arr (2D array to turn into a 2D DLL)
- output: top_left_ptr (head pointer to the top left node of the 2D DLL)



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDoublyLinkedListLoop

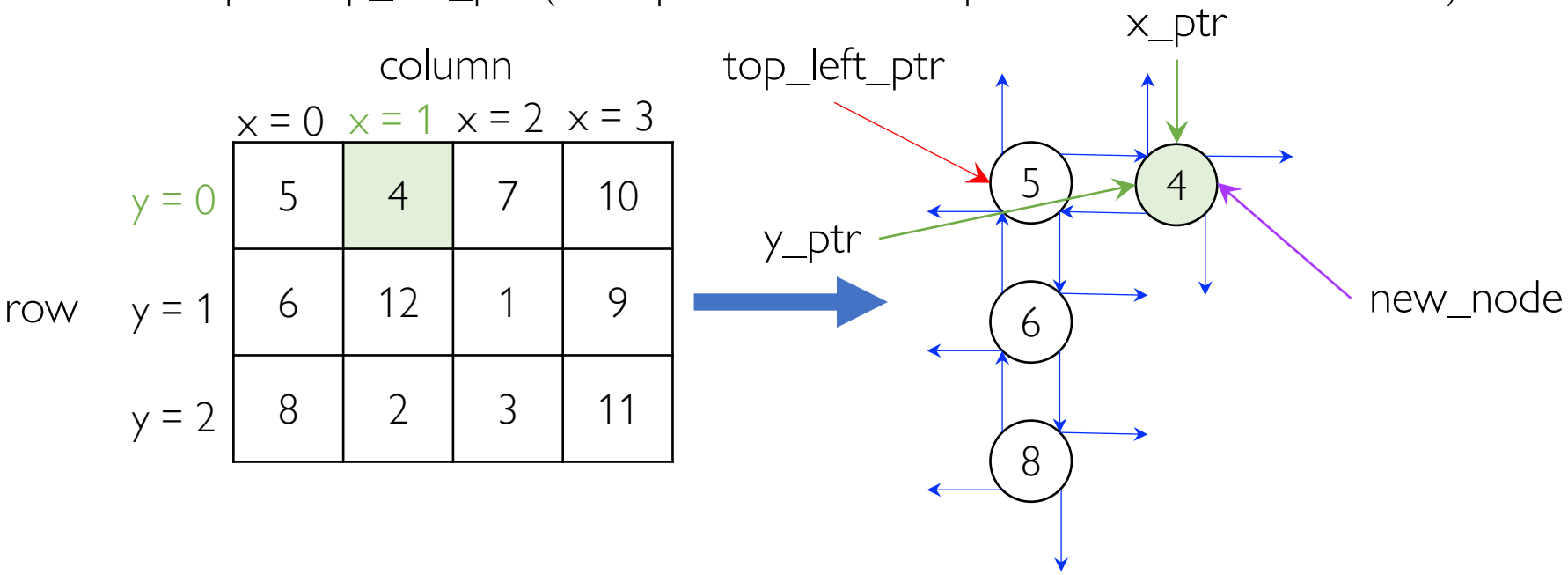
- input: arr (2D array to turn into a 2D DLL)
- output: top_left_ptr (head pointer to the top left node of the 2D DLL)



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDoublyLinkedListLoop

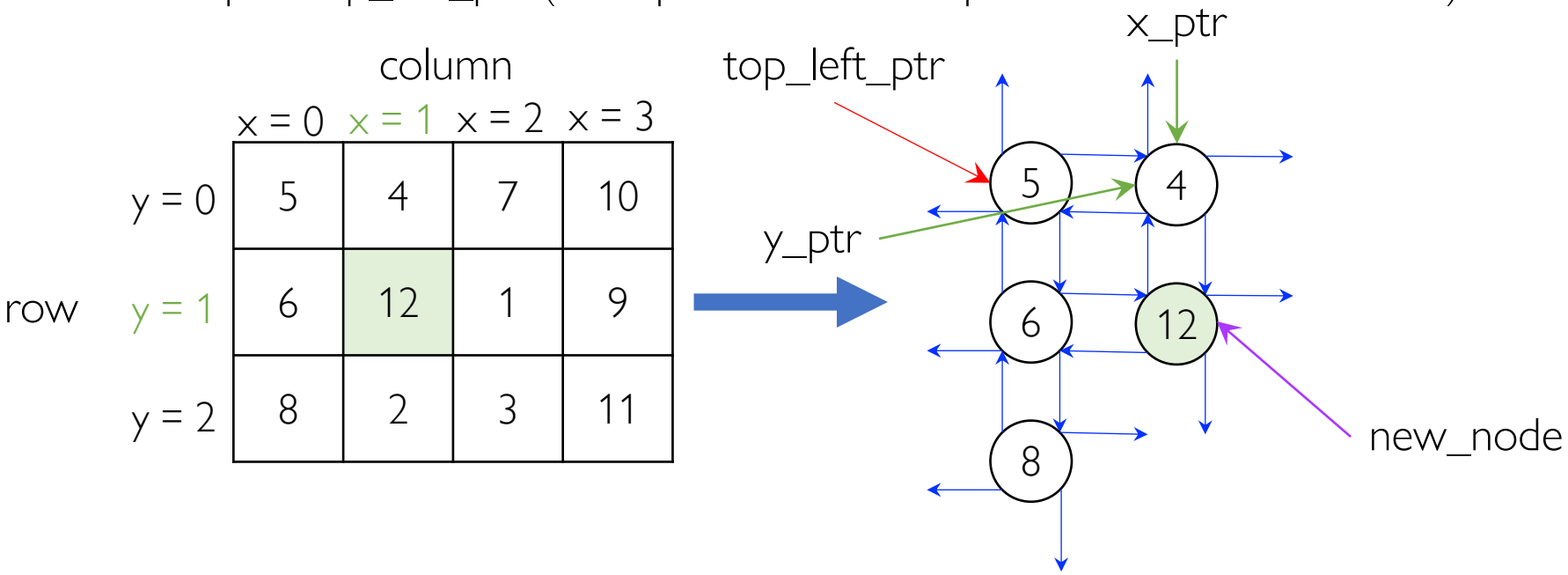
- input: arr (2D array to turn into a 2D DLL)
- output: top_left_ptr (head pointer to the top left node of the 2D DLL)



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDoublyLinkedListLoop

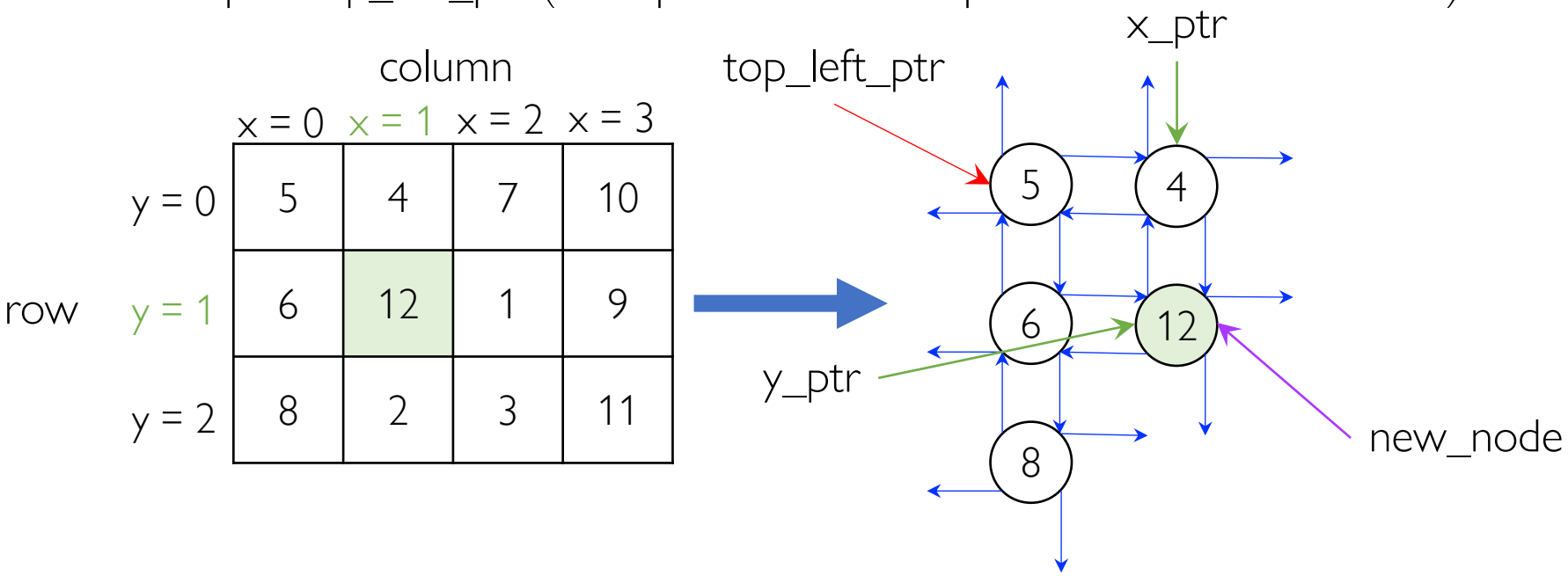
- input: arr (2D array to turn into a 2D DLL)
- output: top_left_ptr (head pointer to the top left node of the 2D DLL)



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDoublyLinkedListLoop

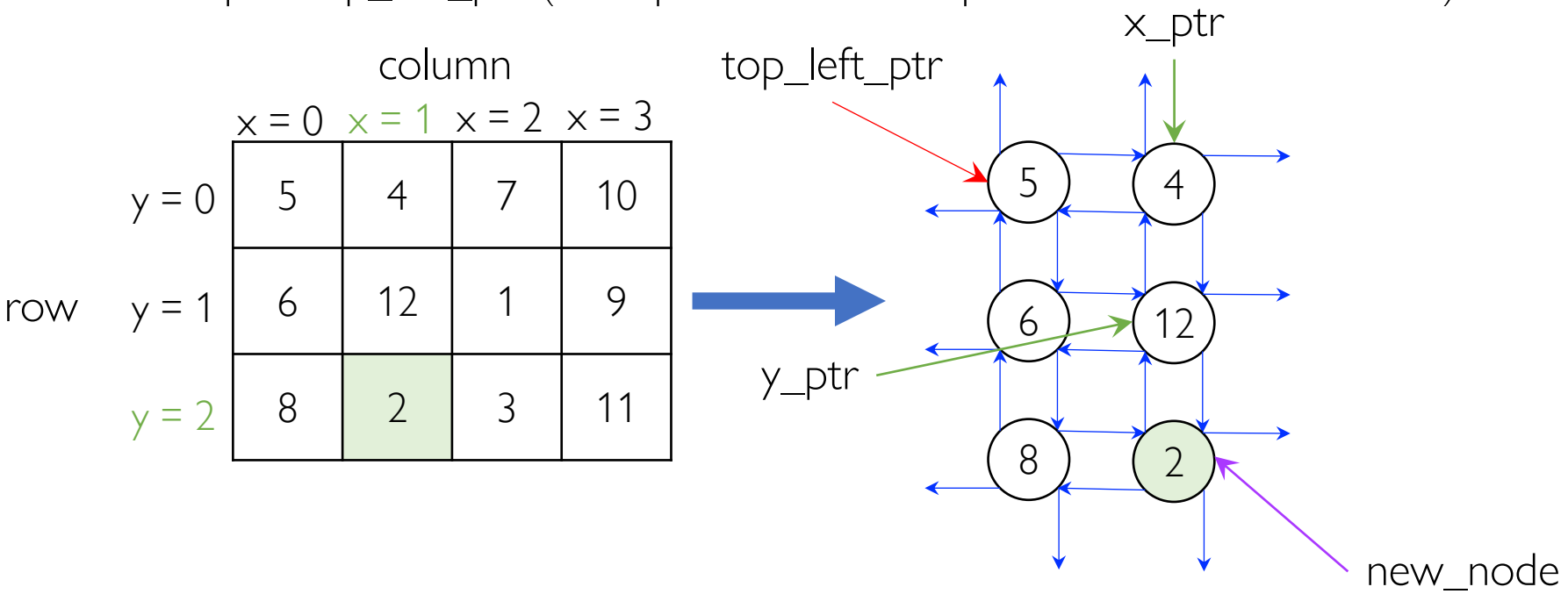
- input: arr (2D array to turn into a 2D DLL)
- output: top_left_ptr (head pointer to the top left node of the 2D DLL)



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDoublyLinkedListLoop

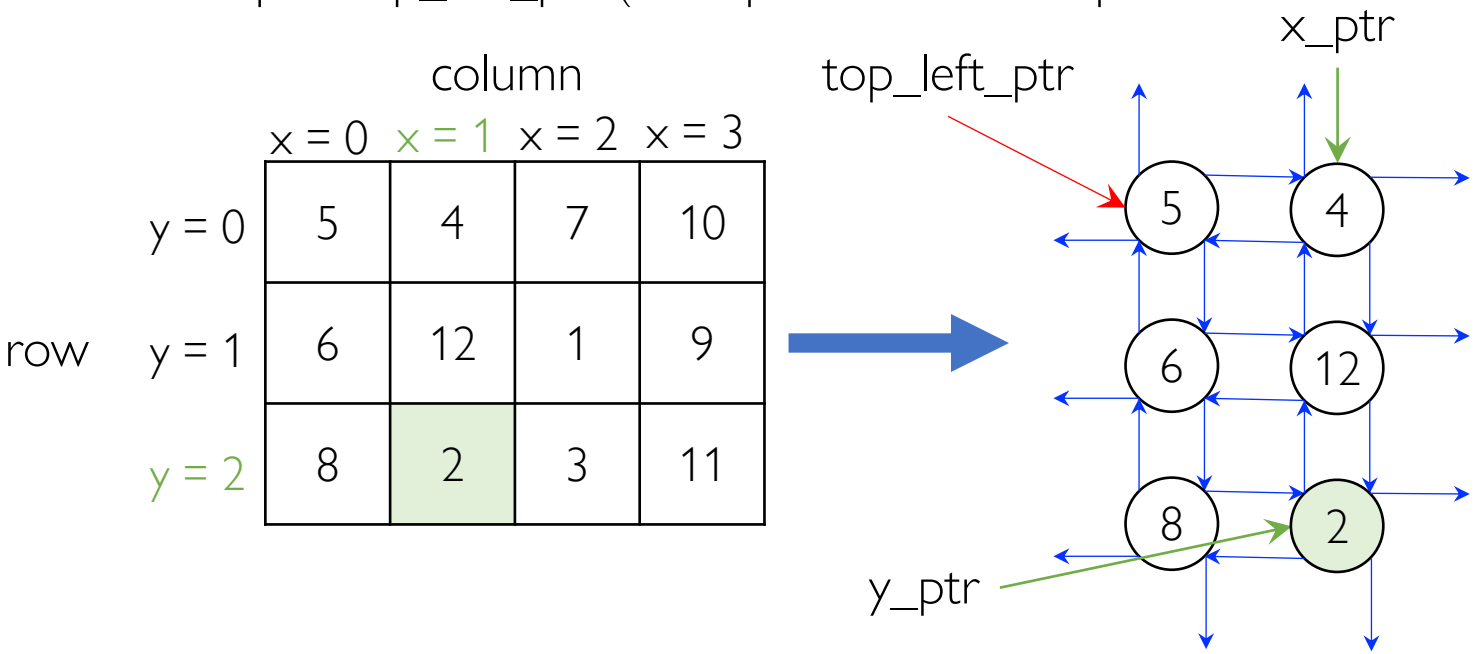
- input: arr (2D array to turn into a 2D DLL)
- output: top_left_ptr (head pointer to the top left node of the 2D DLL)



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDoublyLinkedListLoop

- input: arr (2D array to turn into a 2D DLL)
- output: top_left_ptr (head pointer to the top left node of the 2D DLL)



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDoublyLinkedListRecursion

```
def constructDoublyLinkedListRecursion(arr):
    """ Converts a 2D array into a 2D doubly linked list by calling
    the recursee constructDLLRecursiveStep.

    input:
        arr: 2D array to turn into a 2D DLL

    output:
        head (top left node) of the 2D DLL of the input arr.
    """
    return constructDLLRecursiveStep(arr, 0, 0, None)
```

- Often times, the main function is not actually recursive.
  - `constructDLLRecursiveStep` only takes arr, but this is not enough to perform recursive operations
- The actual recursive step (`constructDLLRecursiveStep`)is called as a separate function to include additional arguments

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

```
def constructDLLRecursiveStep(arr, y, x, curr):
    """ Recursively construct the 2D DLL from the given array.
    This is the "recursee" of constructDoublyLinkedListRecursion.

    input:
        arr: The 2D array to construct the 2D DLL from.
        y: y-coordinate of the array to get the value from.
        x: x-coordinate of the array to get the value from.
        curr: The current node to connect the new node from.

    output:
        new_node: The newly created node which connects to curr node.
    """

    Base case

    Operations

    Recursive Calls
```

arr: always need to pass in arr

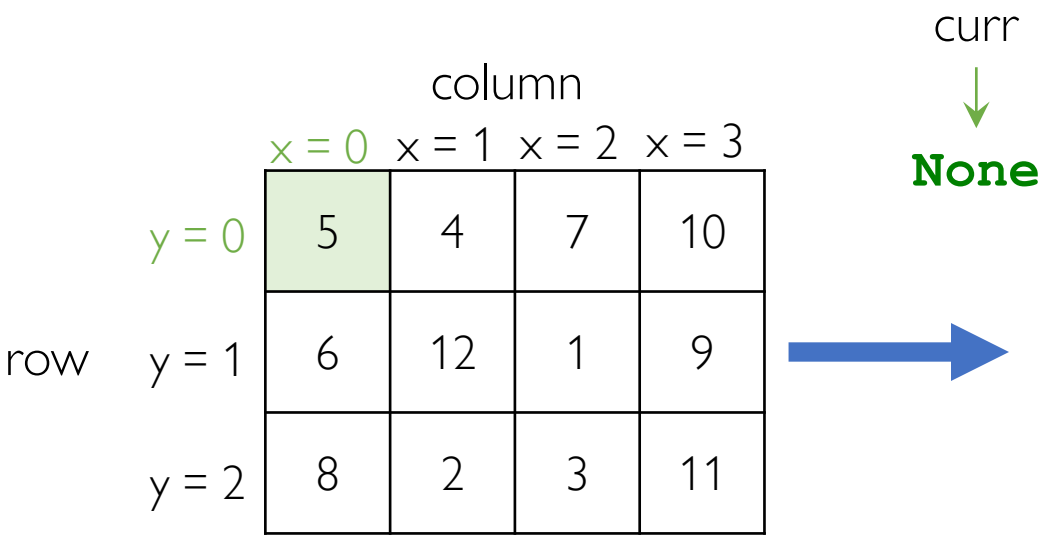
y and x: keep track of the array indices you work on in each recursive step

curr: the current node pointer within the DLL being constructed

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

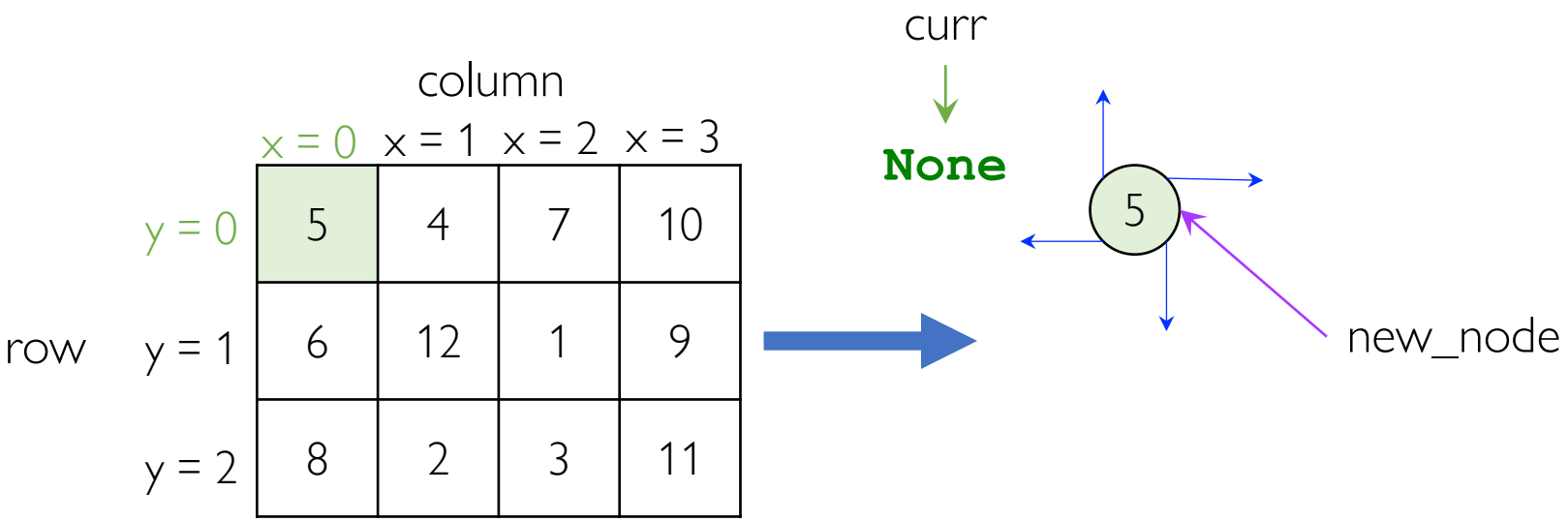
- `constructDLLRecursiveStep(arr, y=0, x=0, curr=None)`



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, y=0, x=0, curr=None)`

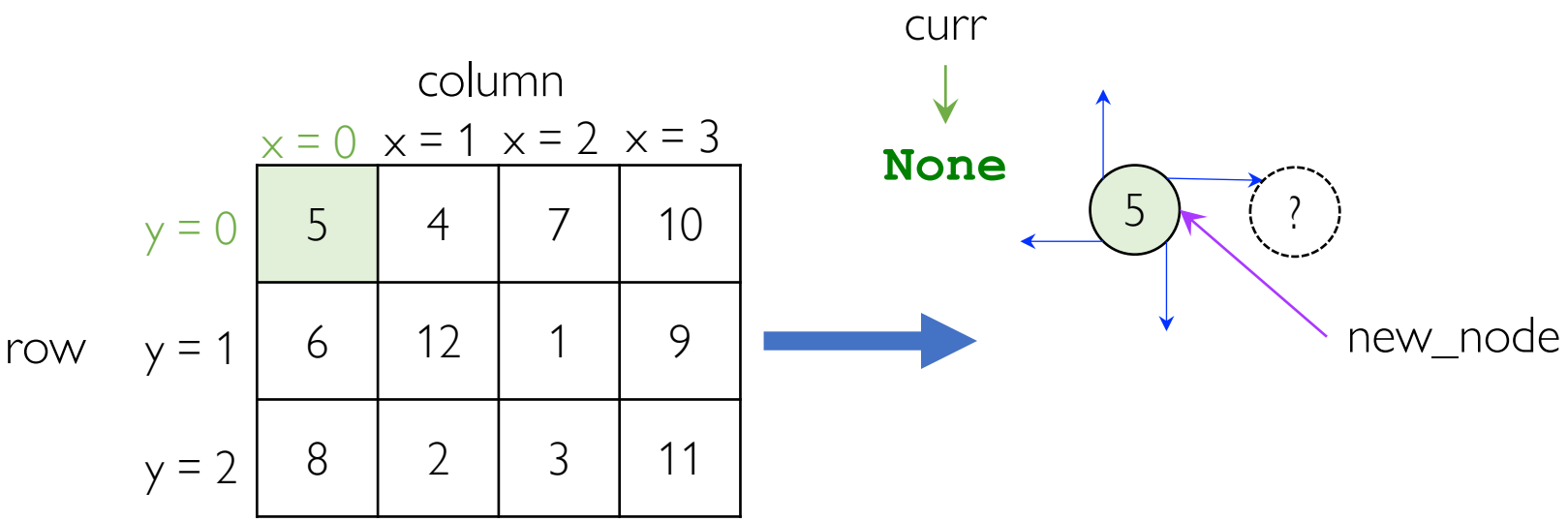



```
new_node = Node(arr[y,x])

... do necessary operations
```

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, y=0, x=0, curr=None)`

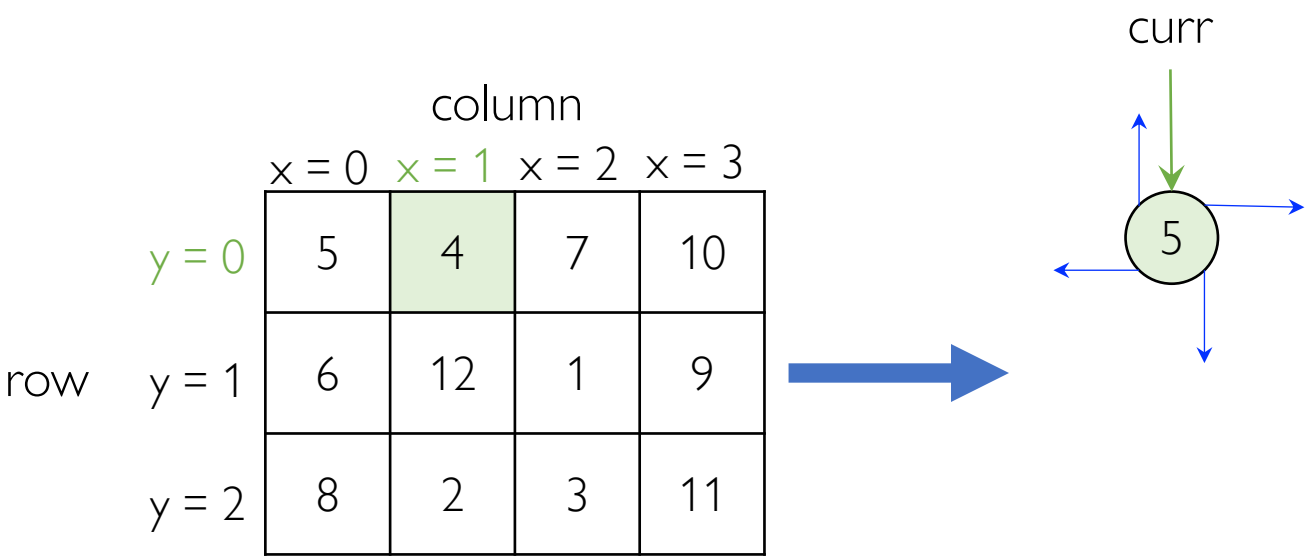



```
new_node = Node(arr[y,x])

... do necessary operations

new_node.right = constructDLLRecursiveStep(arr, y, x+1, new_node)
```

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

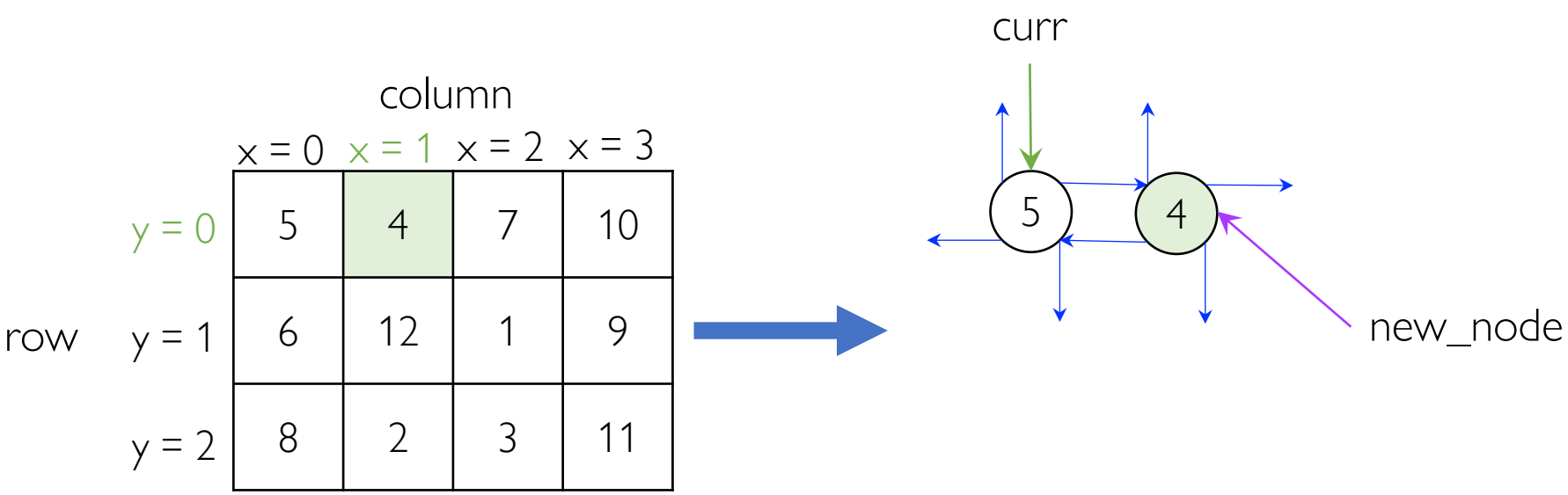
- `constructDLLRecursiveStep(arr, 0, 1, curr)`



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 0, 1, curr)`

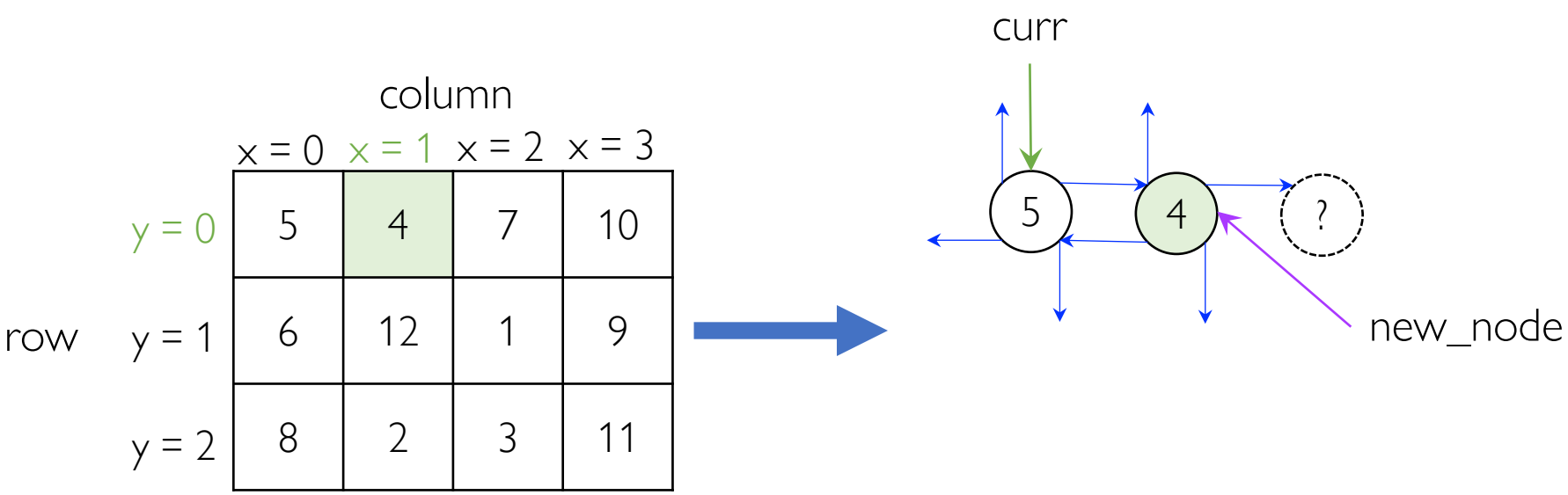



```
new_node = Node(arr[y,x])

... do necessary operations
```

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 0, 1, curr)`

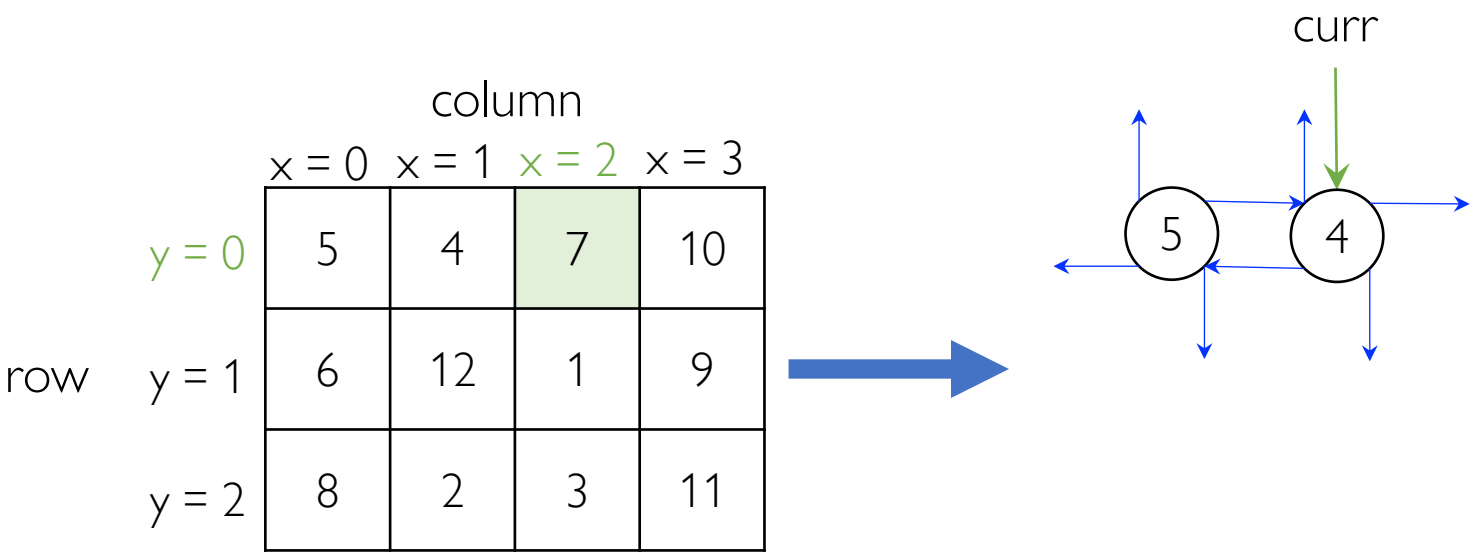



```
new_node = Node(arr[y,x])

... do necessary operations

new_node.right = constructDLLRecursiveStep(arr, y, x+1, new_node)
```

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 0, 2, curr)`



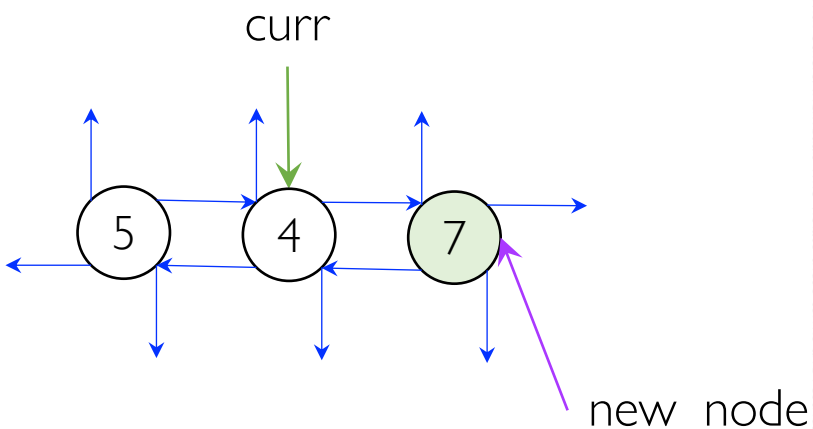
This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 0, 2, curr)`

column

row	x = 0	x = 1	x = 2	x = 3
y = 0	5	4	7	10
y = 1	6	12	1	9
y = 2	8	2	3	11




```
new_node = Node(arr[y,x])

... do necessary operations
```

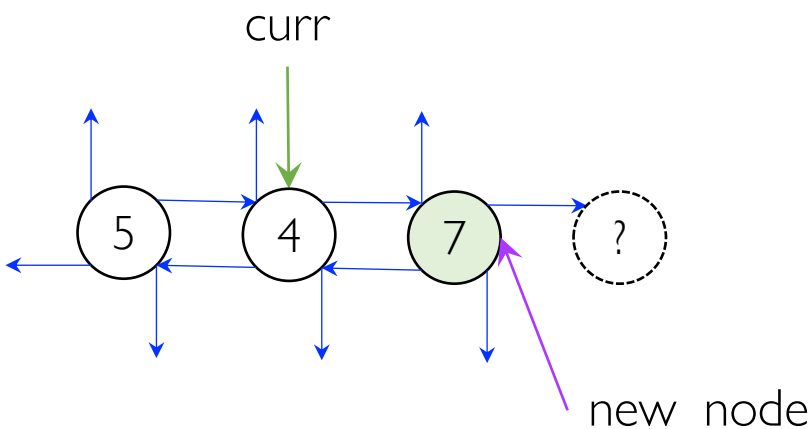
This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 0, 2, curr)`

column

row	x = 0	x = 1	x = 2	x = 3
y = 0	5	4	7	10
y = 1	6	12	1	9
y = 2	8	2	3	11




```
new_node = Node(arr[y,x])

... do necessary operations

new_node.right = constructDLLRecursiveStep(arr, y, x+1, new_node)
```

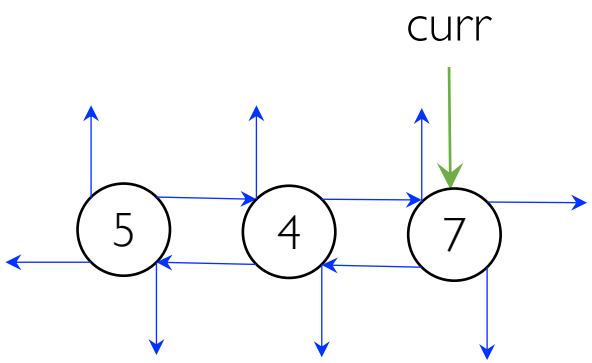
This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 0, 3, curr)`

column

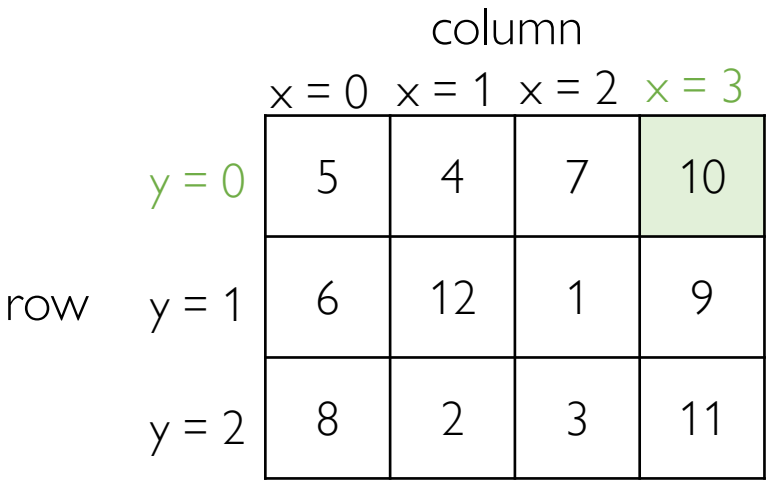
row	x = 0	x = 1	x = 2	x = 3
y = 0	5	4	7	10
y = 1	6	12	1	9
y = 2	8	2	3	11

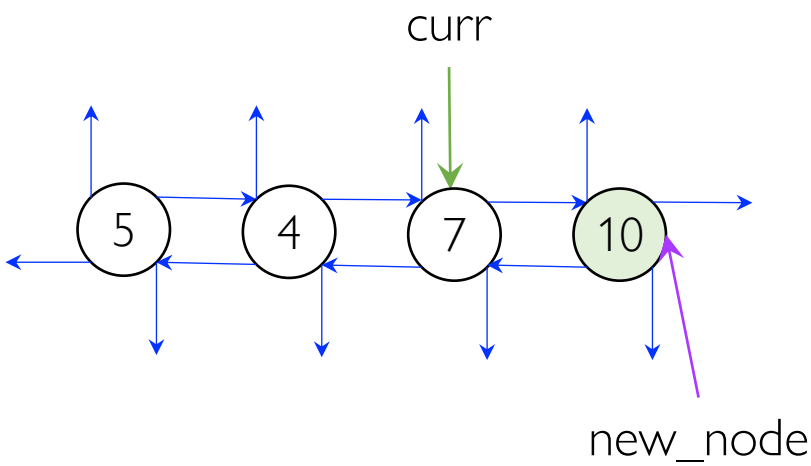


This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 0, 3, curr)`






```
new_node = Node(arr[y,x])

... do necessary operations
```

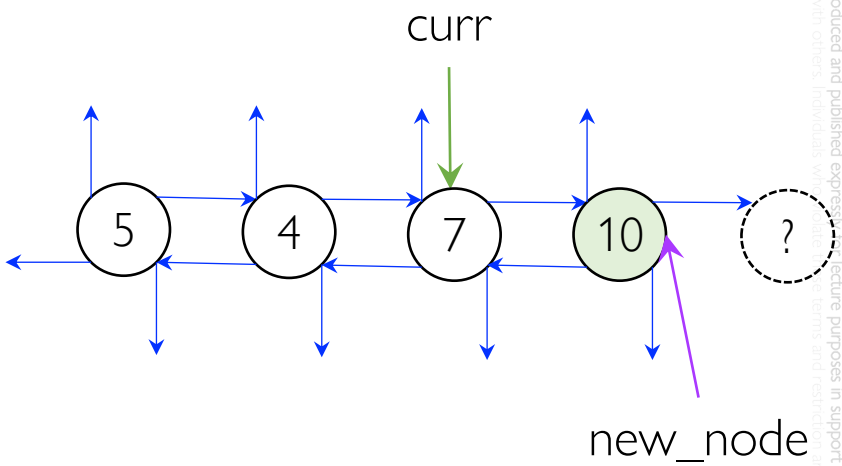
This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 0, 3, curr)`

column

row	x = 0	x = 1	x = 2	x = 3
y = 0	5	4	7	10
y = 1	6	12	1	9
y = 2	8	2	3	11




```
new_node = Node(arr[y,x])

... do necessary operations

new_node.right = constructDLLRecursiveStep(arr, y, x+1, new_node)
```

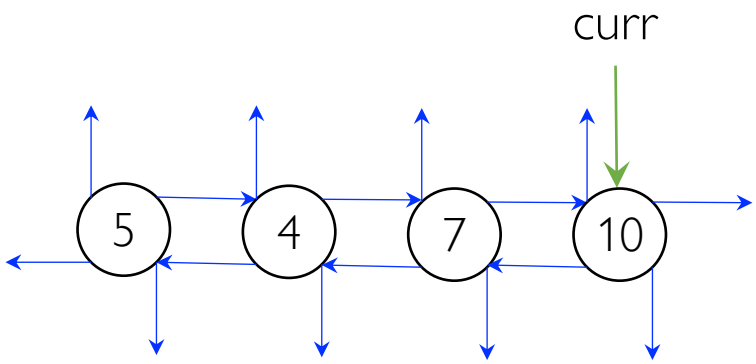
This lecture material has been produced and published exclusively for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 0, 4, curr)`

column

row	x = 0	x = 1	x = 2	x = 3
y = 0	5	4	7	10
y = 1	6	12	1	9
y = 2	8	2	3	11




```
Base case check: x >= width
    return None
```

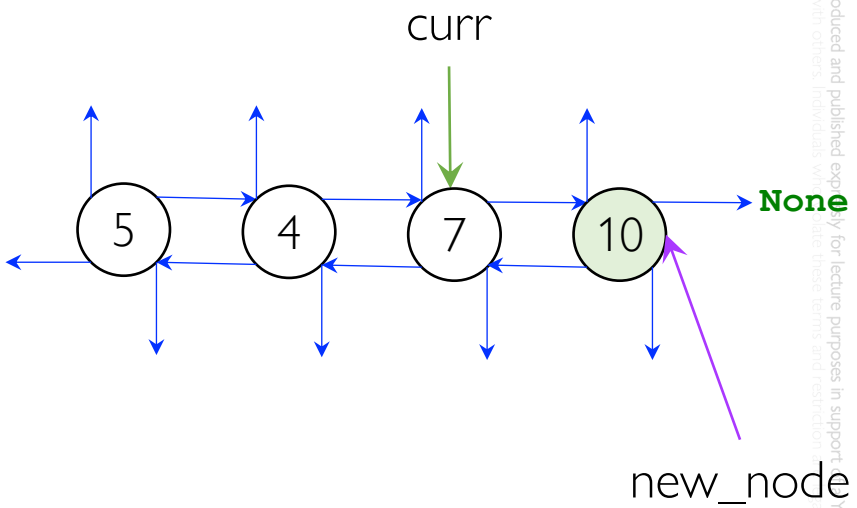
This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 0, 3, curr)`

column

row	x = 0	x = 1	x = 2	x = 3
y = 0	5	4	7	10
y = 1	6	12	1	9
y = 2	8	2	3	11




```
new_node = Node(arr[y,x])

... do necessary operations

new_node.right = constructDLLRecursiveStep(arr, y, x+1, new_node)
```

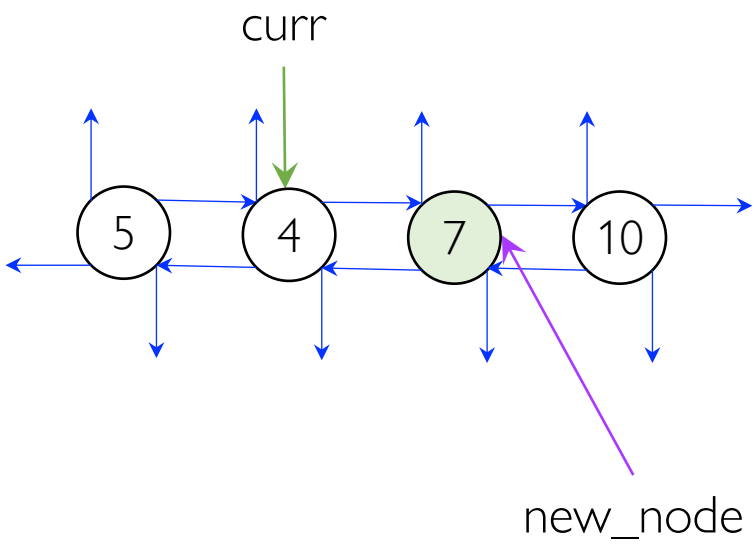
This lecture material has been produced and published exclusively for lecture purposes in support of Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who misuse contents and restrictions may be responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 0, 2, curr)`

column

row	x = 0	x = 1	x = 2	x = 3
y = 0	5	4	7	10
y = 1	6	12	1	9
y = 2	8	2	3	11




```
new_node = Node(arr[y,x])

... do necessary operations

new_node.right = constructDLLRecursiveStep(arr, y, x+1, new_node)
```

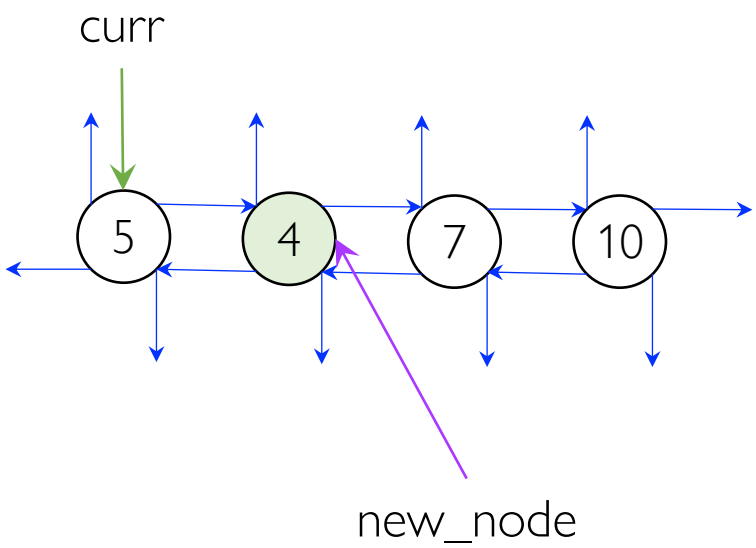
This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 0, 1, curr)`

column

row	x = 0	x = 1	x = 2	x = 3
y = 0	5	4	7	10
y = 1	6	12	1	9
y = 2	8	2	3	11

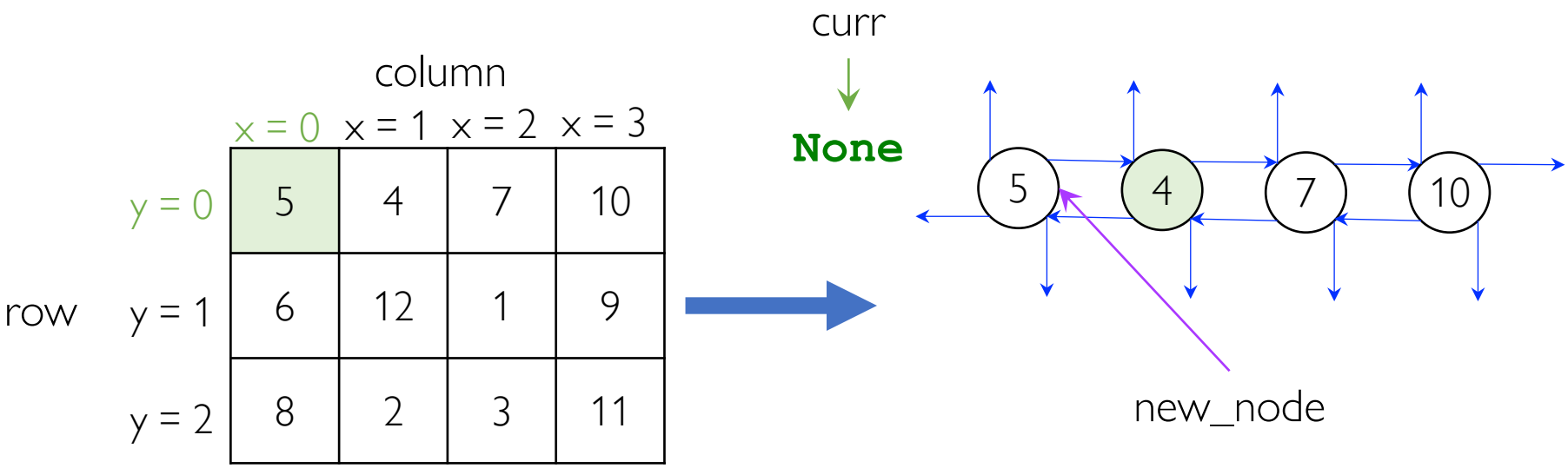



```
new_node = Node(arr[y,x])

... do necessary operations

new_node.right = constructDLLRecursiveStep(arr, y, x+1, new_node)
```

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 0, 0, curr)`



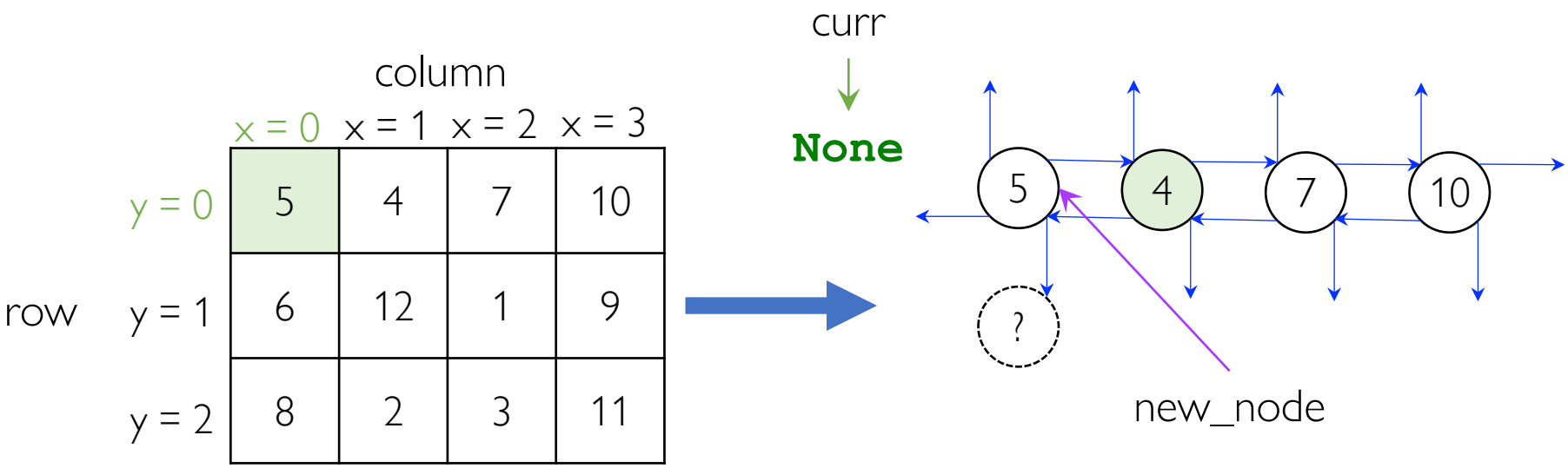

```
new_node = Node(arr[y,x])

... do necessary operations

new_node.down = constructDLLRecursiveStep(arr, y, x+1, new_node)
```

notice this is down instead of right

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 0, 0, curr)`




```
new_node.right = constructDLLRecursiveStep(arr, y+1, x, new_node)
```

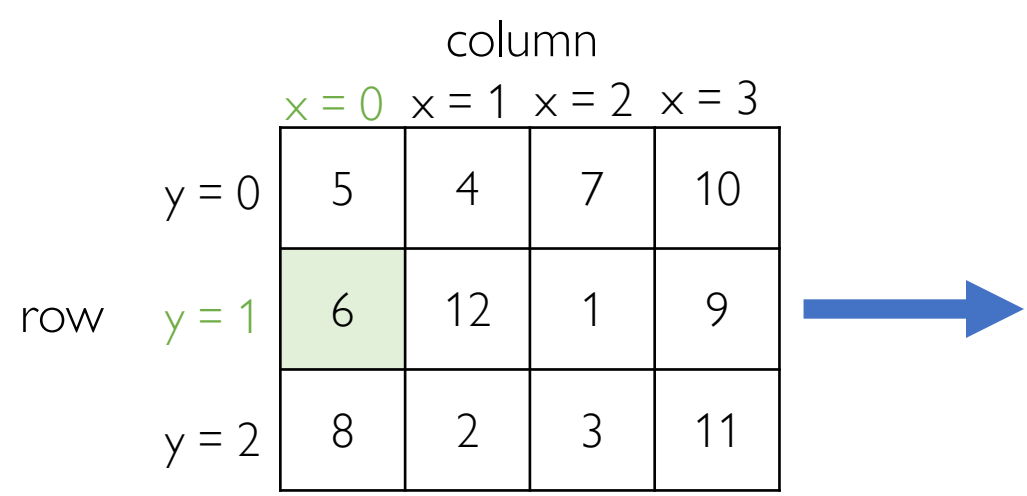
This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

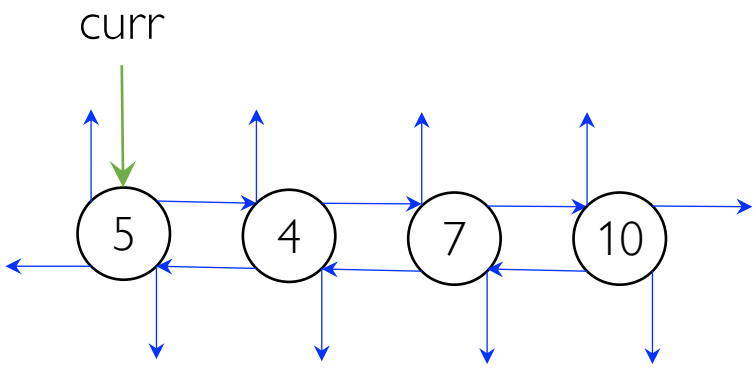
constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 1, 0, curr)`

column

row	x = 0	x = 1	x = 2	x = 3
y = 0	5	4	7	10
y = 1	6	12	1	9
y = 2	8	2	3	11

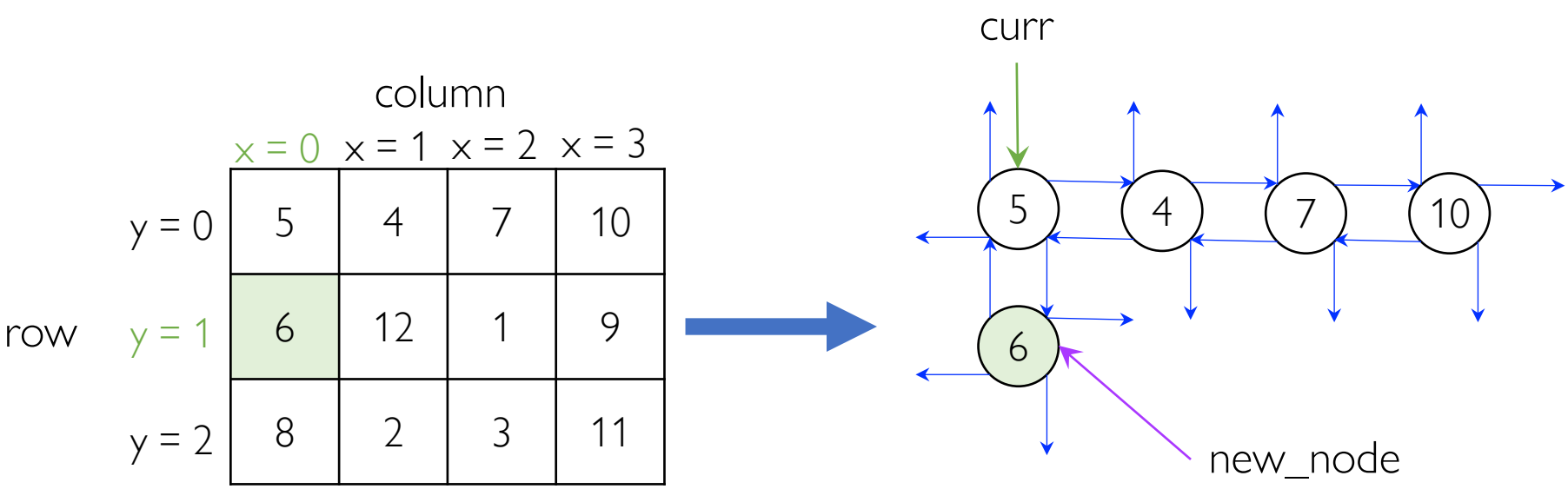




This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 1, 0, curr)`




```
new_node = Node(arr[y,x])

... do necessary operations
```

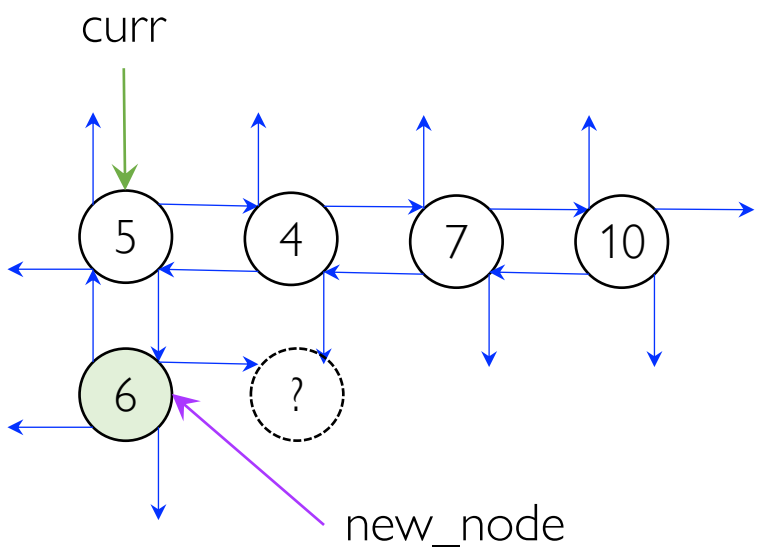
This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 1, 0, curr)`

column

row	x = 0	x = 1	x = 2	x = 3
y = 0	5	4	7	10
y = 1	6	12	1	9
y = 2	8	2	3	11

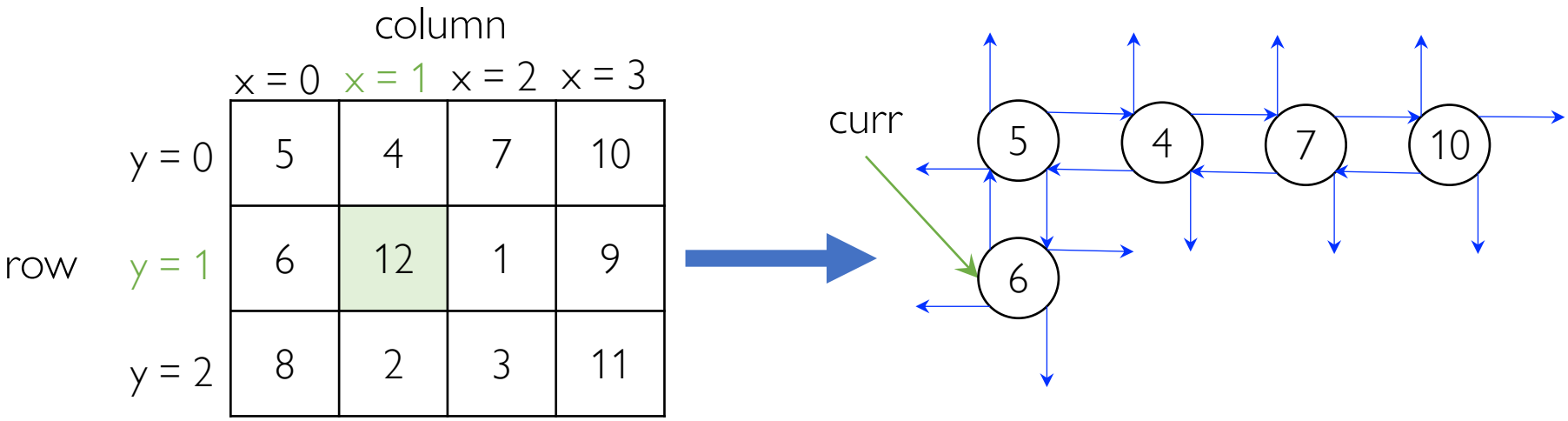



```
new_node = Node(arr[y,x])

... do necessary operations

new_node.right = constructDLLRecursiveStep(arr, y, x+1, new_node)
```

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 1, 1, curr)`



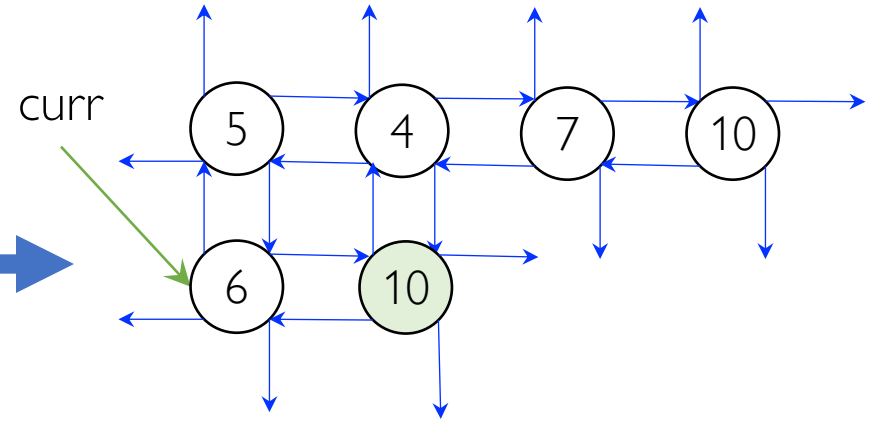
This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

constructDLLRecursiveStep

- `constructDLLRecursiveStep(arr, 1, 1, curr)`

column

row	x = 0	x = 1	x = 2	x = 3
y = 0	5	4	7	10
y = 1	6	12	1	9
y = 2	8	2	3	11




```
new_node = Node(arr[y,x])

... do necessary operations
```

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

Part 2: Context-Aware Image Resizing

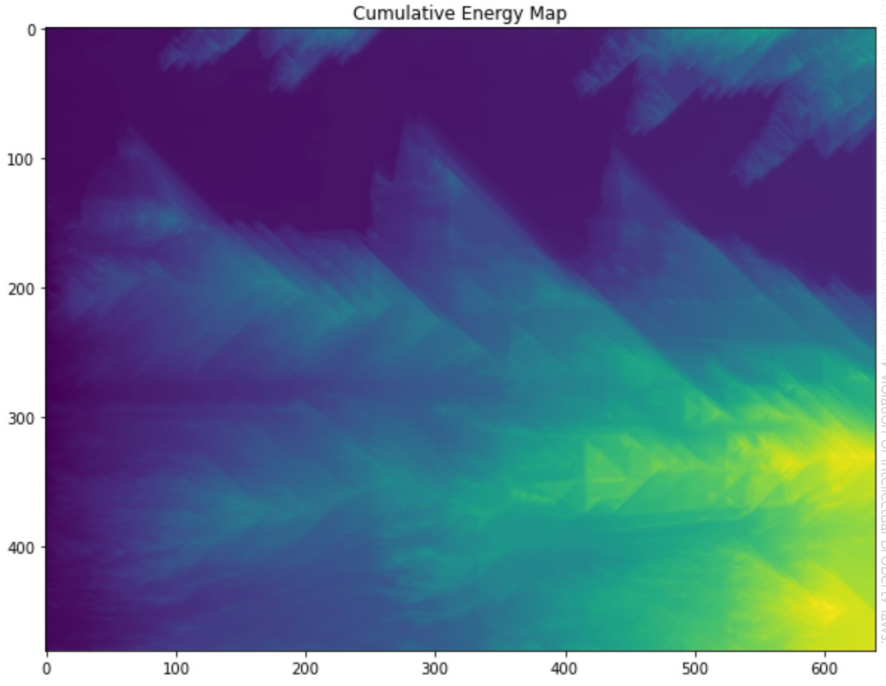
Context-aware resizing: Removes the "seam" which is the horizontal red path of "uninterestingness". This shrinks the image without "squashing" the objects, keeping their original ratios.

Normal resizing: "squashes" the entire image along with the objects, changing their original ratios.

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with individuals who are not registered for the course. Violation of intellectual property laws.

Step 1: Load Image and Compute Energy Map

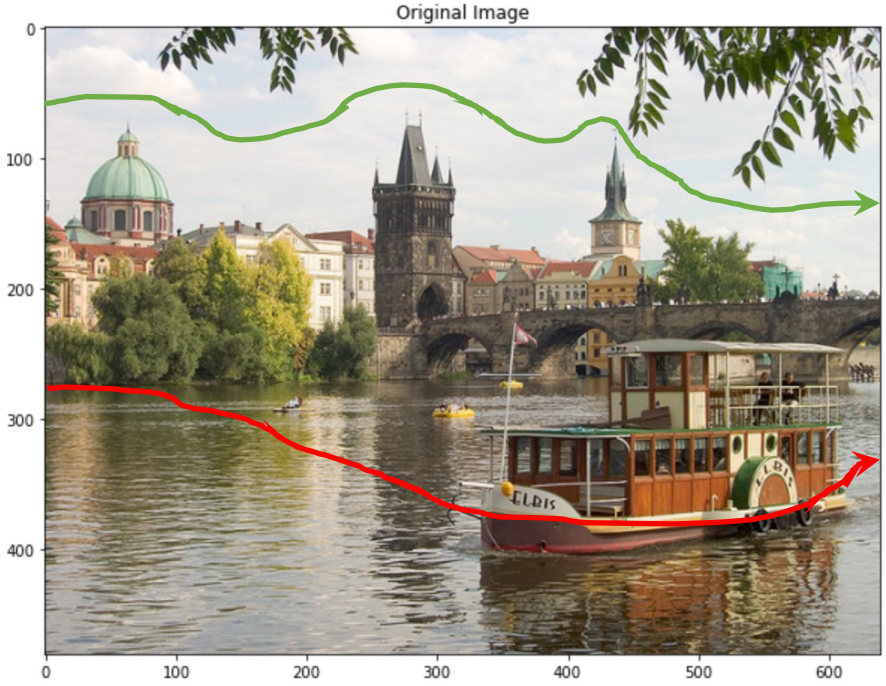
- Load the image as a 2D array
- Compute its cumulative "energy map" (also 2D array) which computes the level of "interestingness" needed from left edge to right edge
  - Already implemented for you

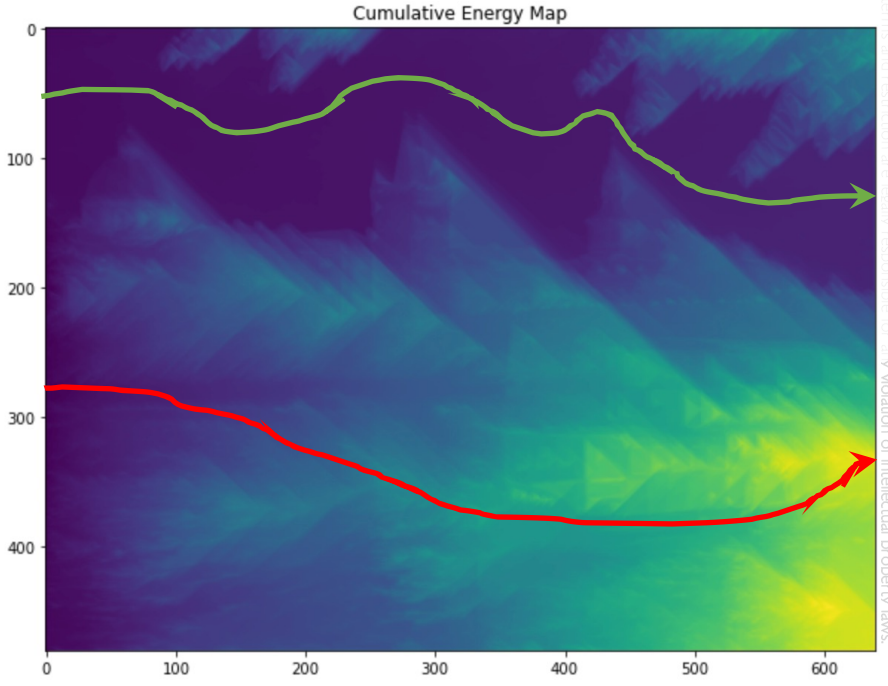


This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms are subject to penalties for violation of intellectual property laws.

Step 1: Load Image and Compute Energy Map

- Load the image as a 2D array
- Compute its cumulative "energy map" (also 2D array) which computes the level of "interestingness" needed from left edge to right edge
  - Already implemented for you

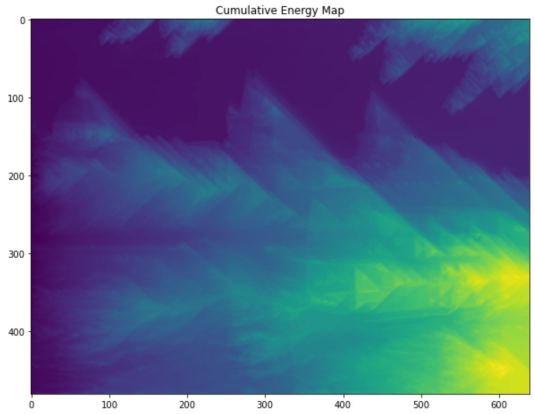




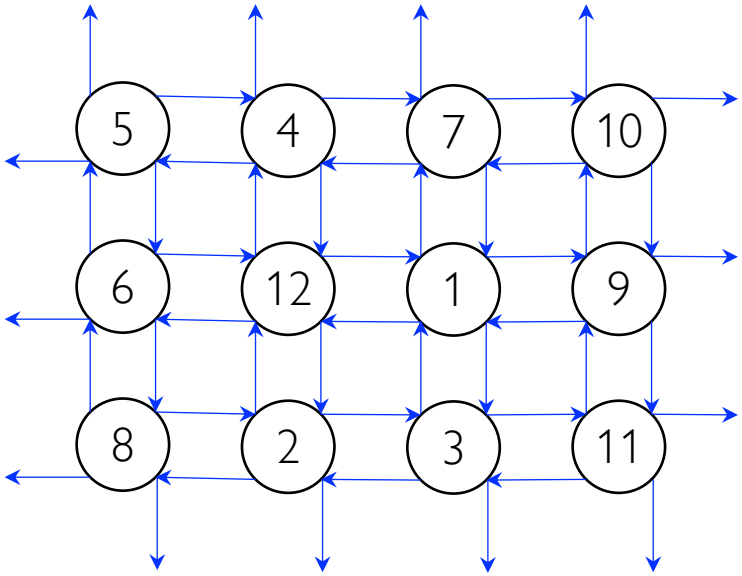
This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms are subject to penalties for violation of intellectual property laws.

Step 2: Convert 2D arrays to 2D DLLs

- Convert both 2D arrays into 2D DLL using constructDoublyLinkedListLoop function from Part 1.
  - If you completed constructDoublyLinkedListLoop from Part 1, this is also implemented for you

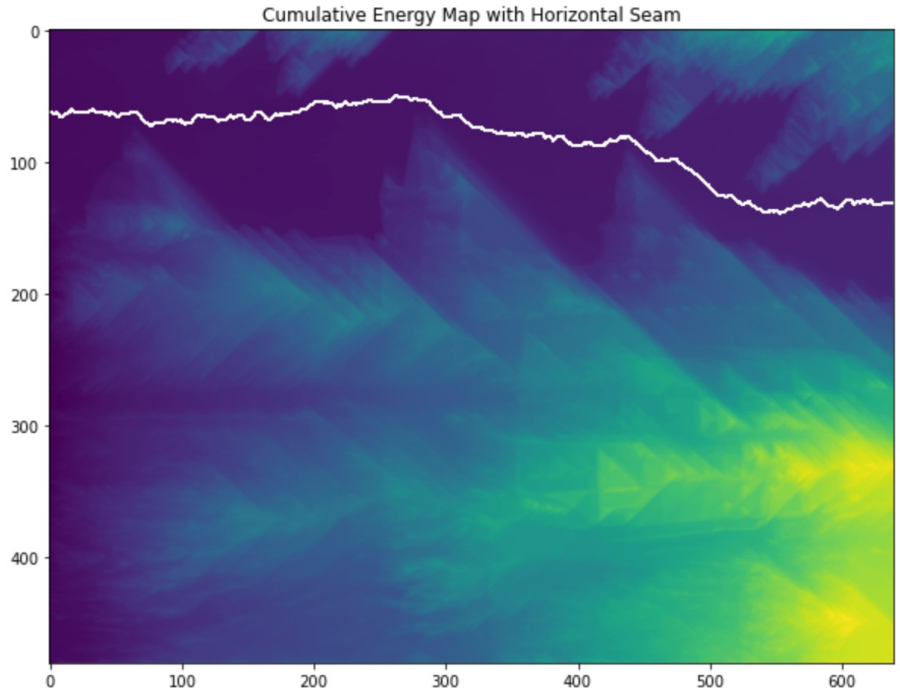


5	4	7	10
6	12	1	9
8	2	3	11



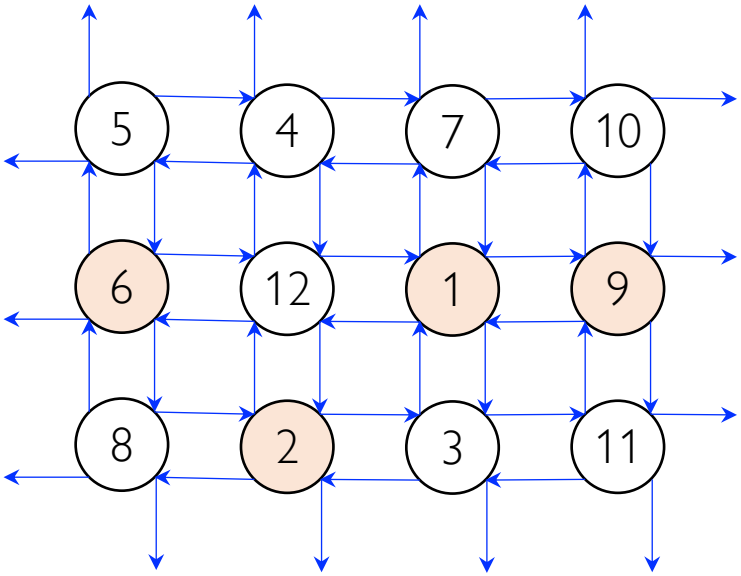
This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

Step 3: Find the "seam" of uninteresting path



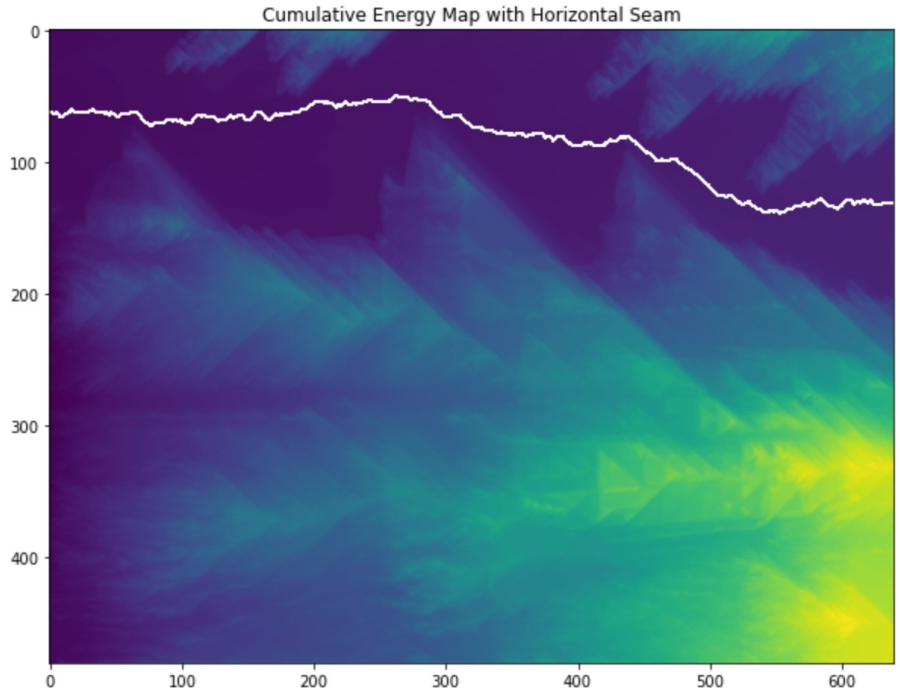
- In the energy_map, find the horizontal path called the "seam" using findSeam function
  - Also implemented for you

5	4	7	10
6	12	1	9
8	2	3	11



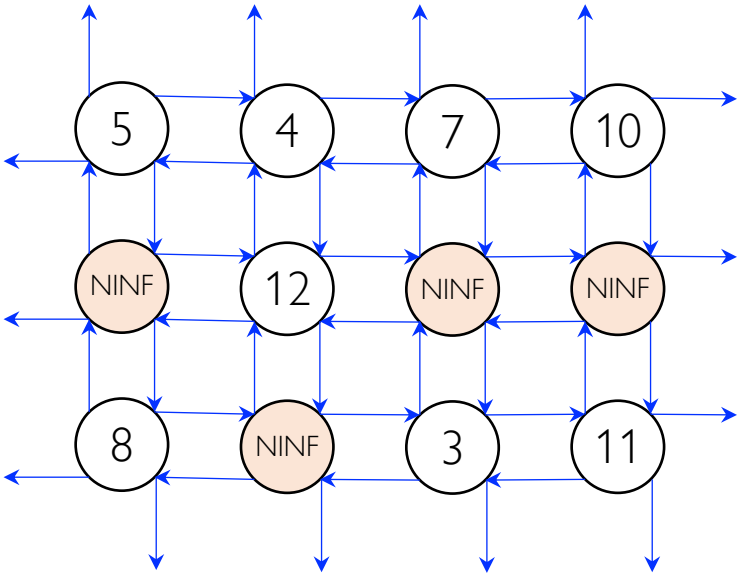
This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

Step 3: Find the "seam" of uninteresting path



- In the energy_map, find the horizontal path called the "seam" using findSeam function
  - Also implemented for you
- The output changes the node values to –infinity (NINF) to "mark" the path

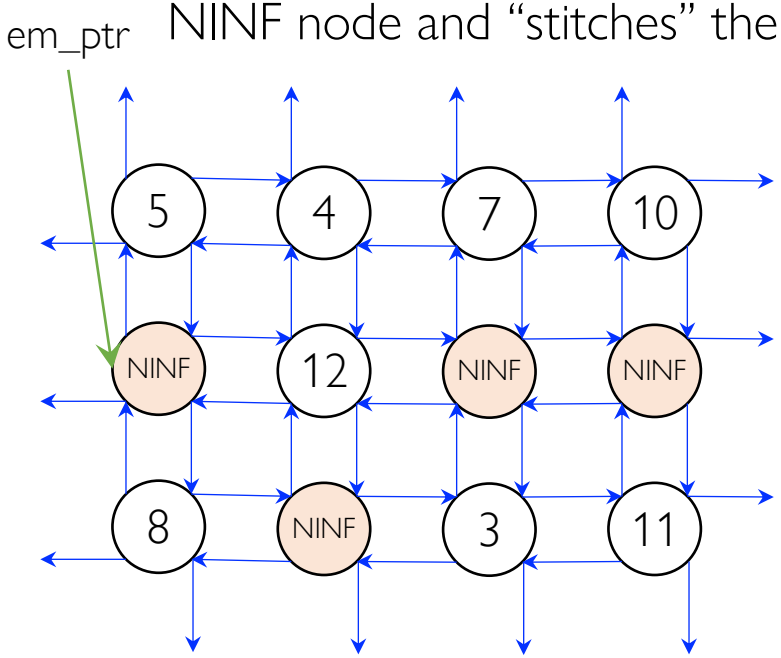
5	4	7	10
6	12	1	9
8	2	3	11



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

Step 4: Remove the nodes belong to the seam

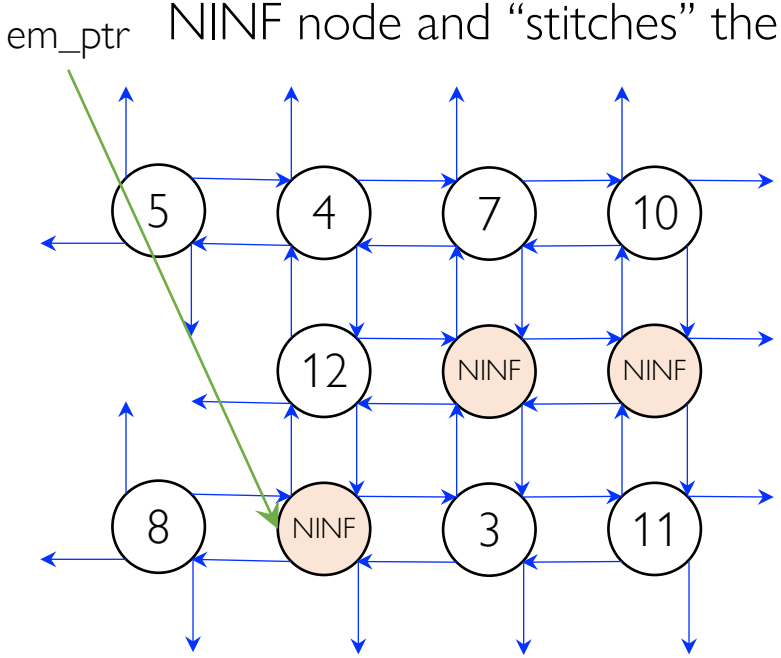
- Call "removeSeam" function to remove –infinity (NINF) value nodes that represent that seam.
- This calls "removeNodeRecursive" function which recursively removes the NINF node and "stitches" the nearby nodes



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with other individuals who do not take these terms and restrictions are legally responsible for any violation of intellectual property laws.

Step 4: Remove the nodes belong to the seam

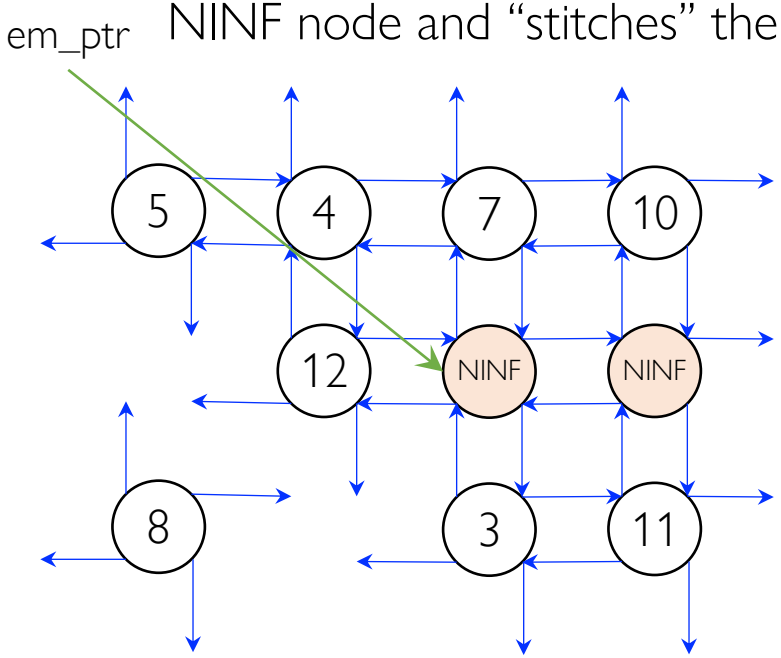
- Call "removeSeam" function to remove –infinity (NINF) value nodes that represent that seam.
- This calls "removeNodeRecursive" function which recursively removes the NINF node and "stitches" the nearby nodes



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with other individuals who do not take these terms and restrictions are legally responsible for any violation of intellectual property laws.

Step 4: Remove the nodes belong to the seam

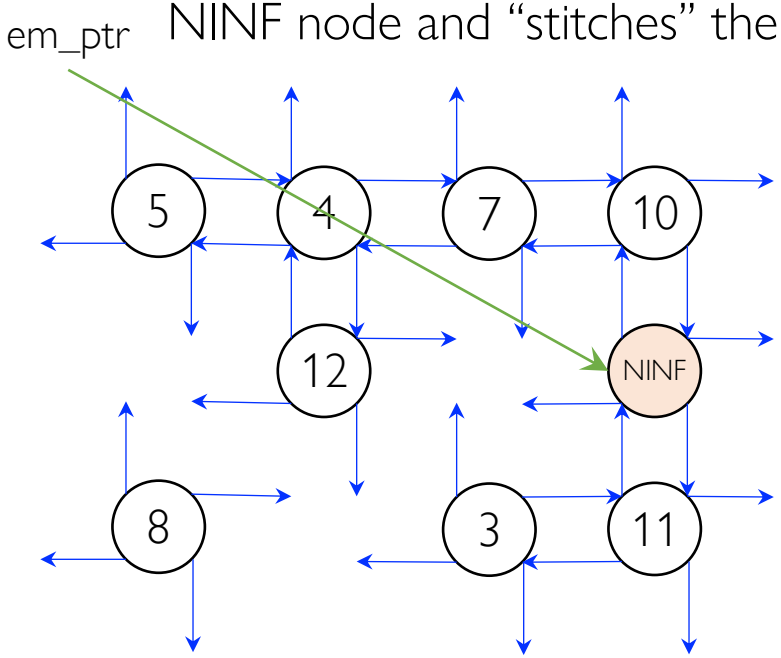
- Call "removeSeam" function to remove –infinity (NINF) value nodes that represent that seam.
- This calls "removeNodeRecursive" function which recursively removes the NINF node and "stitches" the nearby nodes



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with other individuals who would like these items and restriction are legally responsible for any violation of intellectual property laws.

Step 4: Remove the nodes belong to the seam

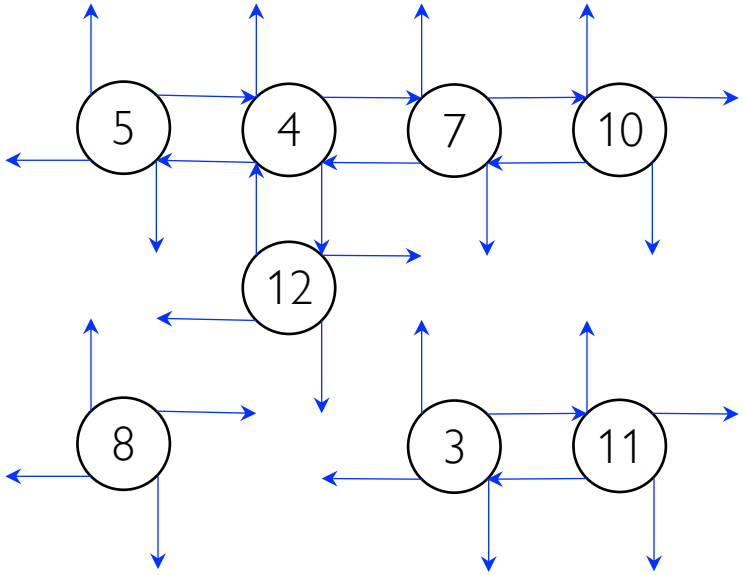
- Call "removeSeam" function to remove –infinity (NINF) value nodes that represent that seam.
- This calls "removeNodeRecursive" function which recursively removes the NINF node and "stitches" the nearby nodes



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with other individuals who would like these items and restriction are legally responsible for any violation of intellectual property laws.

Step 4: Remove the nodes belong to the seam

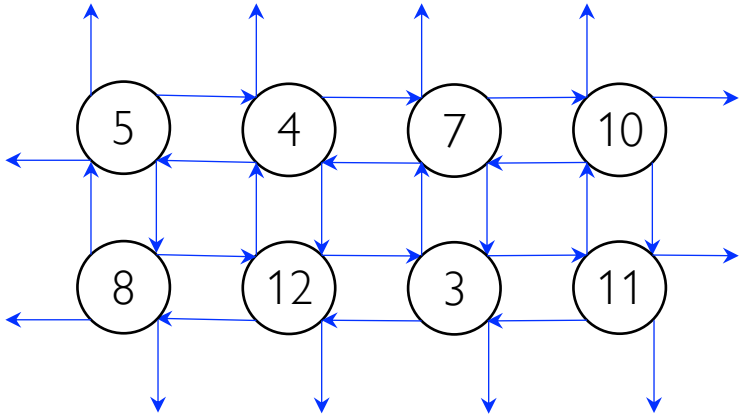
- Call "removeSeam" function to remove –infinity (NINF) value nodes that represent that seam.
- This calls "removeNodeRecursive" function which recursively removes the NINF node and "stitches" the nearby nodes



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with other individuals who would like these items and restriction are legally responsible for any violation of intellectual property laws.

Step 4: Remove the nodes belong to the seam

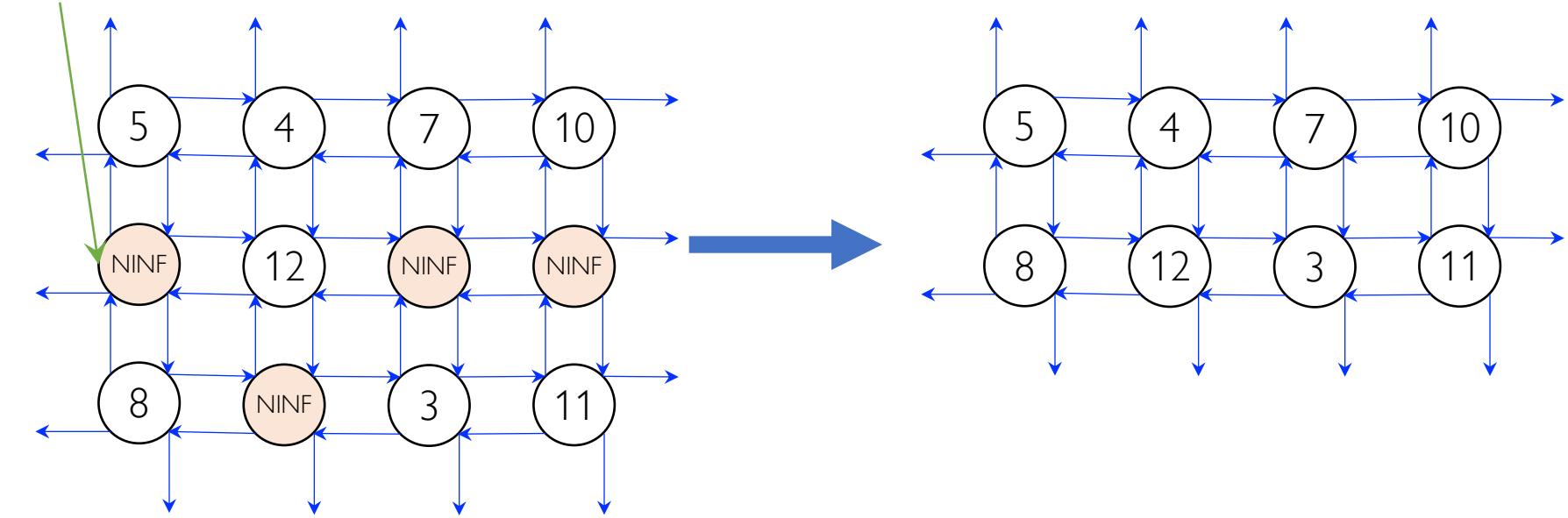
- Call “removeSeam” function to remove –infinity (NINF) value nodes that represent that seam.
- This calls “removeNodeRecursive” function which recursively removes the NINF node and “stitches” the nearby nodes



This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with other individuals who would like these items and restriction are legally responsible for any violation of intellectual property laws.

Step 4: Remove the nodes belong to the seam

- Call "removeSeam" function to remove –infinity (NINF) value nodes that represent that seam.
- This calls "removeNodeRecursive" function which recursively removes the NINF node and "stitches" the nearby nodes

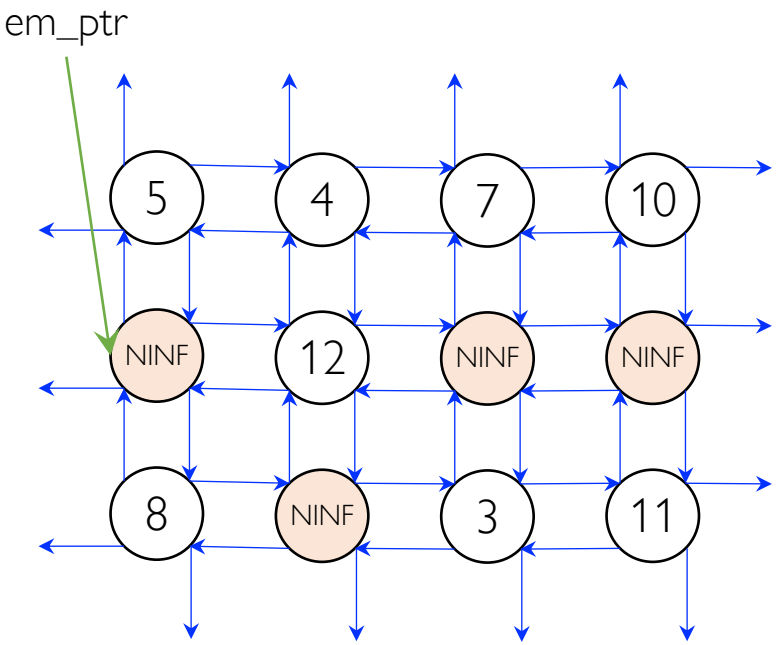


Note: For convenience, you can assume that all NINF nodes have node.up and node.down. The energy map on the provided image will not compute paths along the top and bottom edges of the image.

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with other individuals who were not registered for the course. Any violation of intellectual property laws.

Step 4: Remove the nodes belong to the seam

- Note that this removal based on energy map should also happen to image
  - recall, we use the energy map to identify the seam to ultimately remove the seam of the image



energy map
(2D DLL)

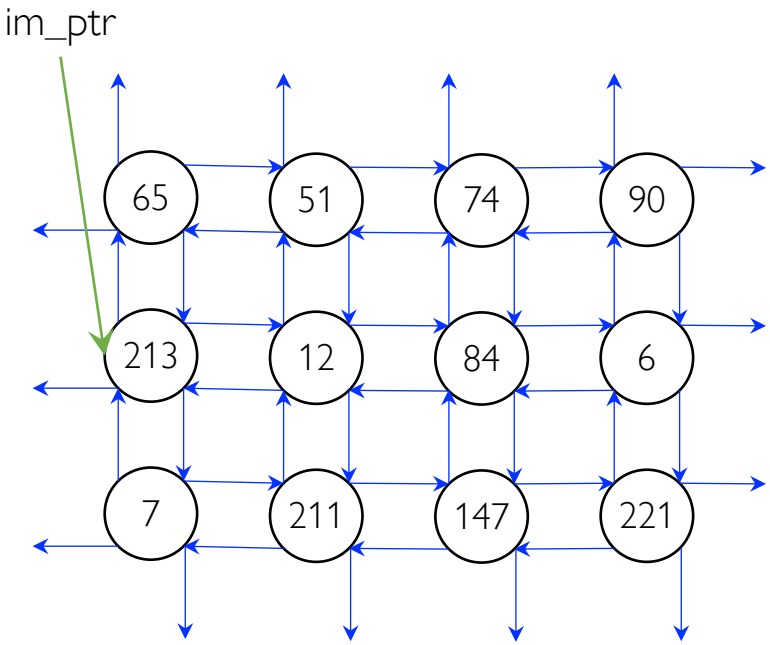
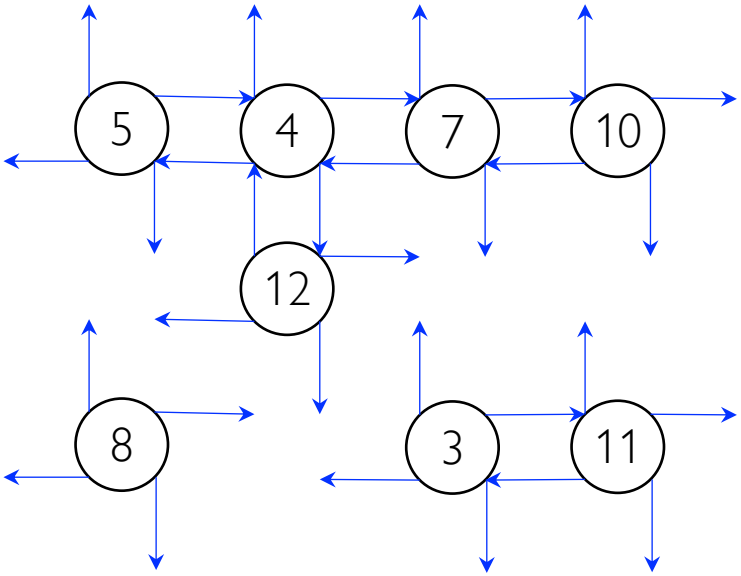


image
(2D DLL)

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

Step 4: Remove the nodes belong to the seam

- Note that this removal based on energy map should also happen to image
  - recall, we use the energy map to identify the seam to ultimately remove the seam of the image
- One call to removeSeam reduces image by 1 pixel



energy map
(2D DLL)

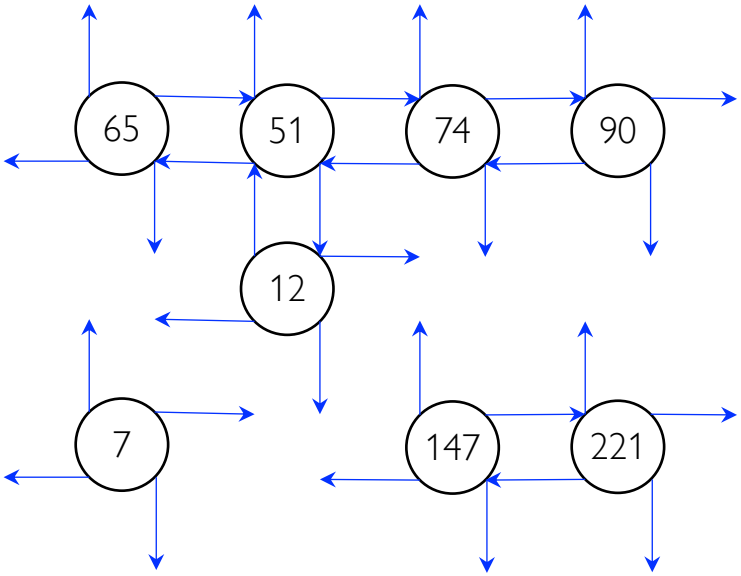


image
(2D DLL)

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

Step 4: Remove the nodes belong to the seam

- We repeatedly findSeam and removeSeam to reduce the image by 1 pixel at a time
- Once removed, turn 2D DLLs into 2D arrays for plotting

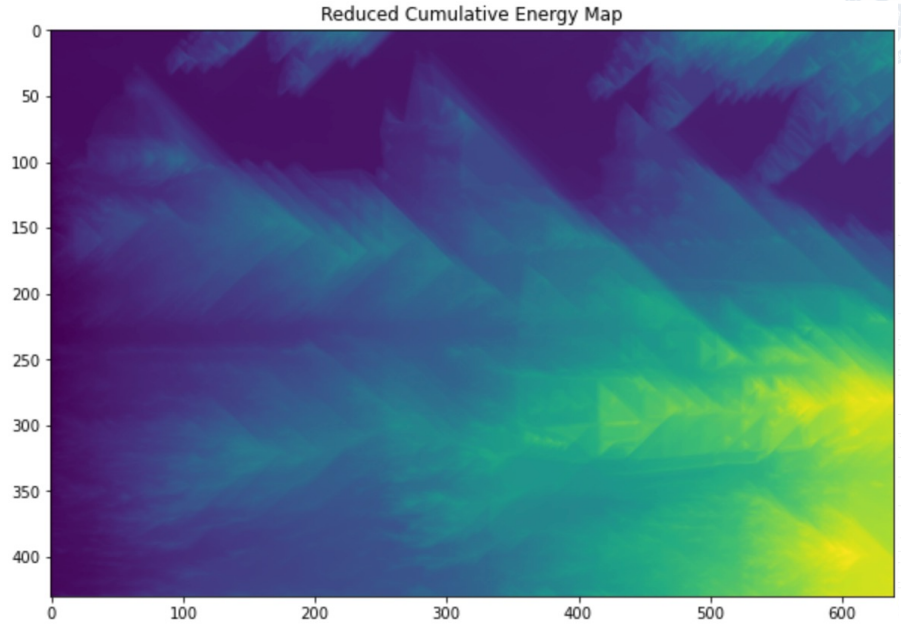
Original image

Repeatedly removing the seam

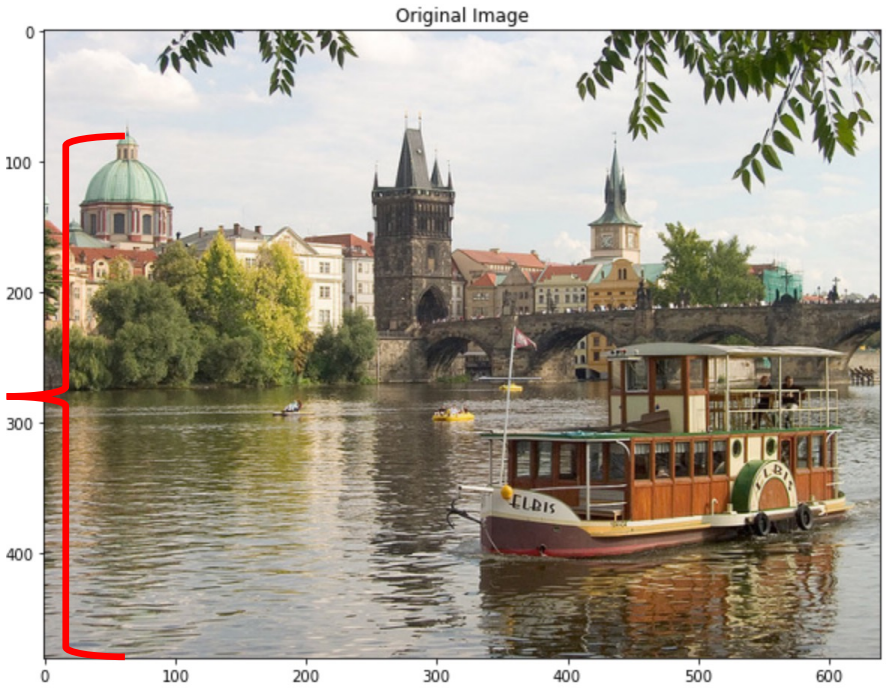
This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.

Step 4

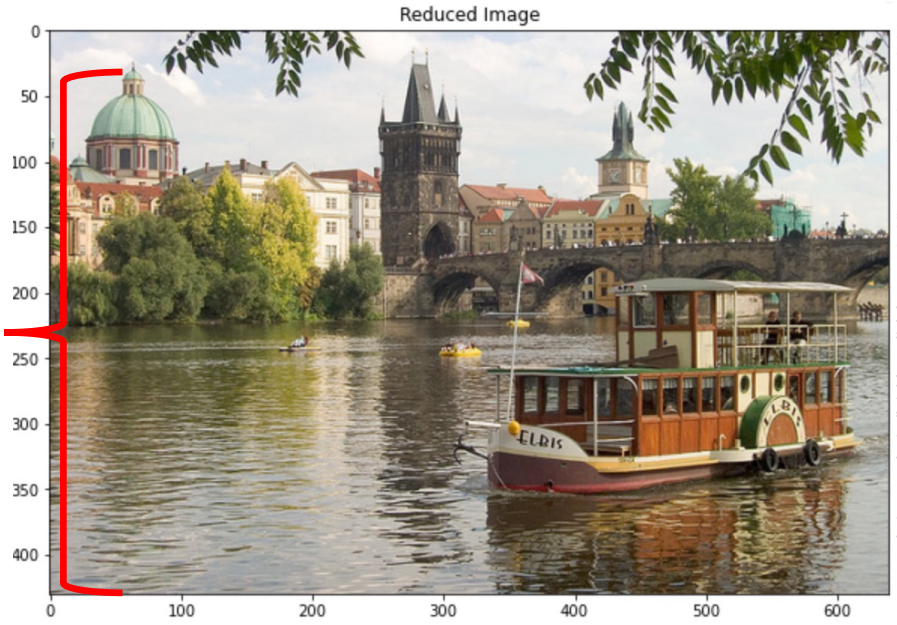
- We repeatedly findSeam and removeSeam to reduce the image by 1 pixel at a time
- Once removed, turn 2D DLLs into 2D arrays for plotting
  - Step 5 implemented



Reduce energy map



Original image



Reduce image

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other

Summary

- Walk through **assignment2.ipynb**
- Implement missing functions in **doublylinkedlist.py**
- Don't modify **vision_utils.py**
- Part 1:
  - constructDoublyLinkedListLoop
  - constructDoublyLinkedListRecursiveStep
- Part 2:
  - getWidth
  - removeNodeRecursive

This lecture material has been produced and published expressly for lecture purposes in support of a Yonsei University course. This material cannot be used for any other purposes and cannot be shared with others. Individuals who violate these terms and restrictions are legally responsible for any violation of intellectual property laws.