THE UNIVERSITY OF
MELBOURNE

COMP90038
Algorithms and Complexity

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Lecture 2 Concepts
(with thanks to Hara ergaard)

Toby Murray

toby.murray@unimelb.edu.au

DMD 8.17 (Level 8, Doug McDonell Bldg)

http://people.eng.unimelb.edu.au/tobym

@tobycmurray

Approaching a problem

- Can we cover this board with 31 tiles of the following form?

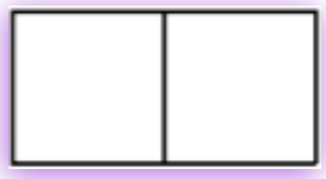
- This is the **mutilated checkerboard problem**.

- There are only finitely many ways we can arrange the 31 tiles, so there is a brute-force (and very inefficient) way of solving the problem.

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Approaching a problem



- Can we cover this board with 31 tiles of the following form?

- This is the **mutilated checkerboard problem**.

Assignment Project Exam Help

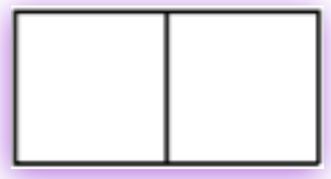
https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- There are only finitely many ways we can arrange the 31 tiles, so there is a brute-force (and very inefficient) way of solving the problem.

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Approaching a problem

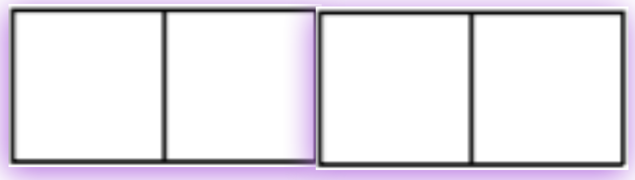


- Can we cover this board with 31 tiles of the following form?
- This is the **mutilated checkerboard problem**.
- There are only finitely many ways we can arrange the 31 tiles, so there is a brute-force (and very inefficient) way of solving the problem.

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Approaching a problem



- Can we cover this board with 31 tiles of the following form?
- This is the **mutilated checkerboard problem**.
- There are only finitely many ways we can arrange the 31 tiles, so there is a brute-force (and very inefficient) way of solving the problem.

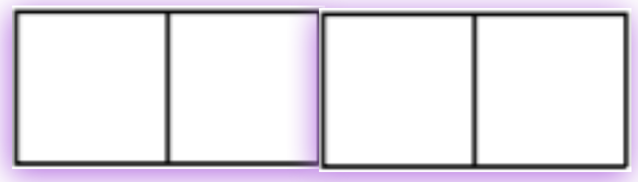
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Approaching a problem



- Can we cover this board with 31 tiles of the following form?

- This is the **mutilated checkerboard problem**.

Assignment Project Exam Help

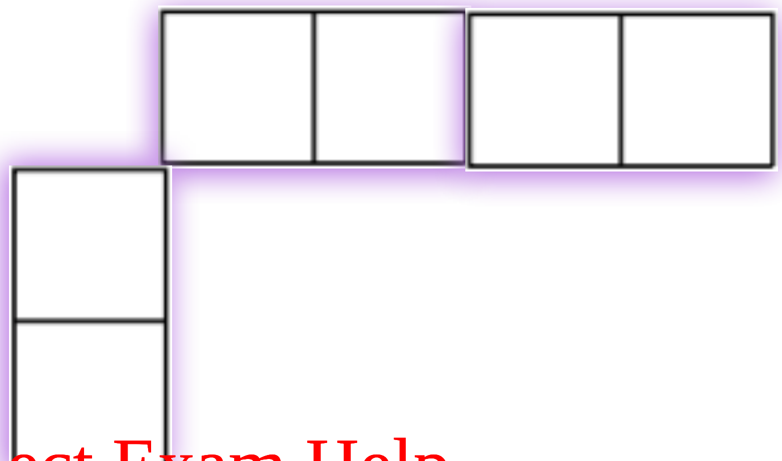
https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- There are only finitely many ways we can arrange the 31 tiles, so there is a brute-force (and very inefficient) way of solving the problem.

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Approaching a problem



- Can we cover this board with 31 tiles of the following form?
- This is the **mutilated checkerboard problem**.
- There are only finitely many ways we can arrange the 31 tiles, so there is a brute-force (and very inefficient) way of solving the problem.

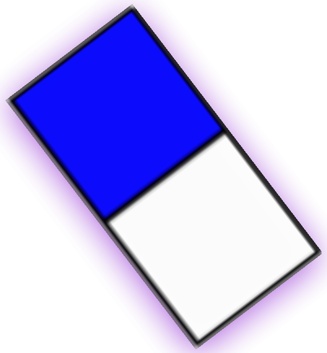
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Transform and Conquer?
Use abstraction?



- Can we cover this board with 31 tiles of the form shown?

Assignment Project Exam Help

https://eduassistpro.github.io/

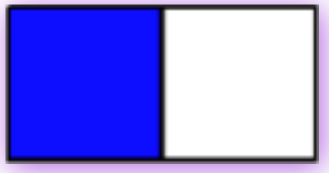
Add WeChat edu_assist_pro

- Why can we quickly deter that the answer is no?

- **Hint:** Using the way the squares are coloured helps.

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Transform and Conquer?
Use abstraction?



- Can we cover this board with 31 tiles of the form shown?

Assignment Project Exam Help

https://eduassistpro.github.io/

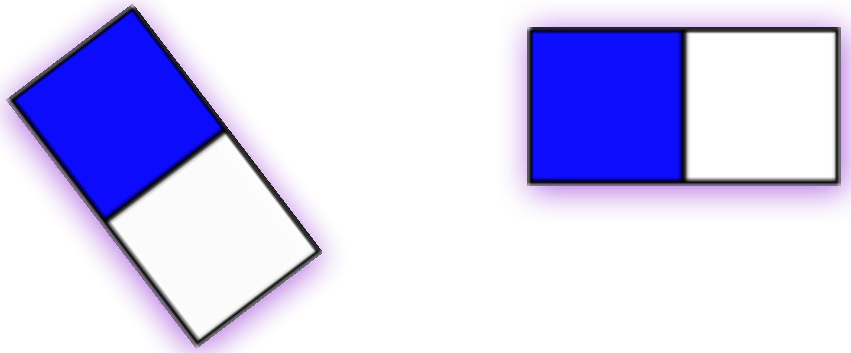
Add WeChat edu_assist_pro

- Why can we quickly deter that the answer is no?

- **Hint:** Using the way the squares are coloured helps.

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Transform and Conquer?
Use abstraction?



- Can we cover this board with 31 tiles of the form shown?

Assignment Project Exam Help

https://eduassistpro.github.io/

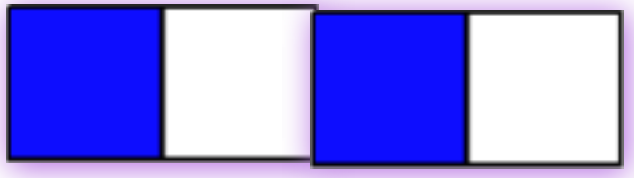
Add WeChat edu_assist_pro

- Why can we quickly deter that the answer is no?

- **Hint:** Using the way the squares are coloured helps.

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Transform and Conquer? Use abstraction?



- Can we cover this board with 31 tiles of the form shown?

Assignment Project Exam Help

- Why can we quickly deter that the answer is no?

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- **Hint:** Using the way the squares are coloured helps.

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Algorithms and Data Structures

- **Algorithms**: for solving problems, transforming data.

- **Data structures**: for storing data; arranging data in a way that suits an algorithm.
  - **Linear** data structures: stacks and q
  - Trees and graphs
  - Dictionaries
- Which data structures are you familiar with?

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Exercise

- Pick you favourite data structure and describe.
    - How to insert and item
    - How to find an item
    - How to handle duplicate items

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Array

- An array corresponds to a sequence of consecutive cells in memory.

- Depending on programming language: `A[0]` up to `A[n-1]`, or `A[1]` up to `A[n]`.

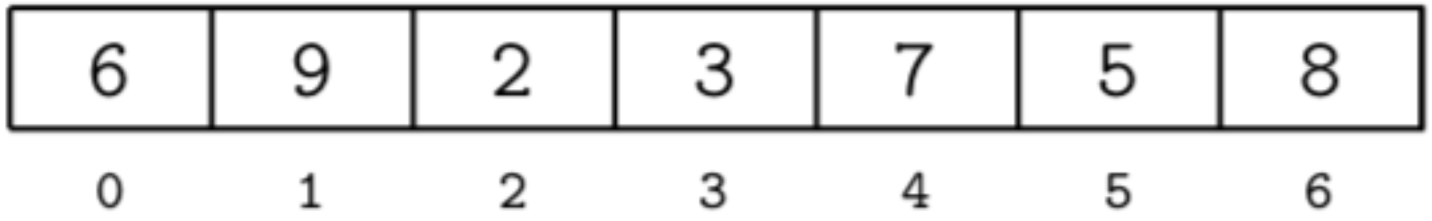
- Locating a cell, and storing or retrieving data at that cell is very fast.

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- The downside of an array is that maintai iguous bank of cells with information can be difficult and time-con



6	9	2	3	7	5	8
0	1	2	3	4	5	6

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Array

- An array corresponds to a sequence of consecutive cells in memory.

- Depending on programming language: `A[0]` up to `A[n-1]`, or `A[1]` up to `A[n]`.

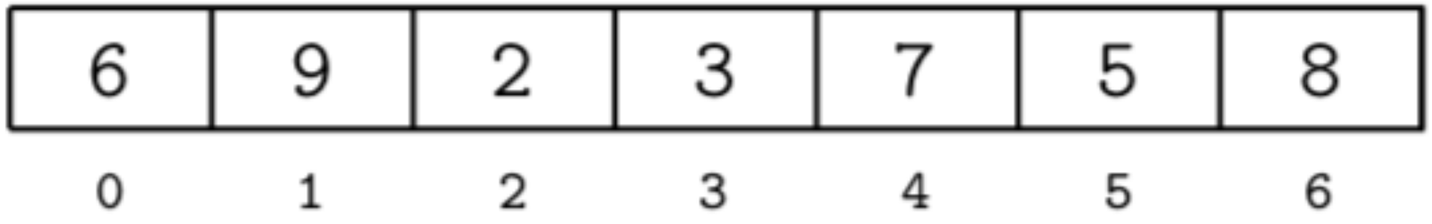
- Locating a cell, and storing or retrieving data at that cell is very fast.

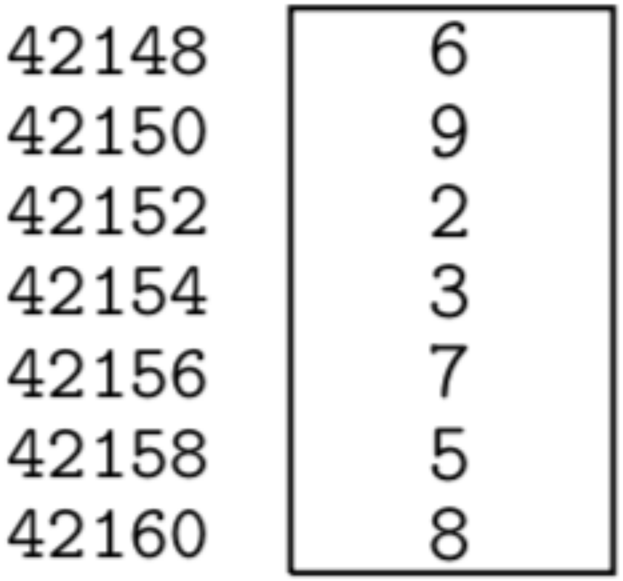
- The downside of an array is that maintai iguous bank of cells with information can be difficult and time-con

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



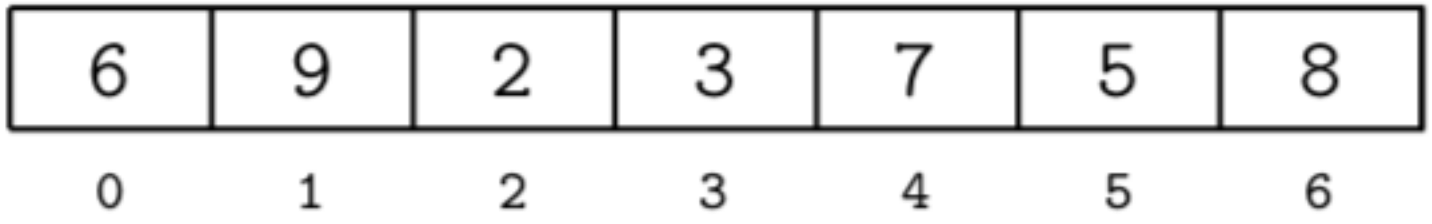


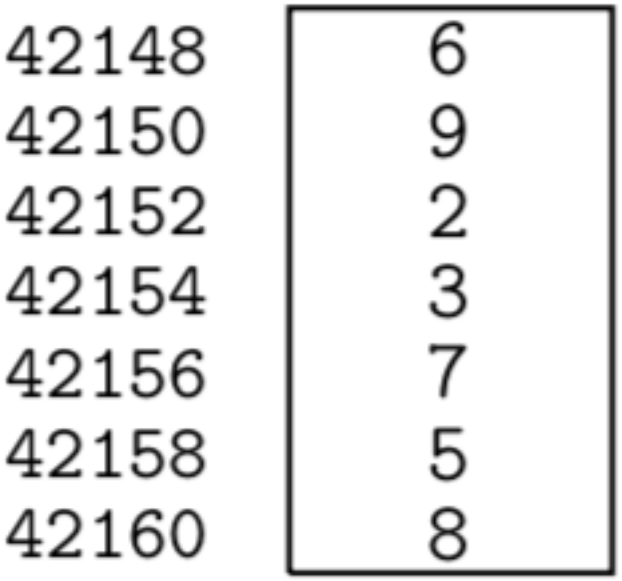
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Array

- An array corresponds to a sequence of consecutive cells in memory.
- Depending on programming language: `A[0]` up to `A[n-1]`, or `A[1]` up to `A[n]`.
- Locating a cell, and storing or retrieving data at that cell is very fast.
- The downside of an array is that maintai iguous bank of cells with information can be difficult and time-con

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro





How many bytes does each integer occupy here?

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Array

THE UNIVERSITY OF MELBOURNE

- An array corresponds to a sequence of consecutive cells in memory.
- Depending on programming language: `A[0]` up to `A[n-1]`, or `A[1]` up to `A[n]`.
- Locating a cell, and storing or retrieving data at that cell is very fast.

Assignment Project Exam Help

https://eduassistpro.github.io/

- The downside of an array is that maintai iguous bank of cells with information can be difficult and time-con

Add WeChat edu_assist_pro

6	9	2	3	7	5	8
0	1	2	3	4	5	6

42148	6
42150	9
42152	2
42154	3
42156	7
42158	5
42160	8

How many bytes does each integer occupy here?

Answer: 2 (16-bit integers)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

An array **x**:

2	3	5	7

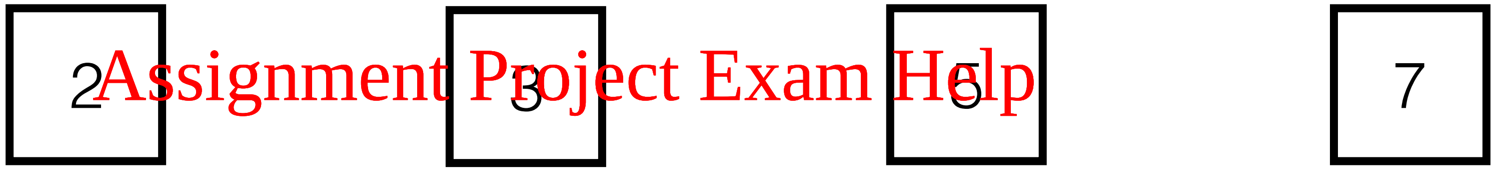
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

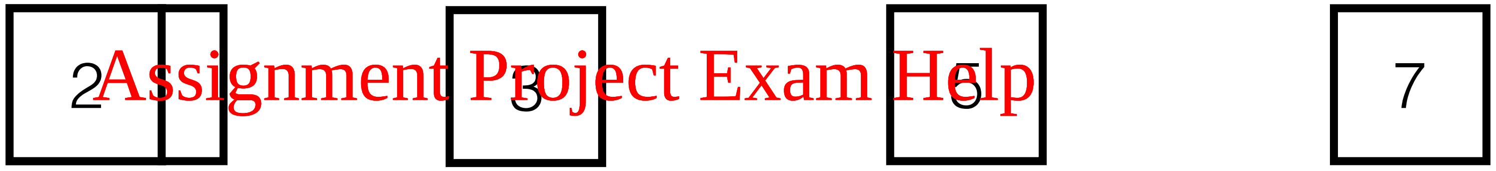
Primitive Data Structures: The Linked List



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

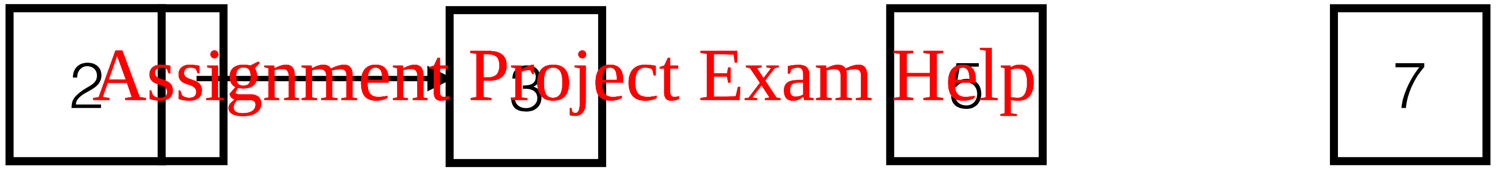
Primitive Data Structures: The Linked List



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

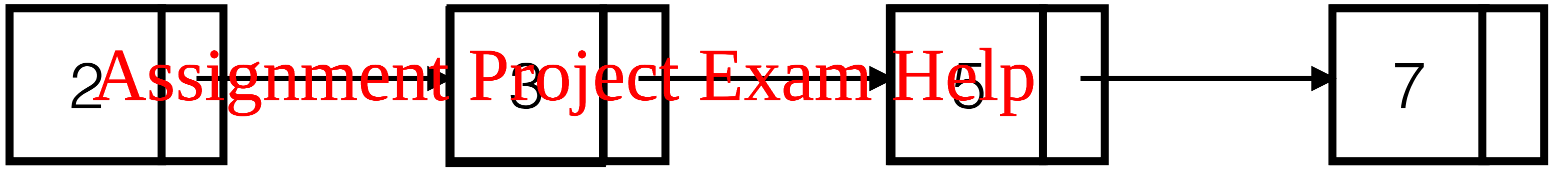
Primitive Data Structures: The Linked List



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List



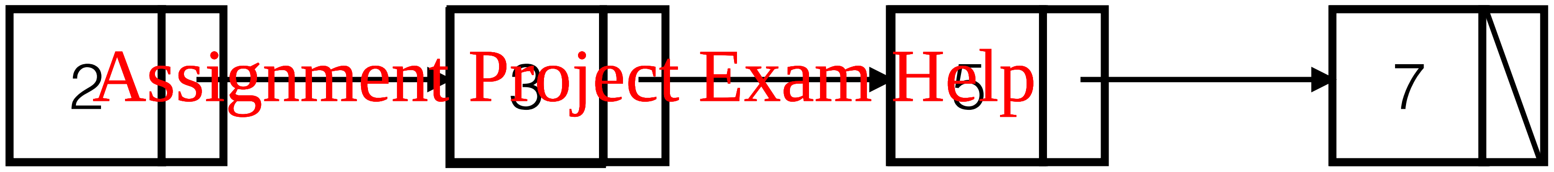
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

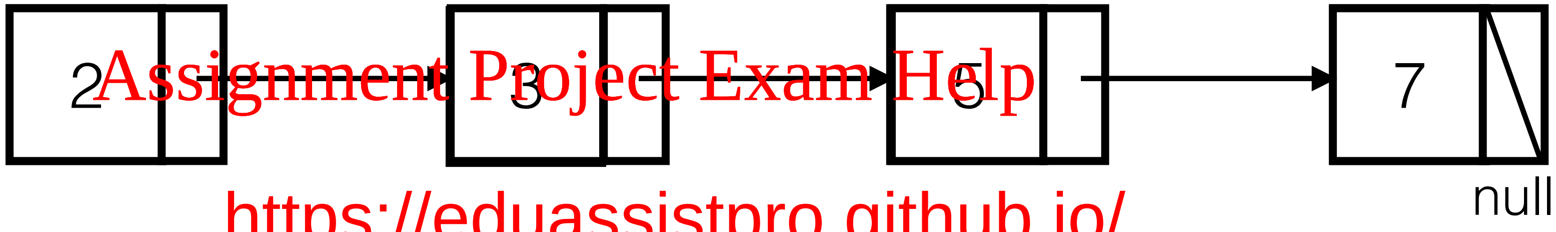
Primitive Data Structures: The Linked List



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

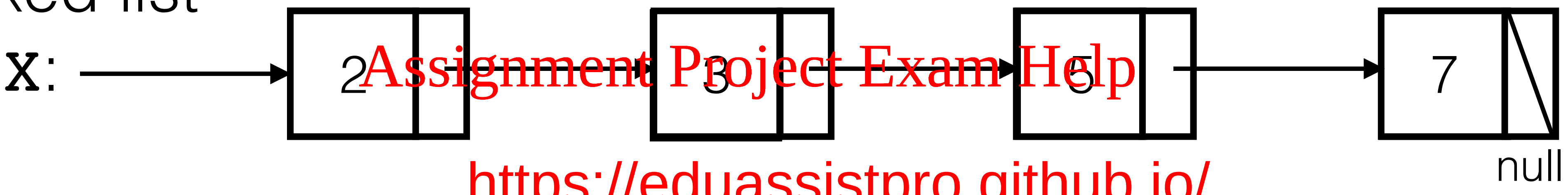


Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

A linked list



Assignment Project Exam Help

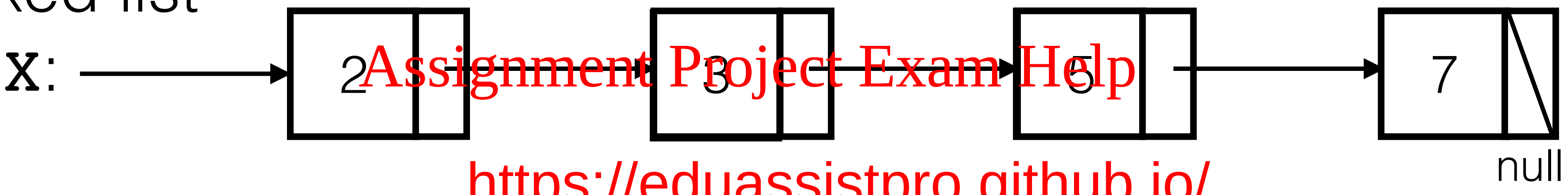
https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

A linked list



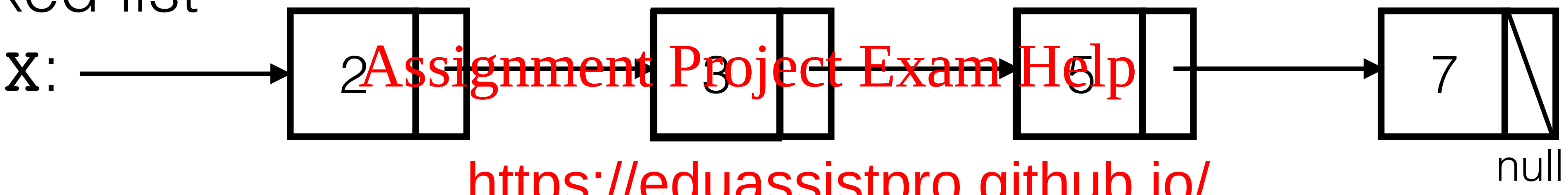
Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Suppose variable **x** holds the address 42160, then the list could look like this in memory:

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

A linked list



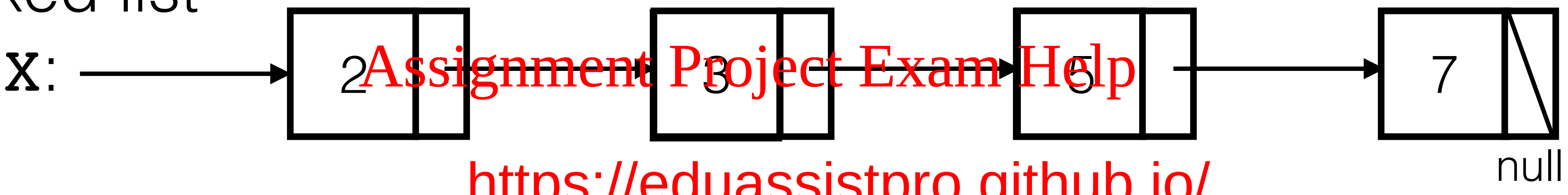
Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Suppose variable **x** holds the address 42160, then the list could look like this in memory:

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

A linked list



Assignment Project Exam Help

https://eduassistpro.github.io/

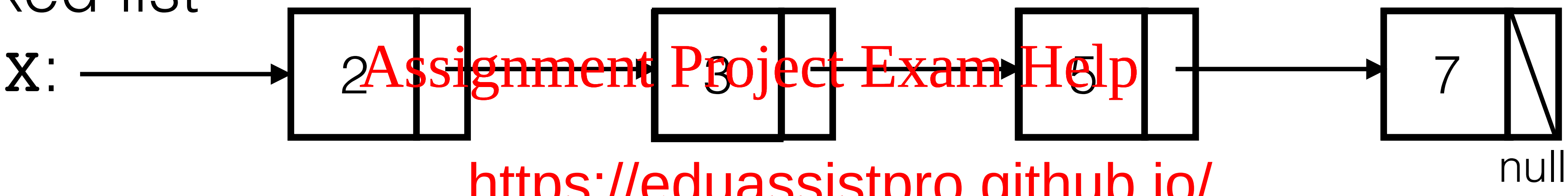
Add WeChat edu_assist_pro

Suppose variable **x** holds the address 42160, then the list could look like this in memory:

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

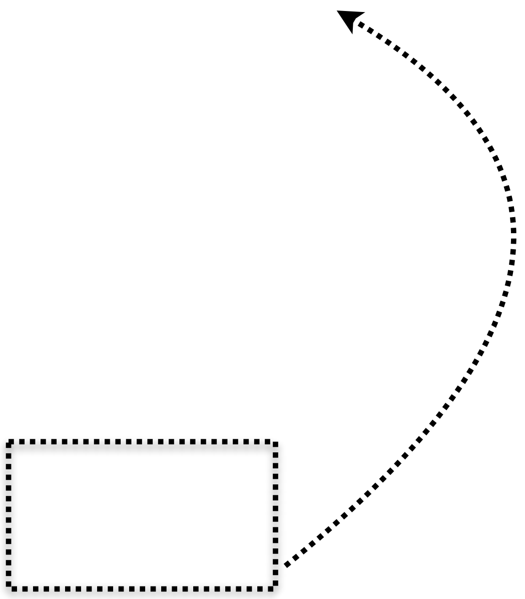
Primitive Data Structures: The Linked List

A linked list



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

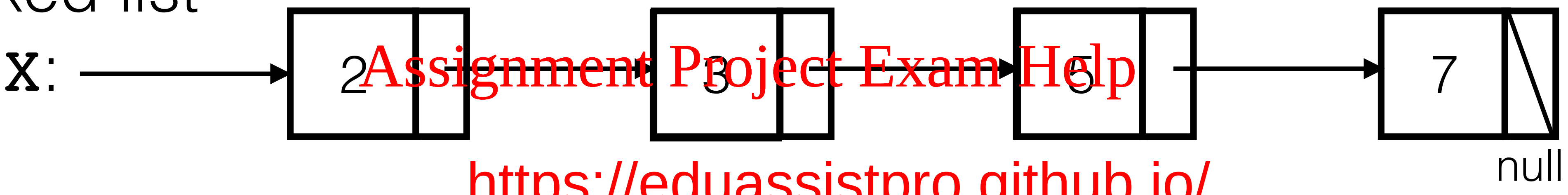
Suppose variable **x** holds the address 42160, then the list could look like this in memory:



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

A linked list

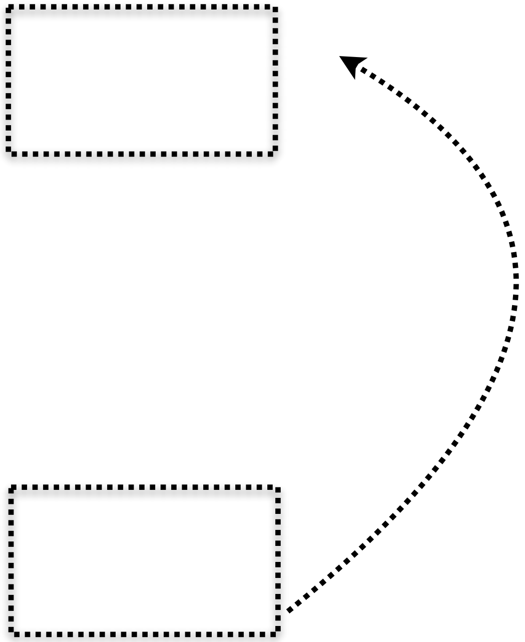


Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

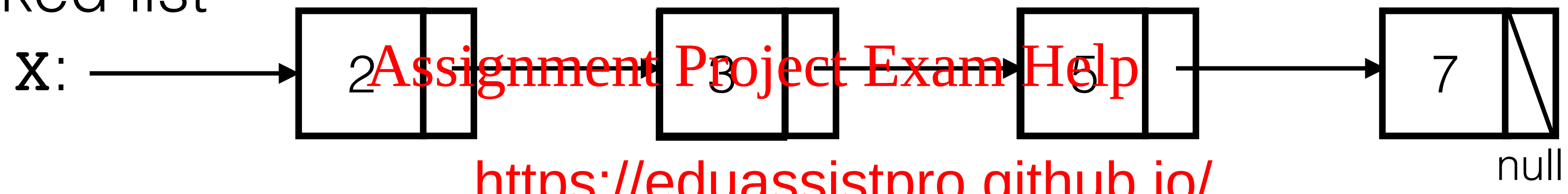
Suppose variable **x** holds the address 42160, then the list could look like this in memory:



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

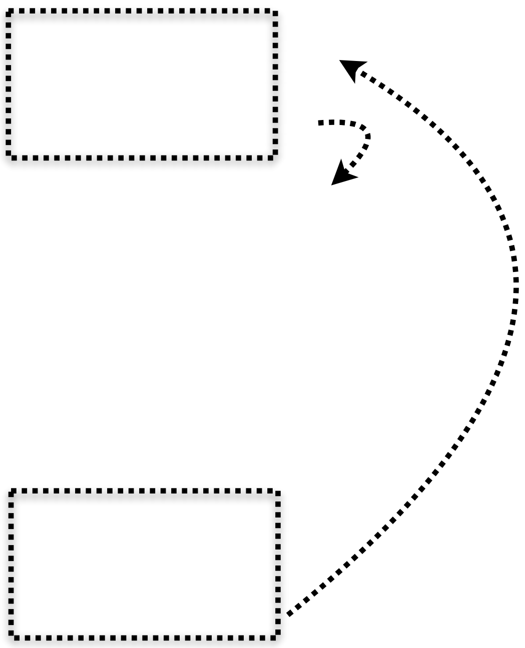
Primitive Data Structures: The Linked List

A linked list



Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

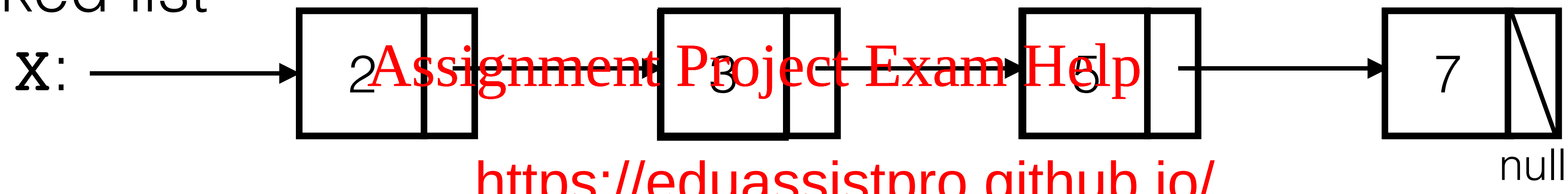
Suppose variable **x** holds the address 42160, then the list could look like this in memory:



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

A linked list

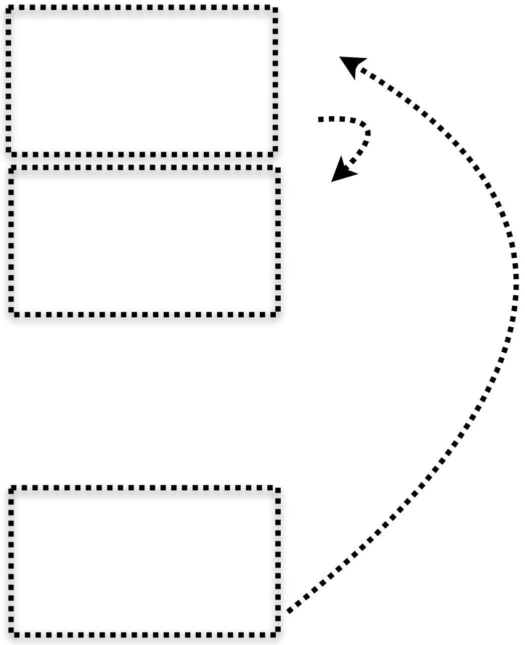


Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

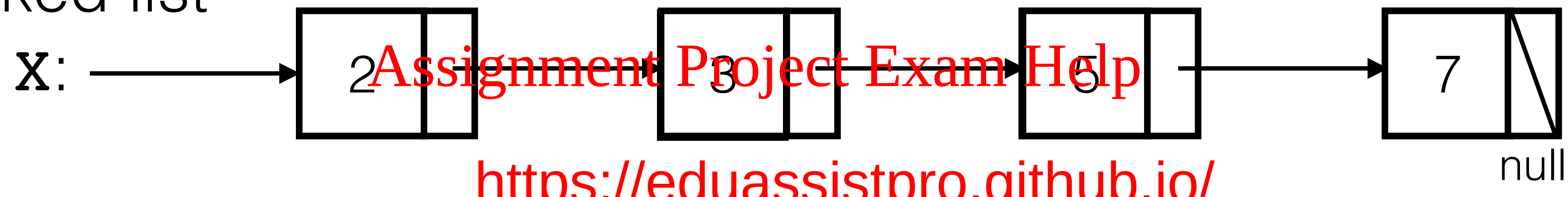
Suppose variable x holds the address 42160, then the list could look like this in memory:



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

A linked list

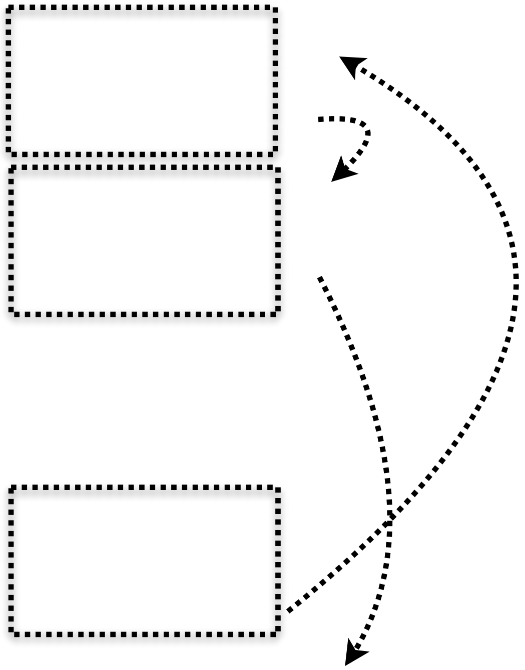


Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

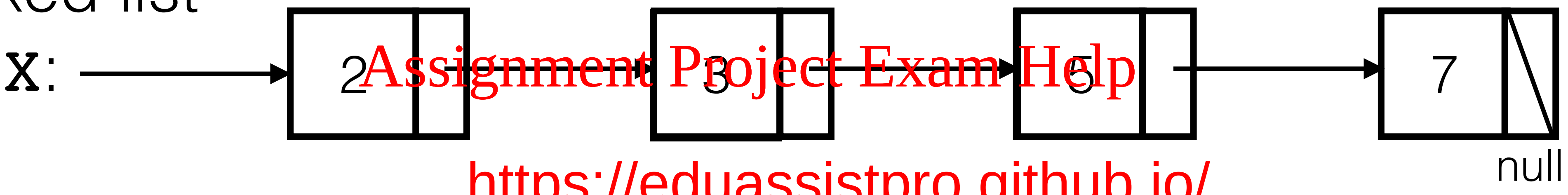
Suppose variable **x** holds the address 42160, then the list could look like this in memory:



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Primitive Data Structures: The Linked List

A linked list

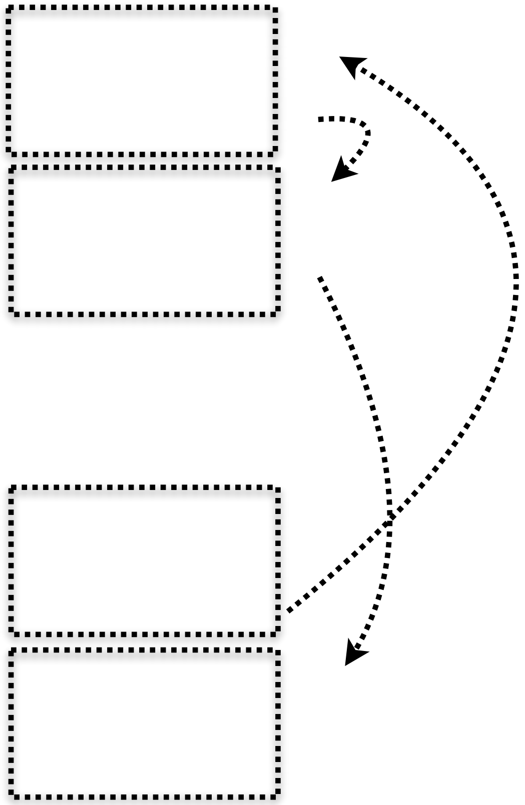


Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Suppose variable **x** holds the address 42160, then the list could look like this in memory:



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology

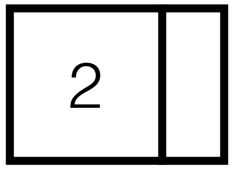
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology



Assignment Project Exam Help

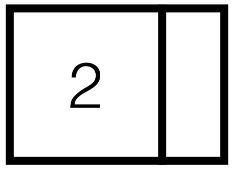
https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology

node



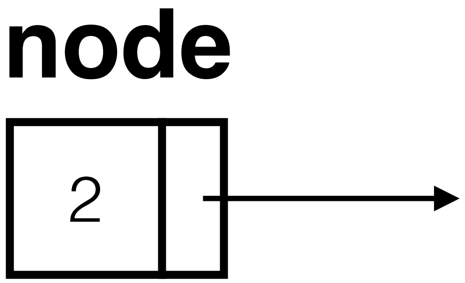
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology



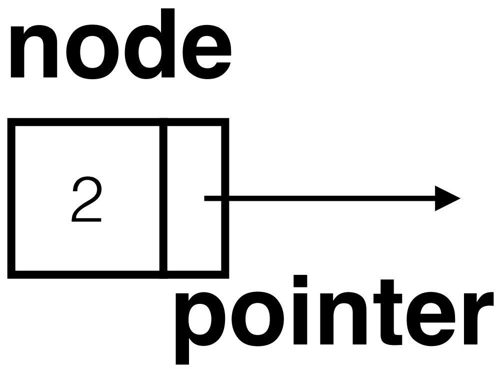
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology



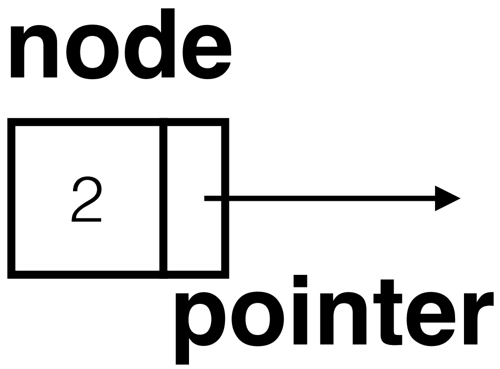
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology



(in Java: “reference”)

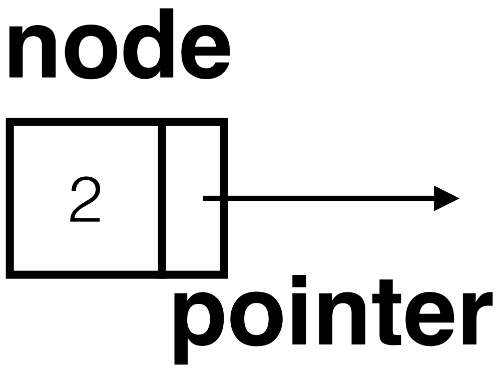
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology

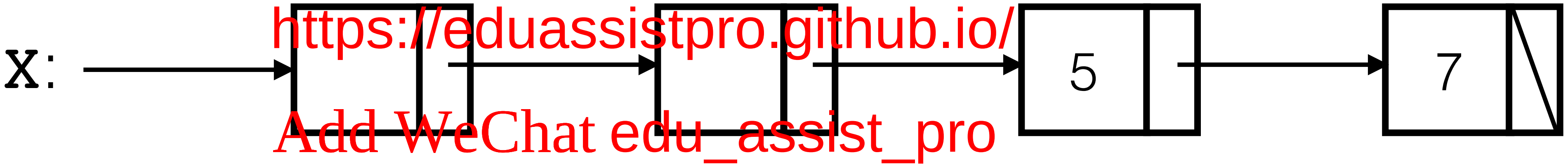


(in Java: “reference”)

Assignment Project Exam Help

https://eduassistpro.github.io/

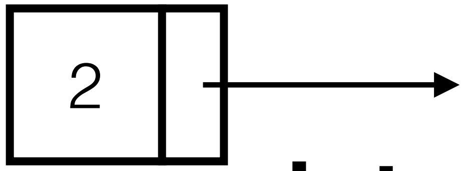
Add WeChat edu_assist_pro



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology

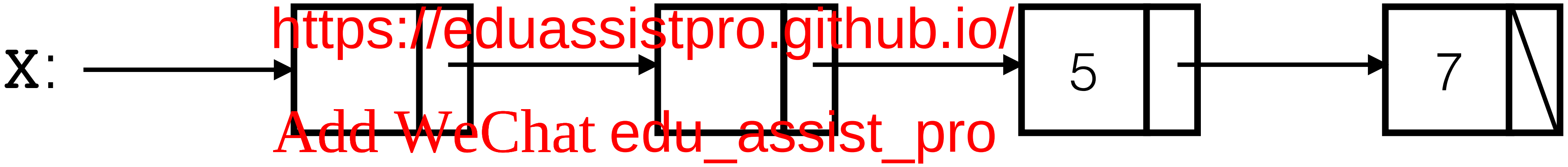
node



pointer

(in Java: “reference”)

Assignment Project Exam Help



https://eduassistpro.github.io/

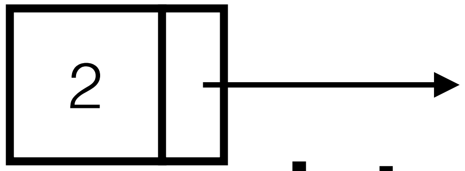
Add WeChat edu_assist_pro

`x` is (a pointer to) the **head node** of the list

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology

node



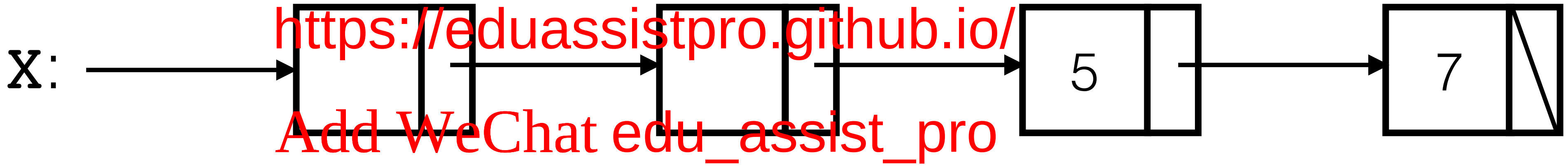
pointer

(in Java: “reference”)

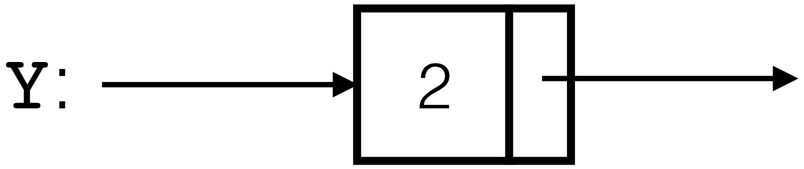
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

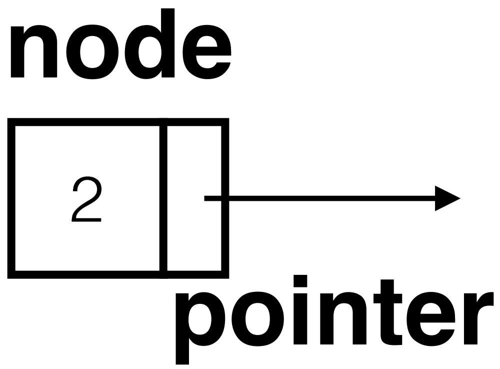


X is (a pointer to) the **head node** of the list



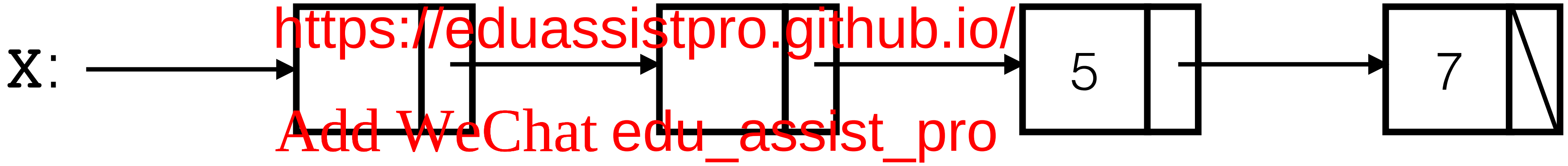
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology



(in Java: “reference”)

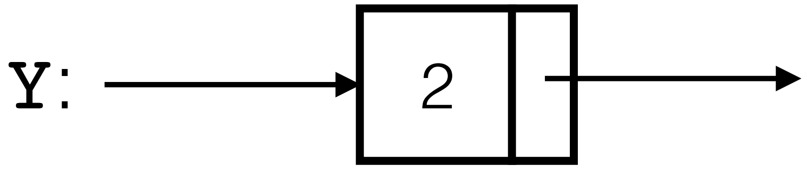
Assignment Project Exam Help



https://eduassistpro.github.io/

Add WeChat edu_assist_pro

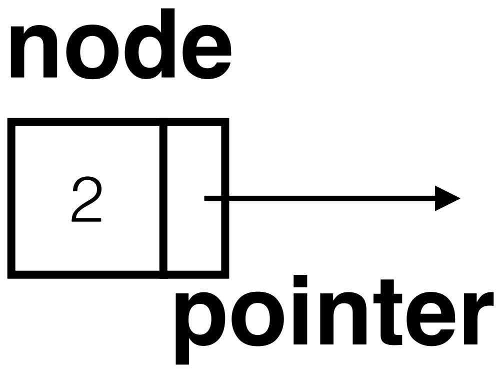
`X` is (a pointer to) the **head node** of the list



“`Y.val`” *refers to*

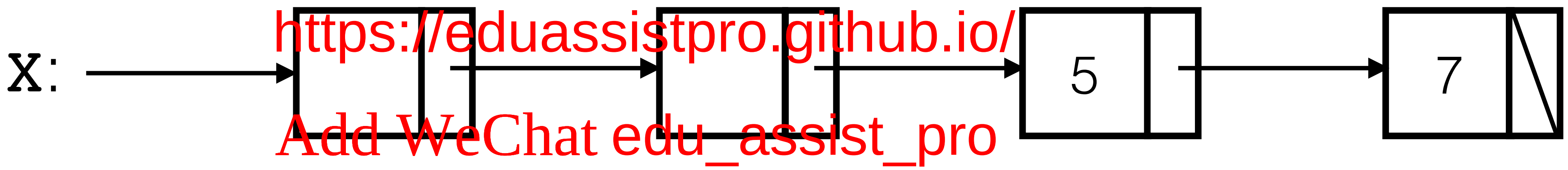
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology

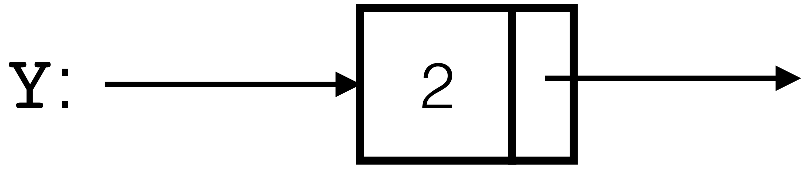


(in Java: “reference”)

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro



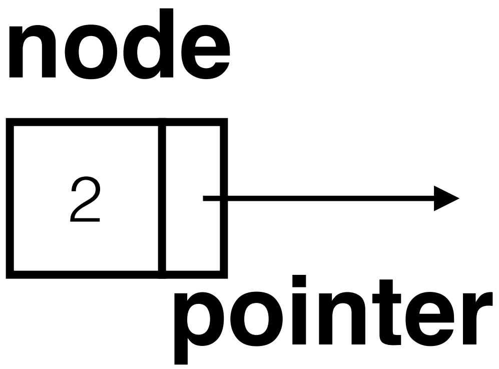
`X` is (a pointer to) the **head node** of the list



“`Y.val`” *refers to*

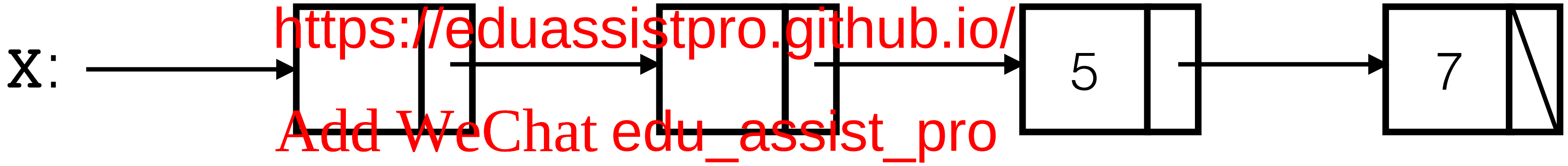
Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology



(in Java: “reference”)

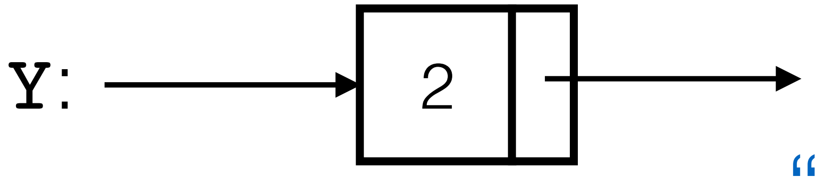
Assignment Project Exam Help



https://eduassistpro.github.io/

Add WeChat edu_assist_pro

`X` is (a pointer to) the **head node** of the list



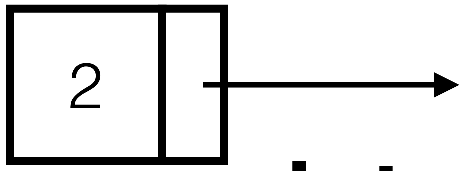
“`Y.next`” *refers to*

“`Y.val`” *refers to*

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Terminology

node



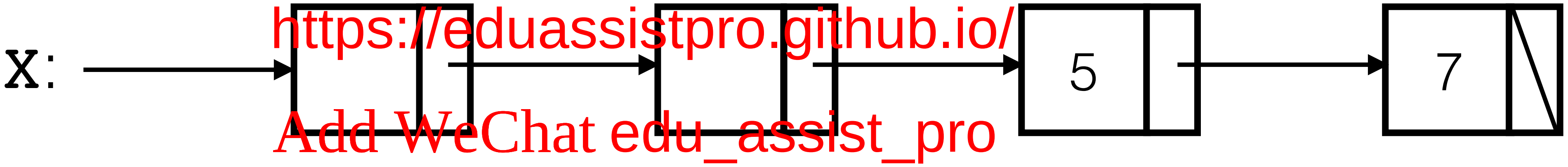
pointer

(in Java: “reference”)

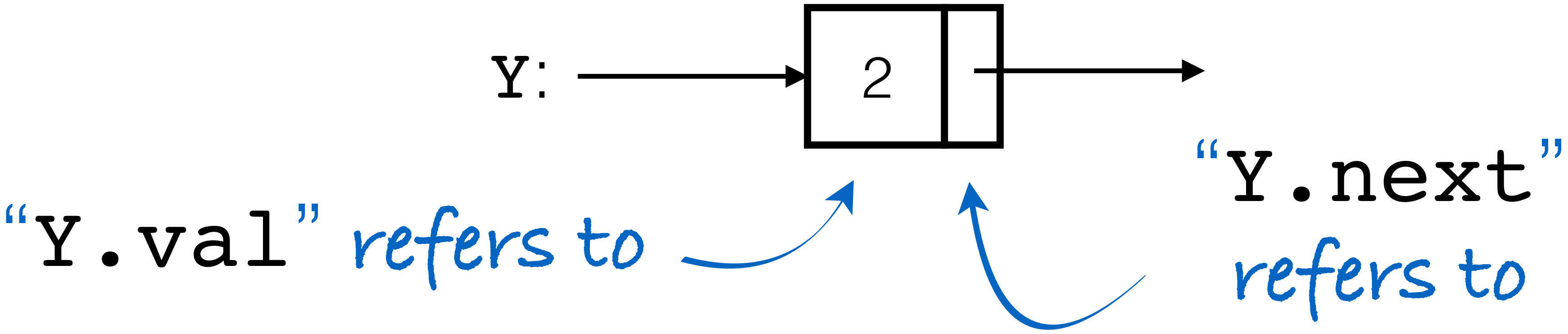
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



`X` is (a pointer to) the **head node** of the list



“`Y.val`” *refers to*

“`Y.next`” *refers to*

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Linked List

THE UNIVERSITY OF MELBOURNE

- Often we use a dummy head node that points to the first object, or to a special `null` object that represents an empty list. This makes it easier to write functions that insert

- Inserting and deleting elements is very f ve a few links around.

- Finding the *i*th element can be time-consuming.

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)

- For example, to locate item **x**.

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)

- For example, to locate item **x**.

```
function find(A,x,n)
    $j \leftarrow 0$
    while $j < n$
        if $A[j] = x$
            return j
        $j \leftarrow j+1$
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)
- For example, to locate item **x**.

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)
- For example, to locate item **x**.

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Y:	6	9	2	3	7	5	8
	0	1	2	3	4	5	6

Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)

- For example, to locate item **x**.

A: Y

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)

- For example, to locate item **x**.

A: Y   x: 7

```
function find(A,x,n)
    $j \leftarrow 0$
    while $j < n$
        if $A[j] = x$
            return j
        $j \leftarrow j+1$
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Y:	6	9	2	3	7	5	8
	0	1	2	3	4	5	6

Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)

- For example, to locate item **x**.

A: Y   x: 7   n: 6

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)
- For example, to locate item **x**.

A: Y   x: 7   n: 6   j: 0

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Y:	6	9	2	3	7	5	8
	0	1	2	3	4	5	6

Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)
- For example, to locate item **x**.

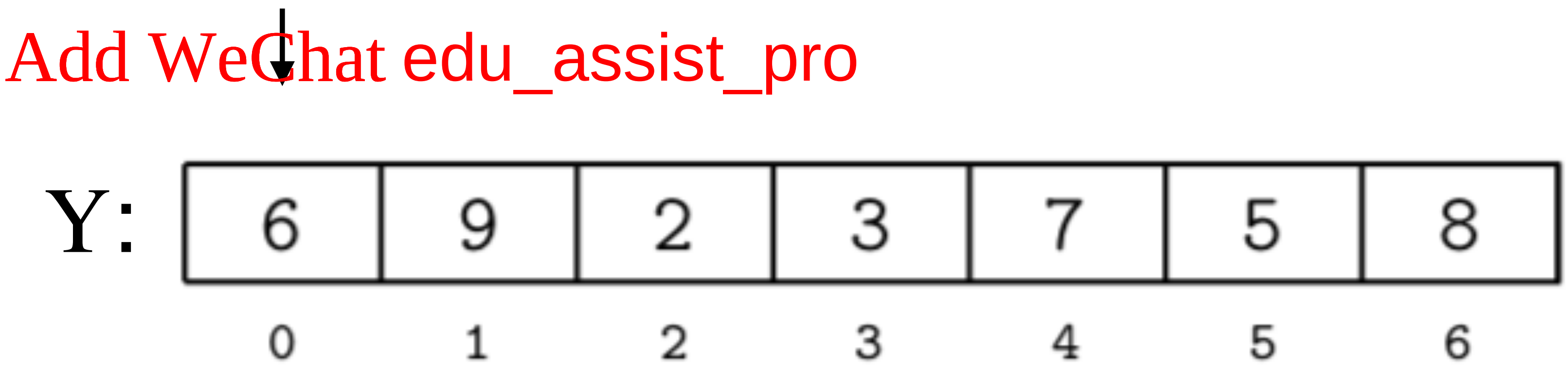
A: Y x: 7 n: 6 j: 0

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



Y:	6	9	2	3	7	5	8
	0	1	2	3	4	5	6

Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)

- For example, to locate item **x**.

A: Y x: 7 n: 6 j: 1

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Y:

	↓					
6	9	2	3	7	5	8
0	1	2	3	4	5	6

Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)
- For example, to locate item **x**.

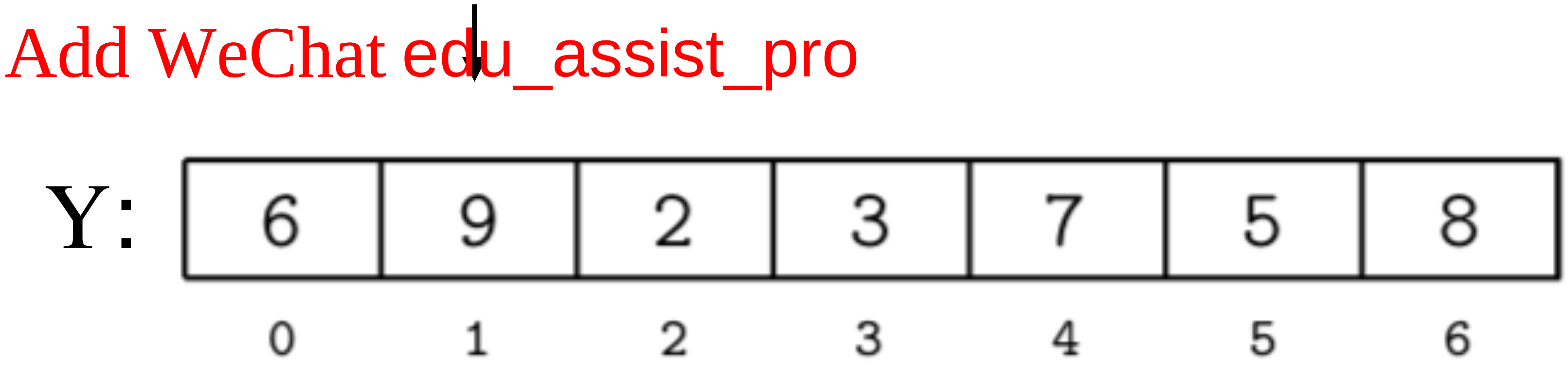
A: Y x: 7 n: 6 j: 1

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



Y:	6	9	2	3	7	5	8
	0	1	2	3	4	5	6

Let’s trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)
- For example, to locate item **x**.

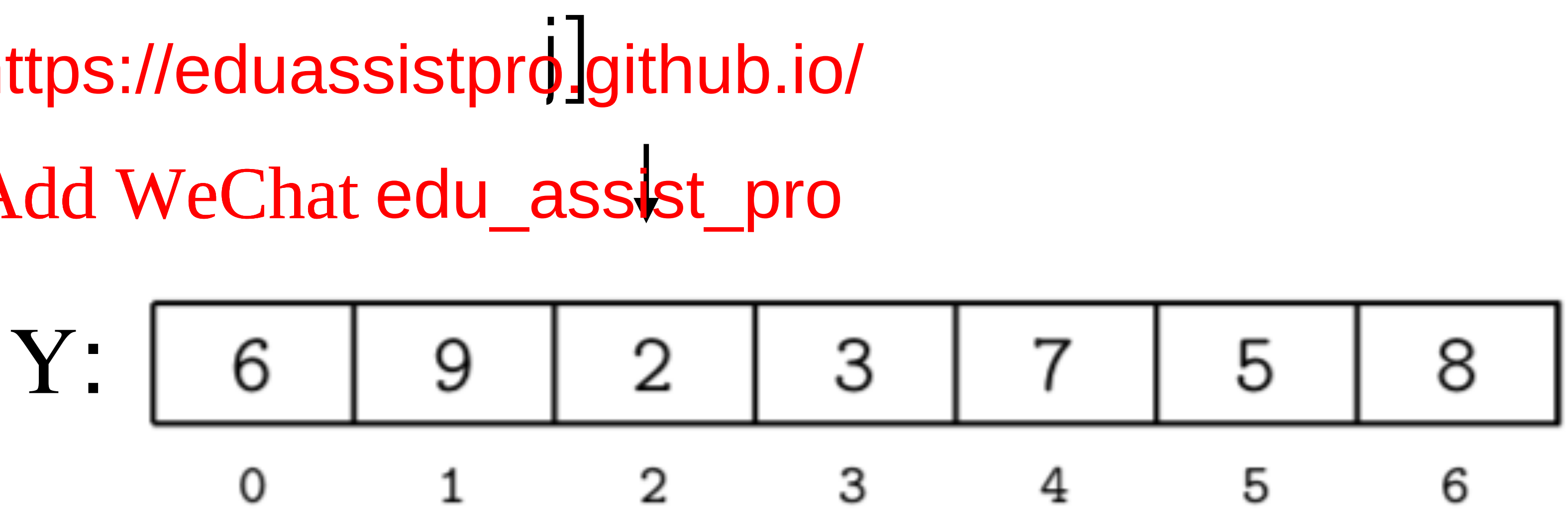
A: Y x: 7 n: 6 j: 2

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



0	1	2	3	4	5	6
6	9	2	3	7	5	8

Let’s trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)

- For example, to locate item **x**.

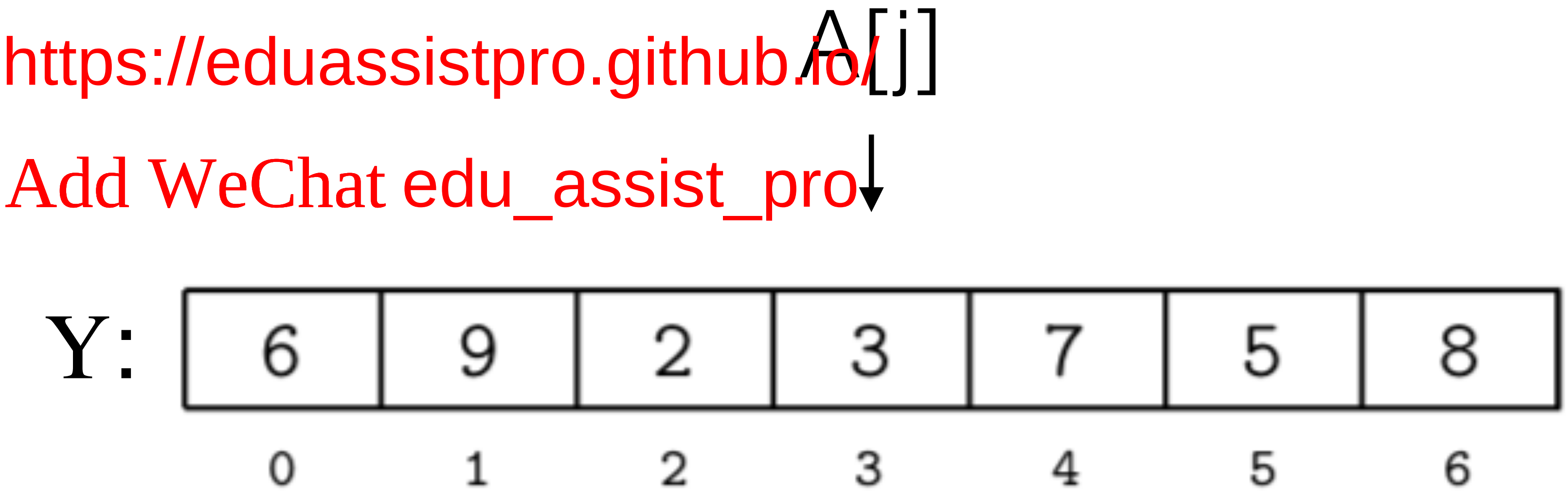
A: Y   x: 7   n: 6   j: 3

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



6	9	2	3	7	5	8
0	1	2	3	4	5	6

Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)
- For example, to locate item **x**.

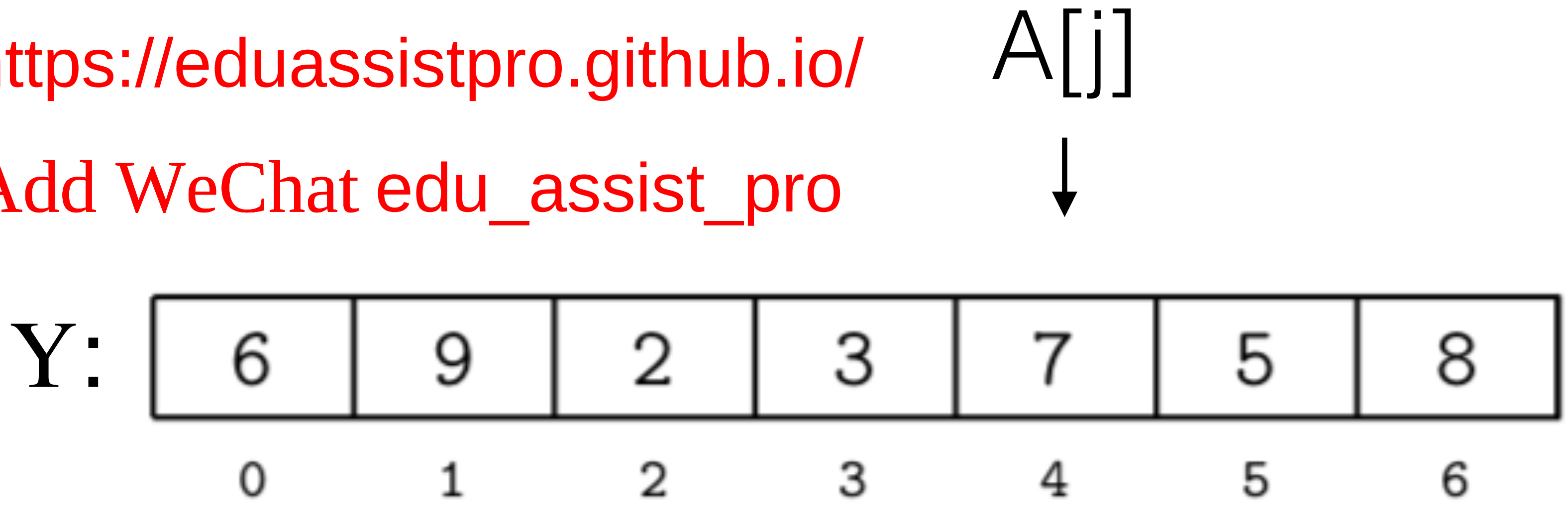
A: Y x: 7 n: 6 j: 4

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



A[j]

Y:	6	9	2	3	7	5	8
	0	1	2	3	4	5	6

Let's trace the execution of find(Y,7,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: Array

- Walk through the array (of length n)
- For example, to locate item **x**.

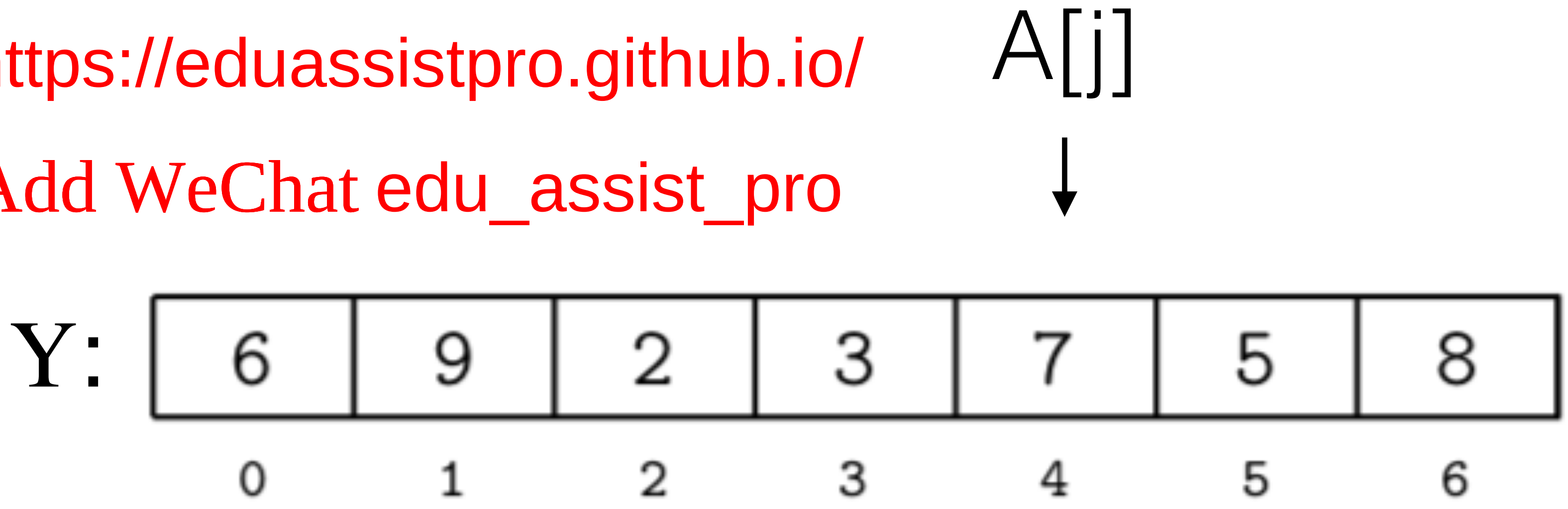
A: Y   x: 7   n: 6   j: 4

```
function find(A,x,n)
    j ← 0
    while j < n
        if A[j] = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



0	1	2	3	4	5	6
6	9	2	3	7	5	8

Let's trace the execution of find(Y,7,6)

(returns 4)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.

- For example, to locate item **x**.

```
function find(head,x)
    p ← head
    while p ≠ null
        if p.val = x
            return p
        p ← p.next
    return null
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.

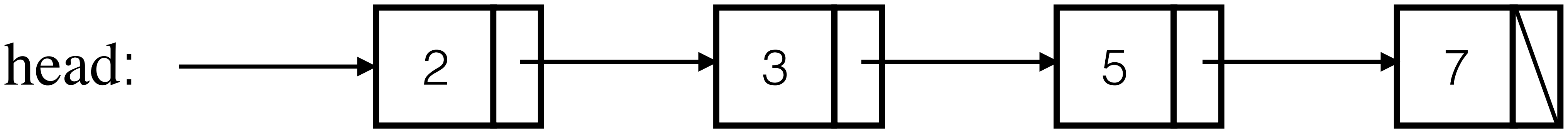
- For example, to locate item **x**.

```
function find(head,x)
    p ← head
    while p ≠ null
        if p.val = x
            return p
        p ← p.next
    return null
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.
- For example, to locate item **x**.

(note similarity to array version)

```
function find(head,x)
    p ← head
    while p ≠ null
        if p.val = x
            return p
        p ← p.next
    return null
```

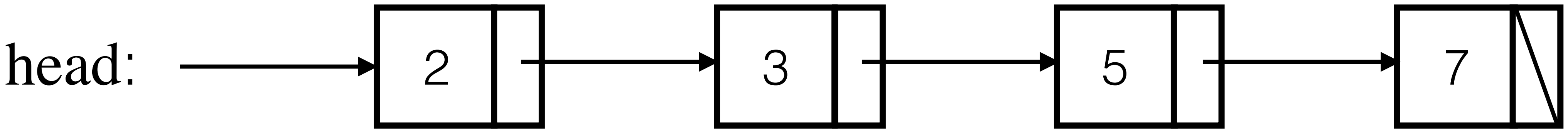
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

```
function find(A,x,n)

                n
              = x
            return j
        j ← j+1
    return -1
```



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.
- For example, to locate item **x**.

(note similarity to array version)

```
function find(head,x)
  p ← head
  while p ≠ null
    if p.val = x
      return p
    p ← p.next
  return null
```

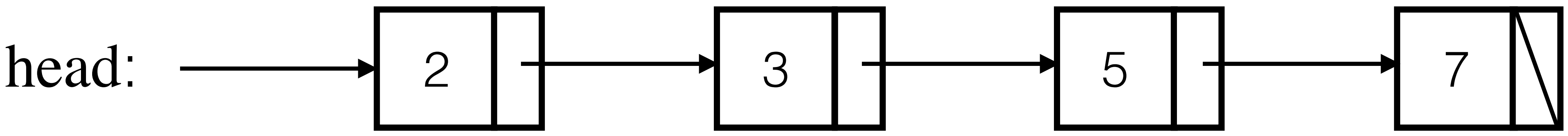
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

```
function find(A,x,n)

         n
         = x
      return j
    j ← j+1
  return -1
```



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.

- For example, to locate item **x**.

(note similarity to array version)

```
function find(head,x)
    p ← head
    while p ≠ null
        if p.val = x
            return p
        p ← p.next
    return null
```

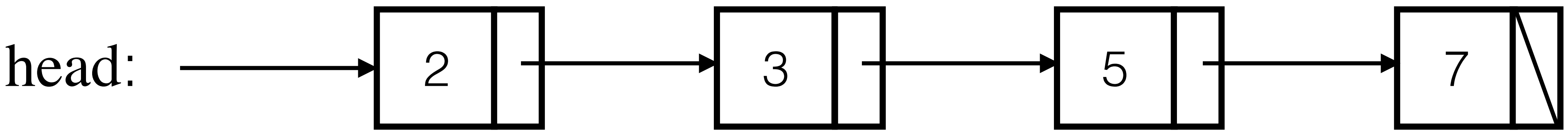
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

```
function find(A,x,n)
                  n
                  = x
            return j
        j ← j+1
    return -1
```

p:



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.
- For example, to locate item **x**.

(note similarity to array version)

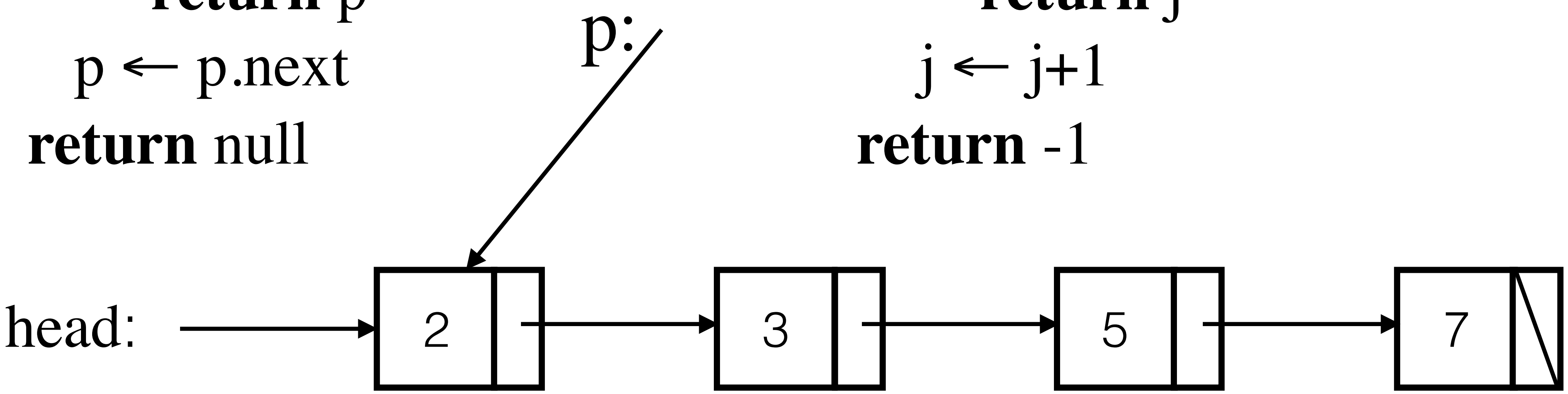
```
function find(head,x)
    p ← head
    while p ≠ null
        if p.val = x
            return p
        p ← p.next
    return null
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

```
function find(A,x,n)
                n
                = x
            return j
        j ← j+1
    return -1
```



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.
- For example, to locate item **x**.

(note similarity to array version)

```
function find(head,x)
    p ← head
    while p ≠ null
        if p.val = x
            return p
        p ← p.next
    return null
```

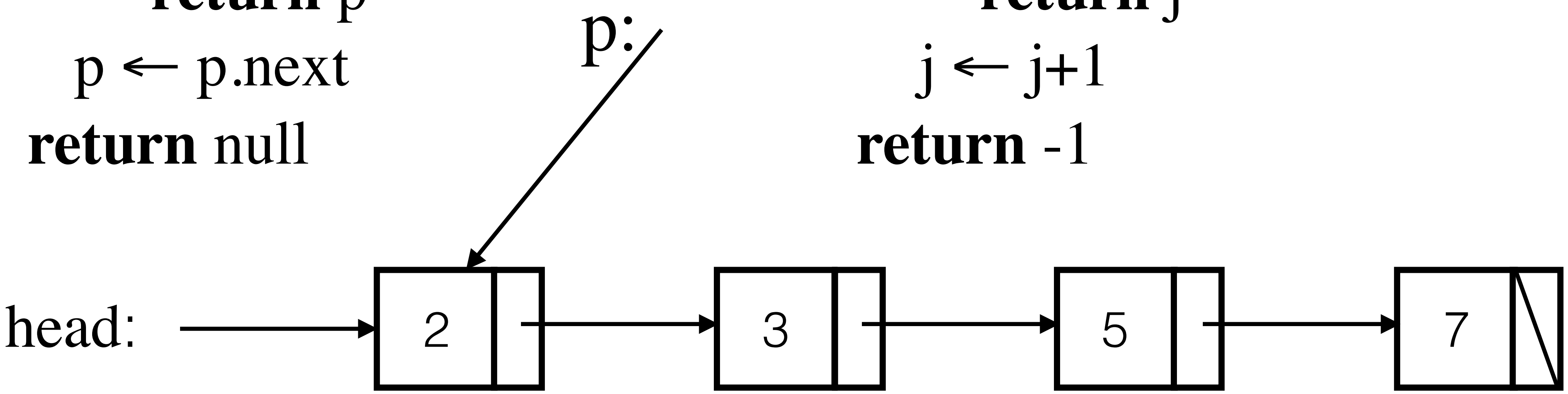
```
function find(A,x,n)

                n
                = x
            return j
        j ← j+1
    return -1
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.
- For example, to locate item **x**.

(note similarity to array version)

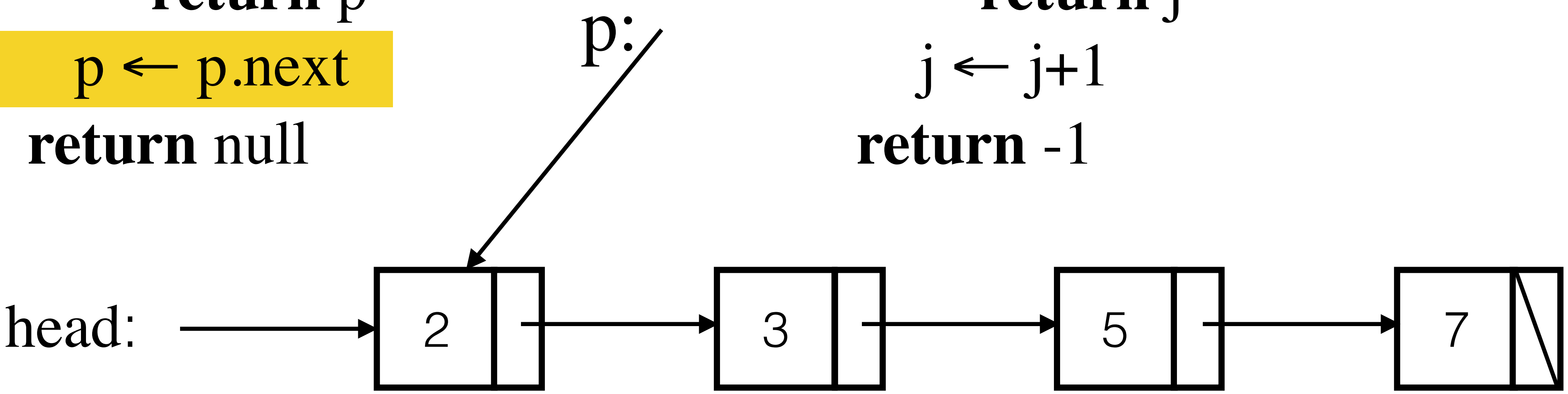
```
function find(head,x)
    p ← head
    while p ≠ null
        if p.val = x
            return p
        p ← p.next
    return null
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

```
function find(A,x,n)
                     n
                     = x
            return j
        j ← j+1
    return -1
```



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.

- For example, to locate item **x**.

(note similarity to array version)

```
function find(head,x)
    p ← head
    while p ≠ null
        if p.val = x
            return p
        p ← p.next
    return null
```

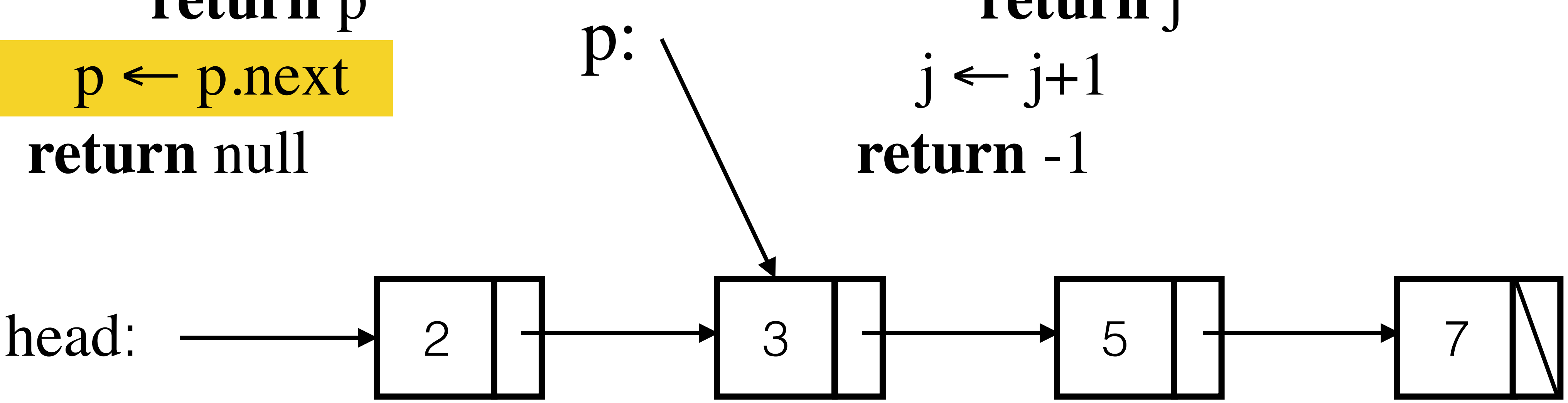
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

```
function find(A,x,n)

                n
                = x
            return j
        j ← j+1
    return -1
```



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Iterative Processing: List

- Walk through a linked list.

- For example, to locate item **x**.

(note similarity to array version)

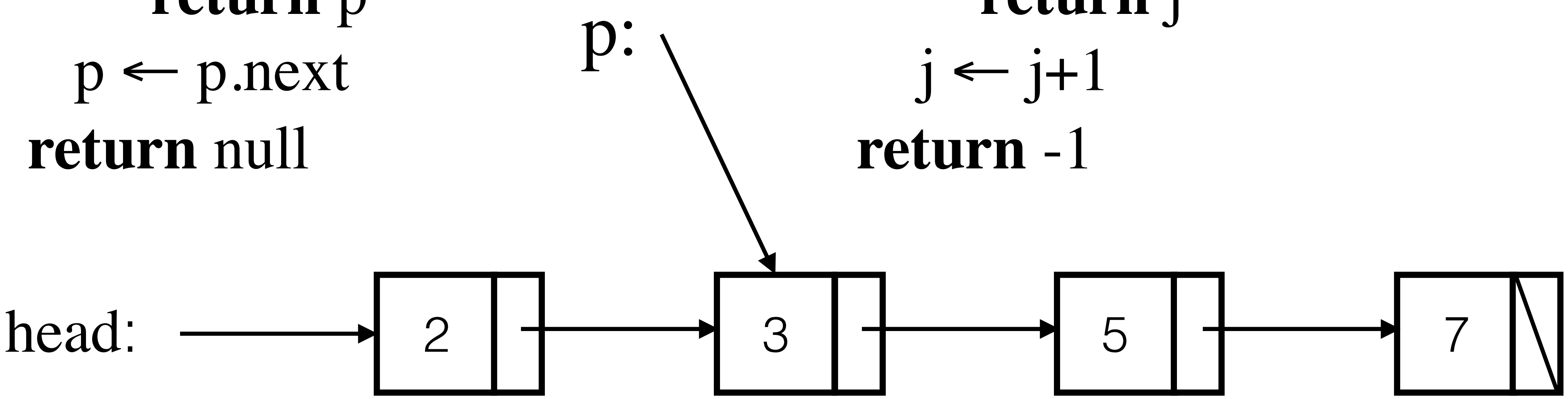
```
function find(head,x)
    p ← head
    while p ≠ null
        if p.val = x
            return p
        p ← p.next
    return null
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

```
function find(A,x,n)
            n
           = x
        return j
    j ← j+1
return -1
```



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item `x`.

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item `x`.

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Initial call: find(A,x,0,n-1)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Y:	6	9	2	3	7	5	8
	0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Y:	6	9	2	3	7	5	8
	0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

A: Y

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Y:	6	9	2	3	7	5	8
	0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

A: Y   x: 7

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Y:	6	9	2	3	7	5	8
	0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

A: Y x: 7 lo: 0

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

A: Y   x: 7   lo: 0   hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Y:

6	9	2	3	7	5	8
0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

A: Y   x: 7   lo: 0   hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Y:	6	9	2	3	7	5	8
	0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let’s trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

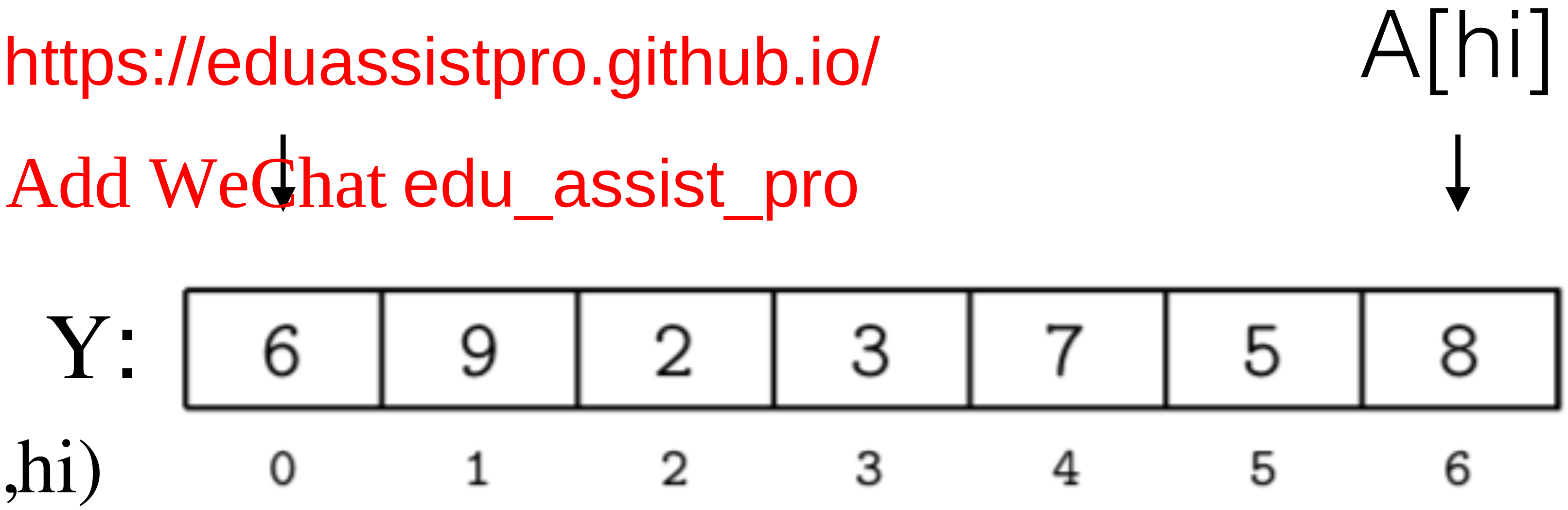
A: Y x: 7 lo: 0 hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

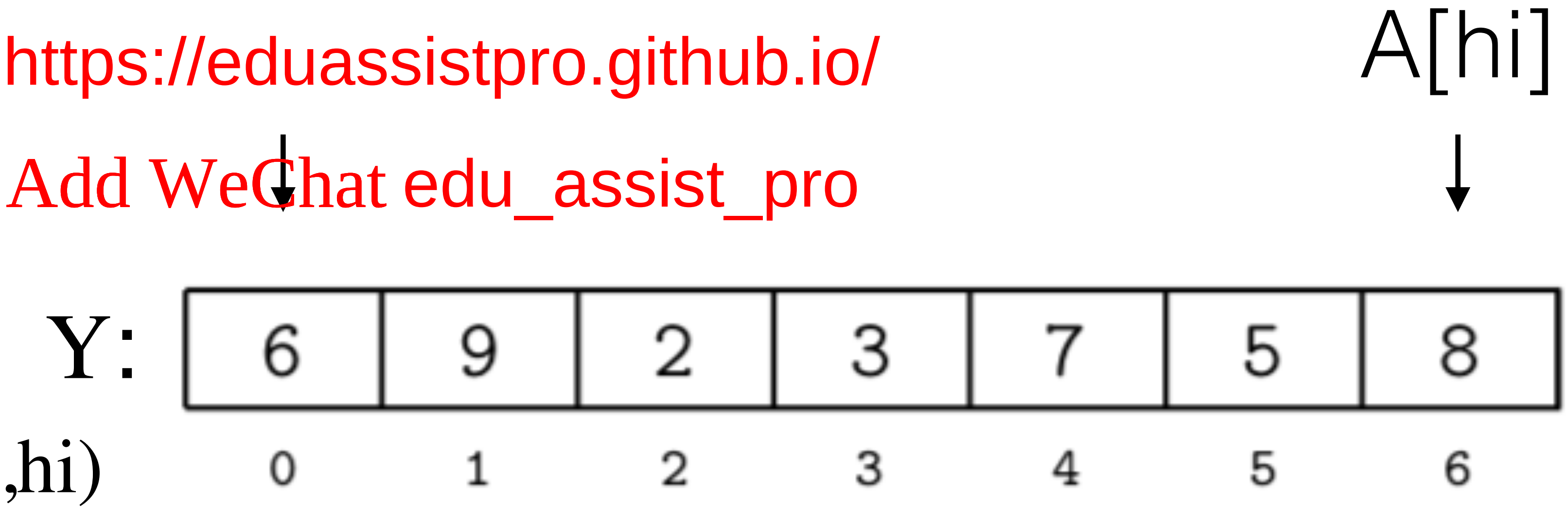
A: Y x: 7 lo: 1 hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



6	9	2	3	7	5	8
0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

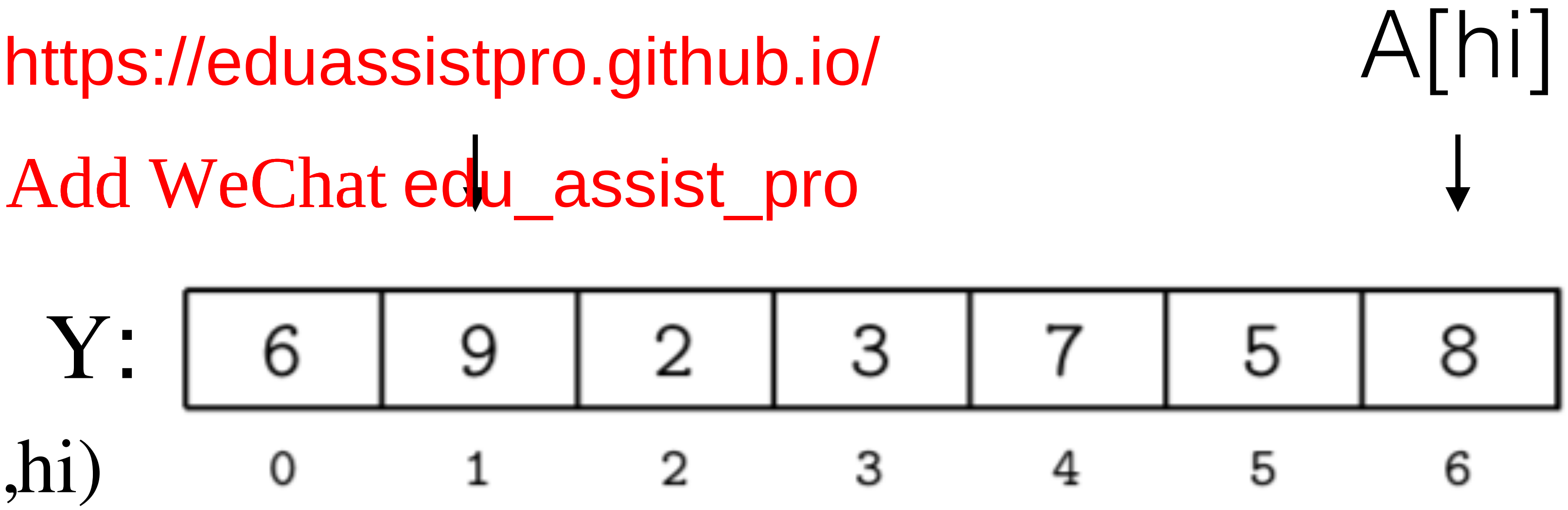
A: Y x: 7 lo: 1 hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



A[hi]

Y:	6	9	2	3	7	5	8
	0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

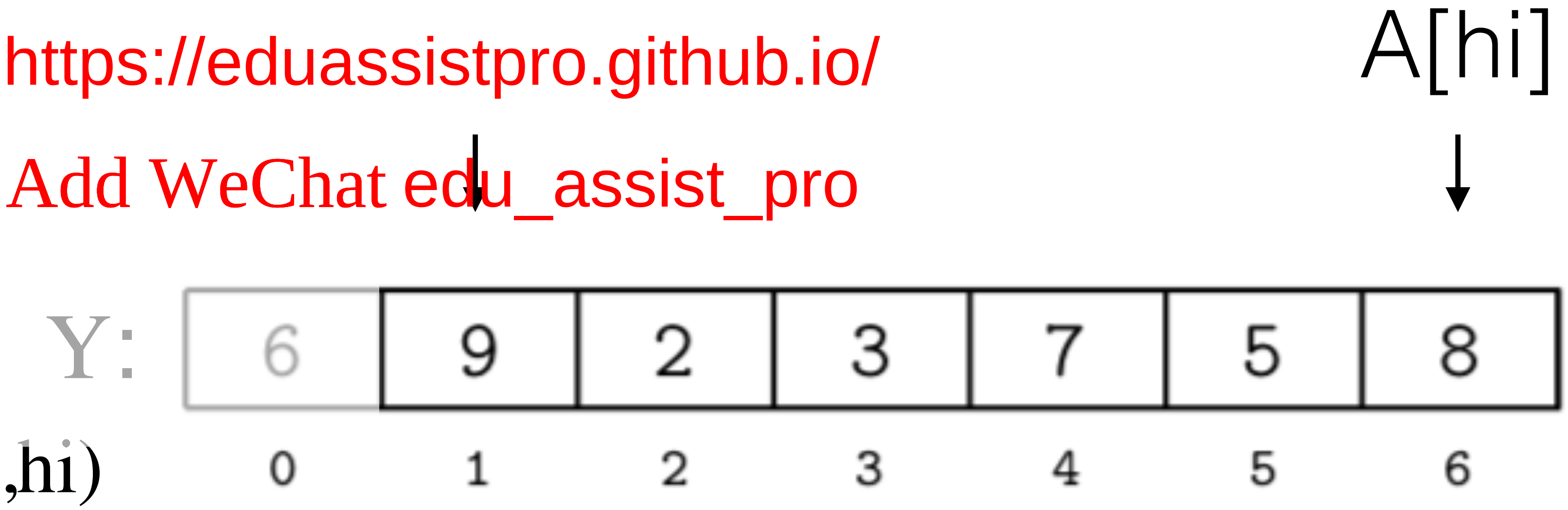
A: Y x: 7 lo: 1 hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



6	9	2	3	7	5	8
0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

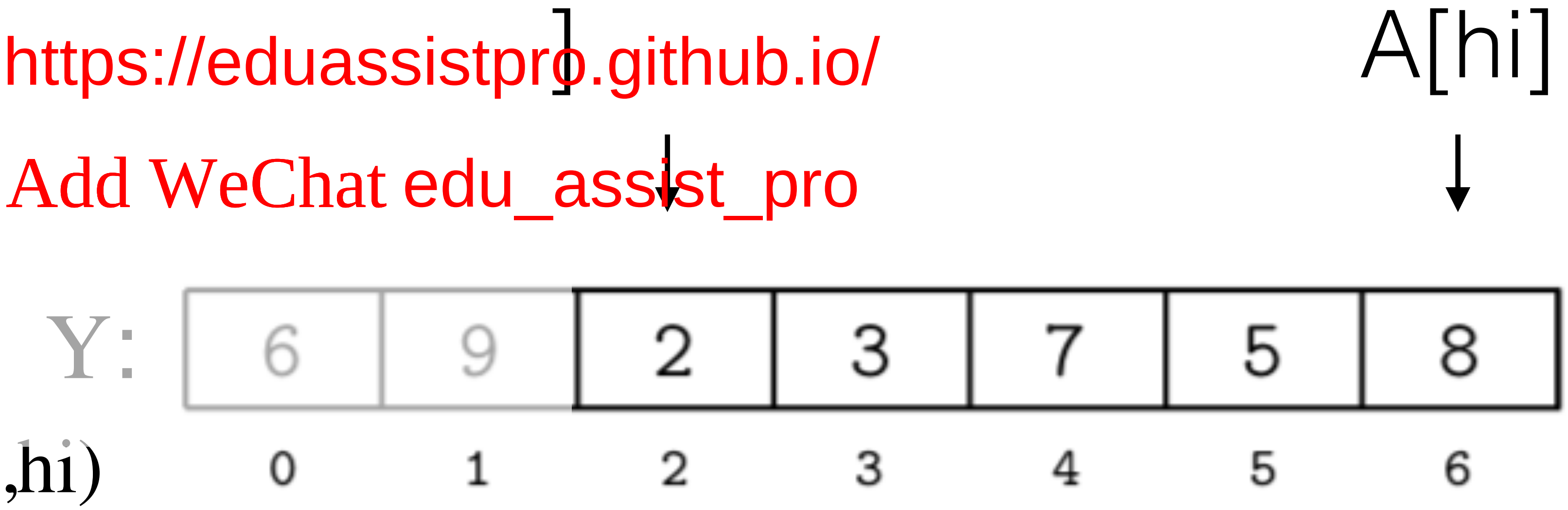
A: Y x: 7 lo: 2 hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



Y:	6	9	2	3	7	5	8
	0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

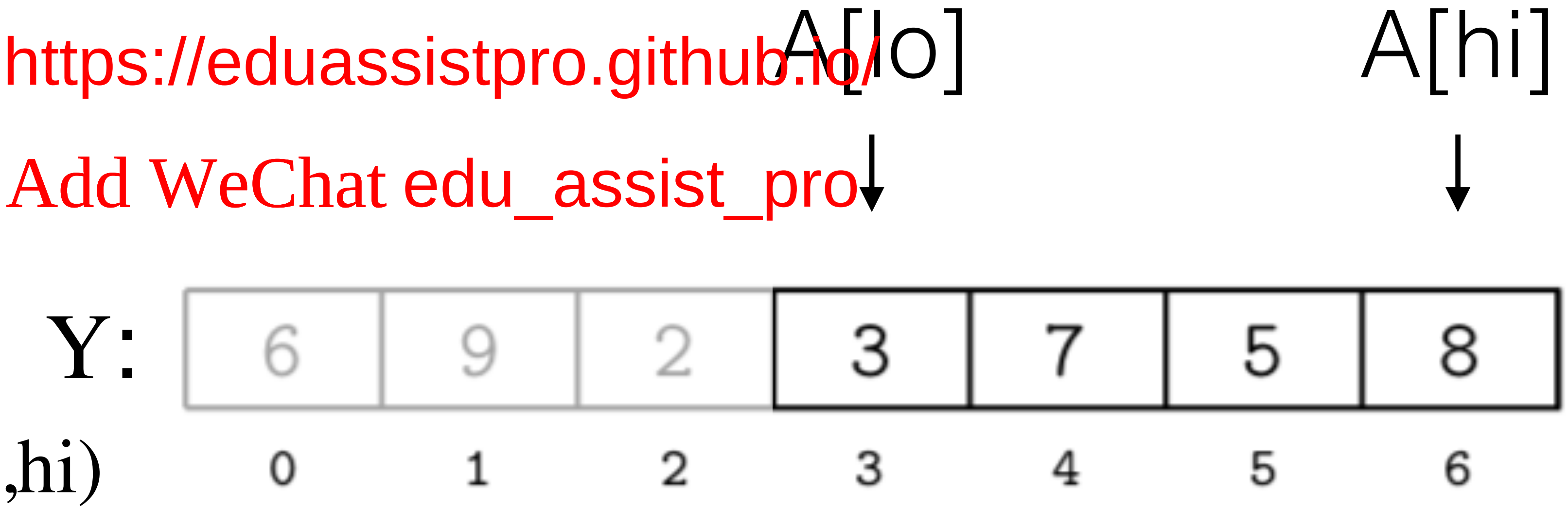
A: Y x: 7 lo: 3 hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



				A[lo]			A[hi]
Y:	6	9	2	3	7	5	8
	0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

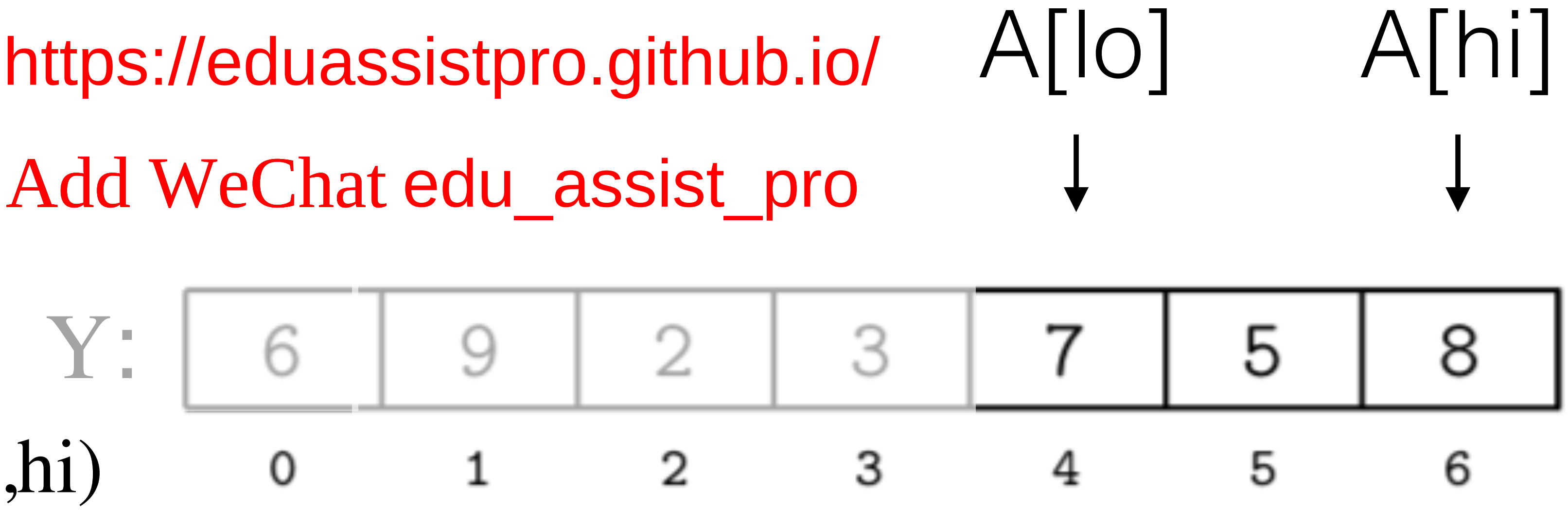
A: Y x: 7 lo: 4 hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: Array

- Solve the problem for a sub-instance and use the solution to solve the full instance

- For example, to locate item **x**.

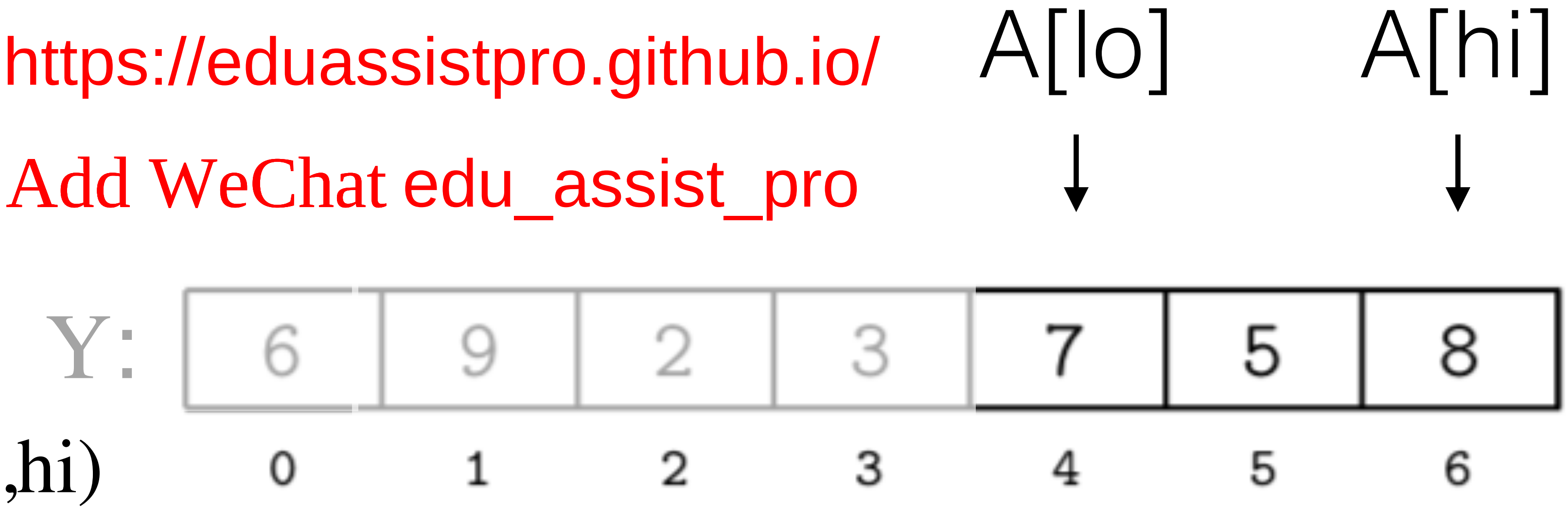
A: Y   x: 7   lo: 4   hi: 6

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



A[lo]   A[hi]

Y:	6	9	2	3	7	5	8
	0	1	2	3	4	5	6

Initial call: find(A,x,0,n-1)

Let's trace the execution of find(Y,7,0,6)

(returns 4)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

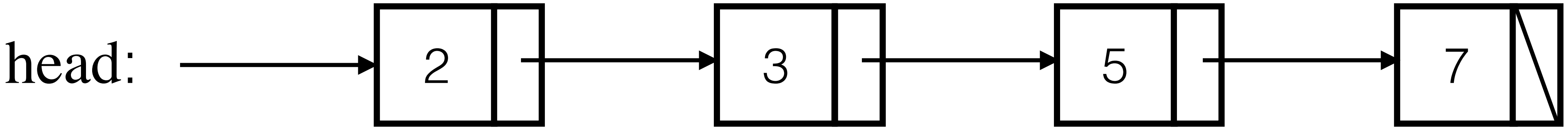
- Solve the problem for a sub-instance and use the solution to solve the full instance

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

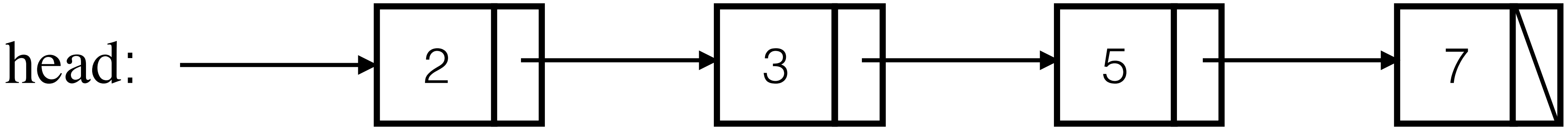
```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

(note similarity to array version)

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

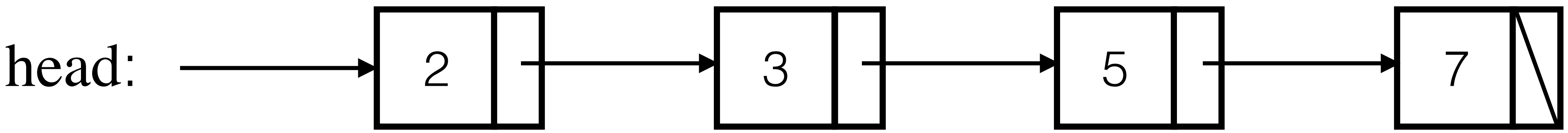
```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

(note similarity to array version)

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

p:

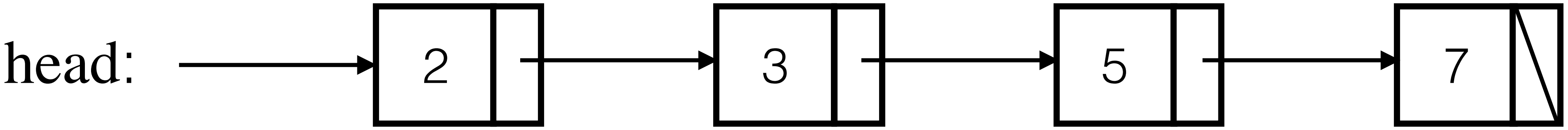
```
function find(A,x,lo,hi)
    if lo > hi
          turn -1
          if A[lo] = x
            rn lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

(note similarity to array version)

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

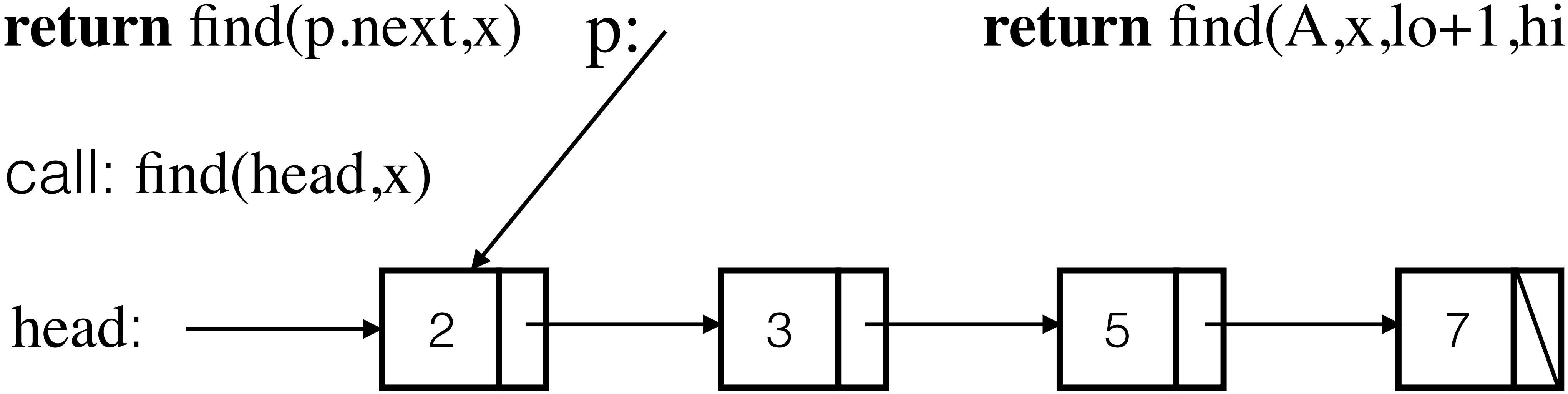
```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

(note similarity to array version)

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

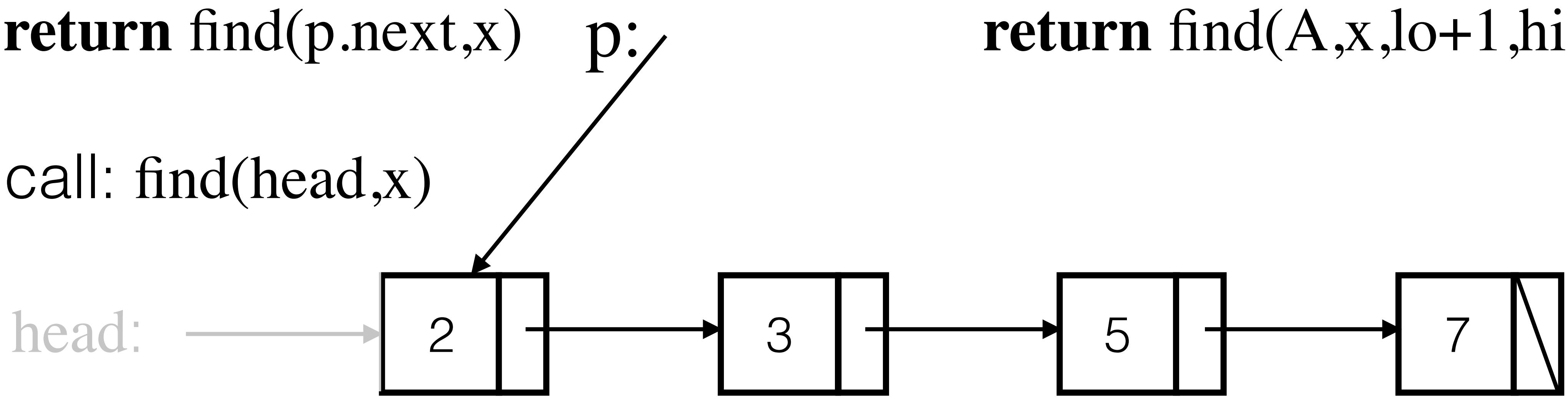
```
function find(A,x,lo,hi)
    if lo > hi
        turn -1
        if A[lo] = x
        rn lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

(note similarity to array version)

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

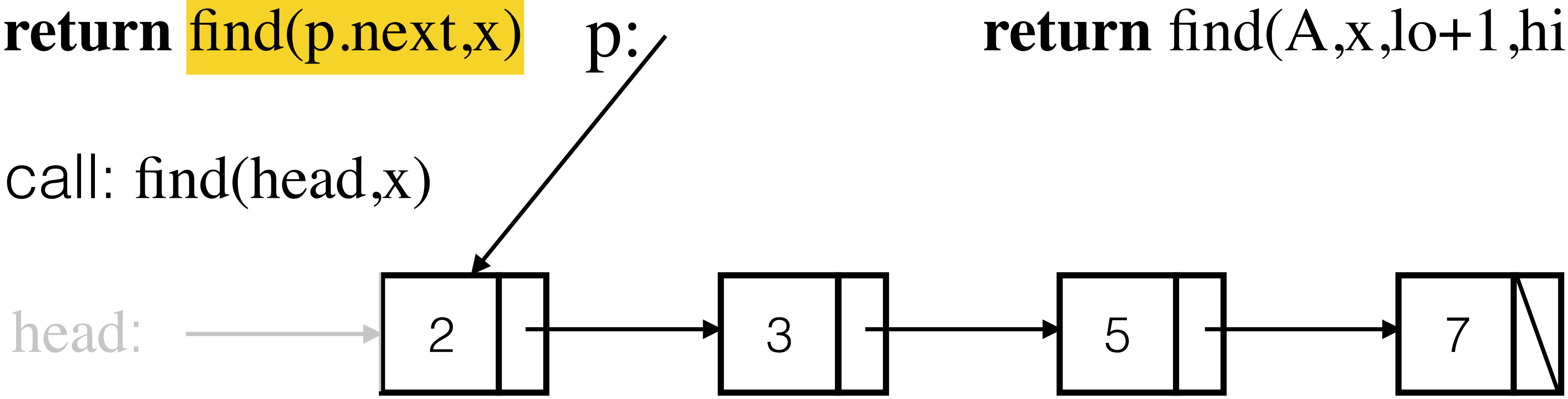
```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

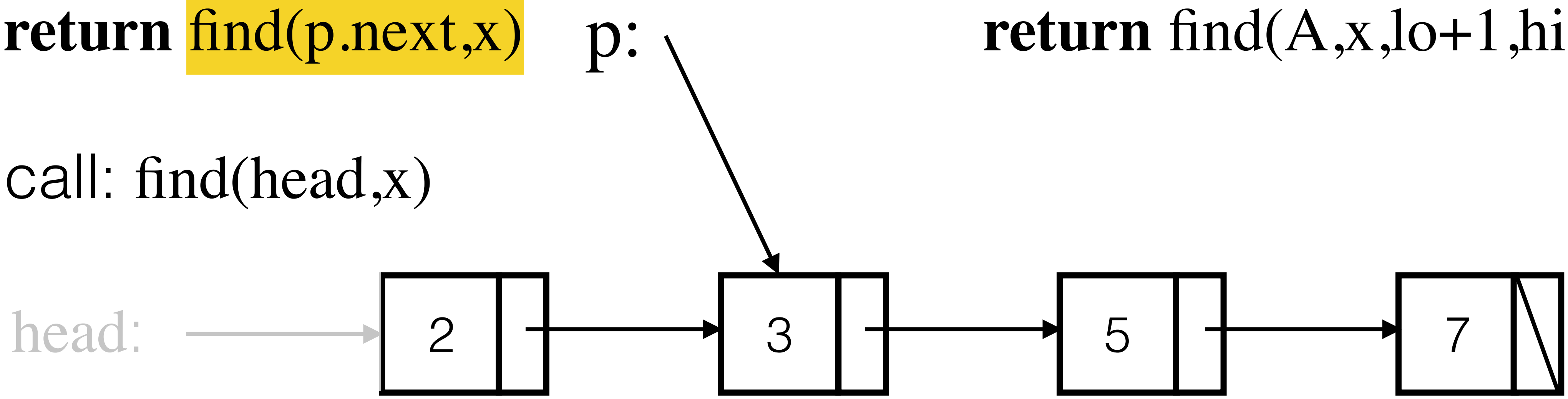
(note similarity to array version)

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

(note similarity to array version)

```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

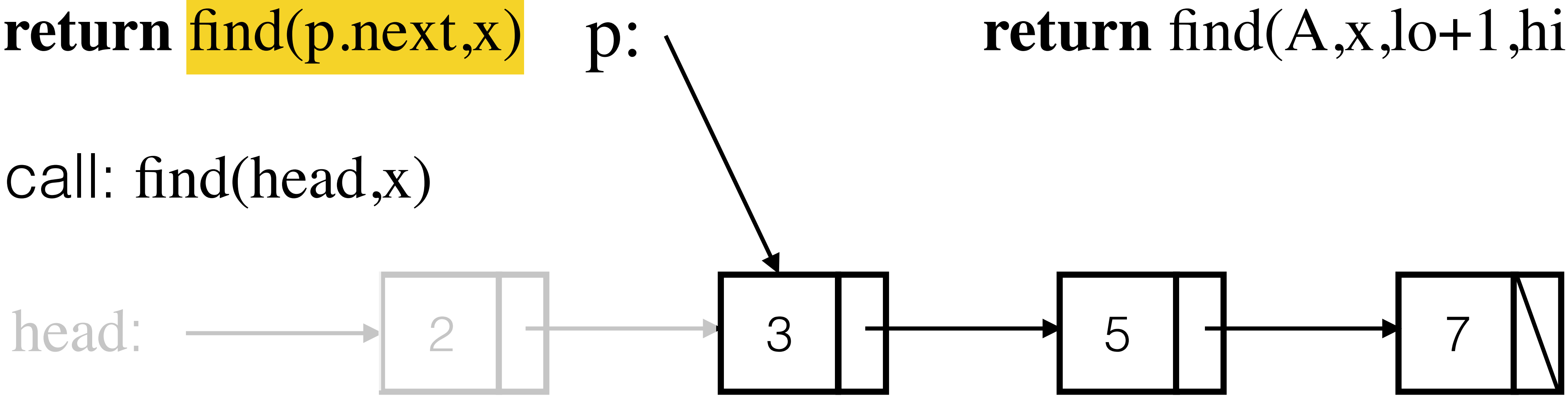
```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Recursive Processing: List

- Solve the problem for a sub-instance and use the solution to solve the full instance

(note similarity to array version)

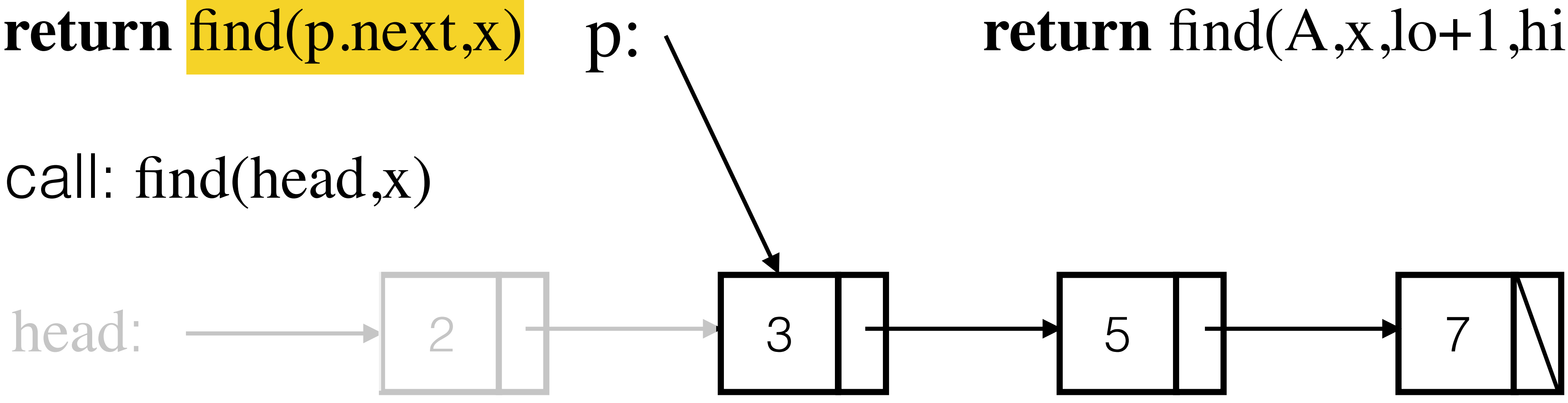
```
function find(p,x)
    if p = null
        return p
    else if p.val = x
        return p
    else
        return find(p.next,x)
```

we will discuss sion w

```
function find(A,x,lo,hi)
    if lo > hi
        return -1
    else if A[lo] = x
        return lo
    else
        return find(A,x,lo+1,hi)
```

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Initial call: find(head,x)



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Abstract DataTypes

THE UNIVERSITY OF MELBOURNE

- A collection of data items, and a family of operations that operate on that data

- Think of an ADT as a set of contracts, an **interface**

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- We must still **implement** t an advantage to separate the implementation of the ADT from the “ i.e. the interface it provides)

- Good programming practice is to support this separation
  - Nothing outside of the definition of the ADT should refer to anything inside, except through function calls and basic operations

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …
- Usually implemented as an ADT

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

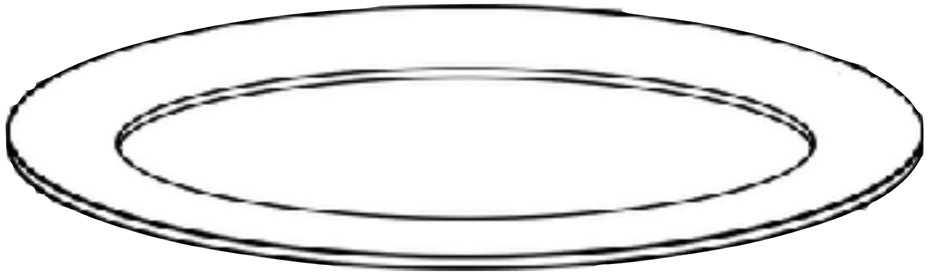
Fundamental Data Structure: The Stack

- Last-In-First-Out (LIFO)
- Operations:
    - CreateStack
    - Push
    - Pop
    - Top
    - EmptyStack?
    - …
- Usually implemented as an ADT

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

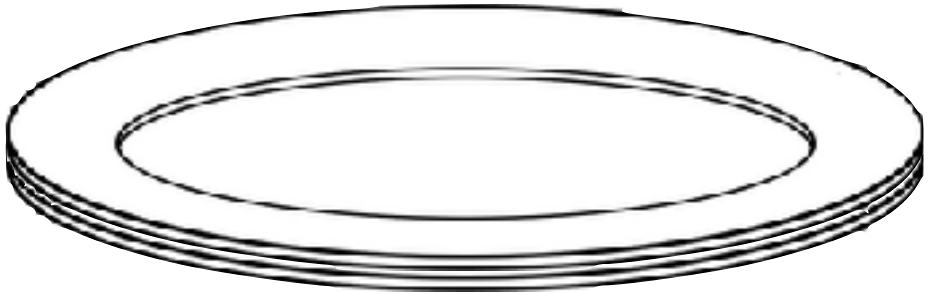
Fundamental Data Structure: The Stack

- Last-In-First-Out (LIFO)
- Operations:
    - CreateStack
    - Push
    - Pop
    - Top
    - EmptyStack?
    - …
- Usually implemented as an ADT

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro



Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- Usually implemented as an ADT

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- Usually implemented as an ADT

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- Usually implemented as an ADT

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- Usually implemented as an ADT

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- Usually implemented as an ADT

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: The Stack

- Last-In-First-Out (LIFO)
- Operations:
  - CreateStack
  - Push
  - Pop
  - Top
  - EmptyStack?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- Usually implemented as an ADT

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Array

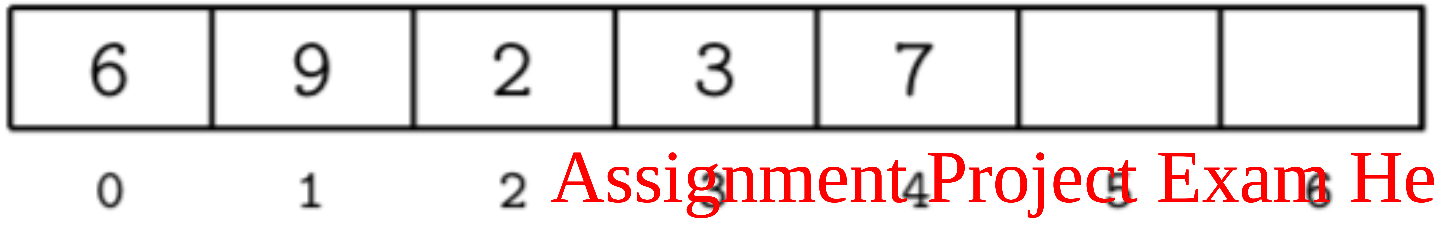
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Array



6	9	2	3	7		
0	1	2	3	4	5	6

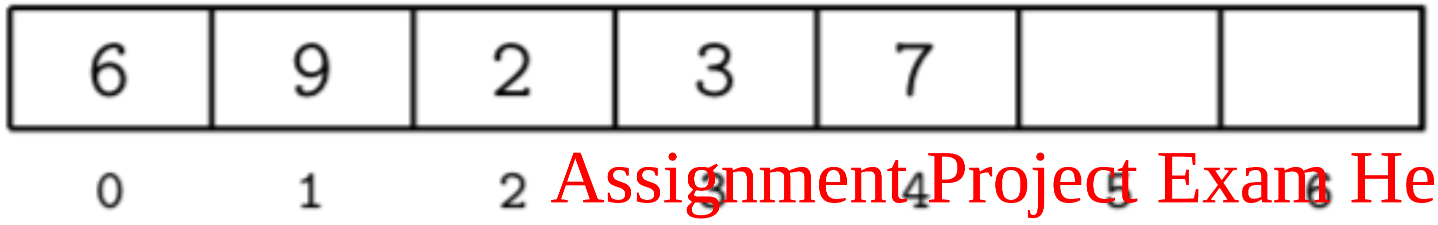
Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Array



6	9	2	3	7		
0	1	2	3	4	5	6

top: 5

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Array

6	9	2	3	7		
0	1	2	3	4	5	6

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

top: 5

Push(5)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Array

6	9	2	3	7	5	
0	1	2	3	4	5	6

Assignment Project Exam Help

top: 5

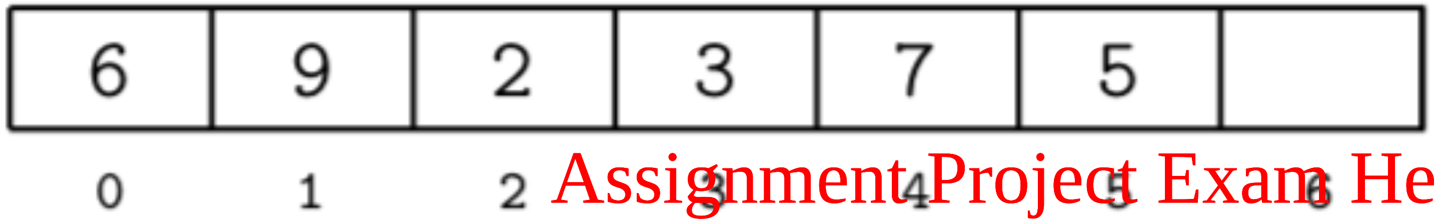
https://eduassistpro.github.io/

Push(5)

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Array



6	9	2	3	7	5	
0	1	2	3	4	5	6

Assignment Project Exam Help

https://eduassistpro.github.io/

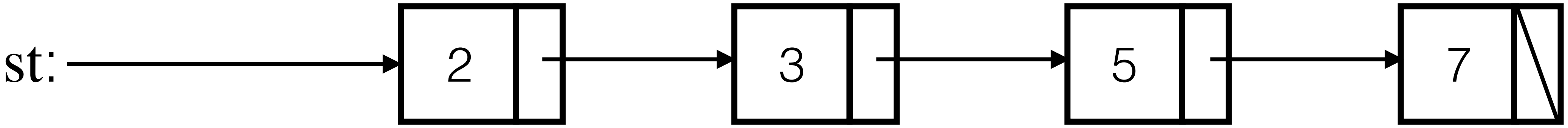
Add WeChat edu_assist_pro

top: 6

Push(5)

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



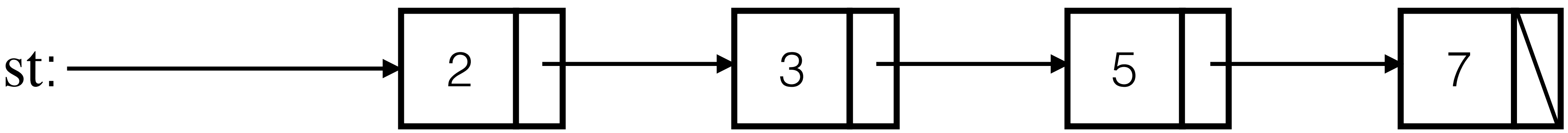
Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

```
function push(st,x)
            ew     node
                  x
                   st

    st ← elt
    return st
```

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



Push(5)

Assignment Project Exam Help

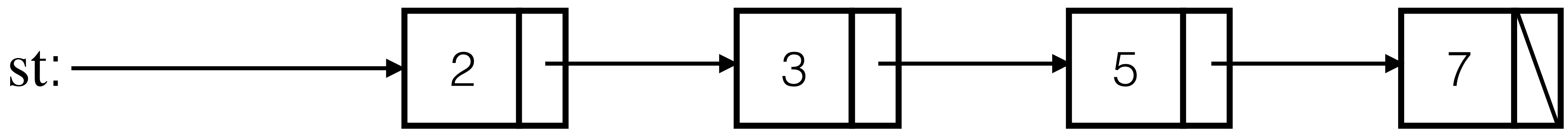
https://eduassistpro.github.io/

Add WeChat edu_assist_pro

```
function push(st,x)
    ew    node
         x
          st
st ← elt
return st
```

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



Push(5)

Assignment Project Exam Help

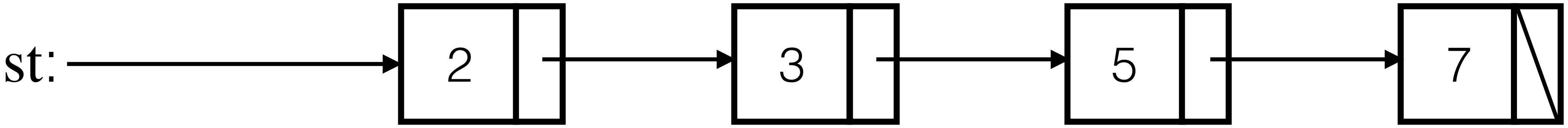
https://eduassistpro.github.io/

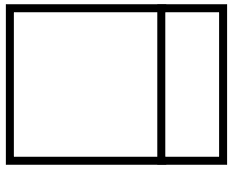
Add WeChat edu_assist_pro

```
function push(st,x)
    **ew** node
            x
            st
    st ← elt
    return st
```

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List





Push(5)

Assignment Project Exam Help

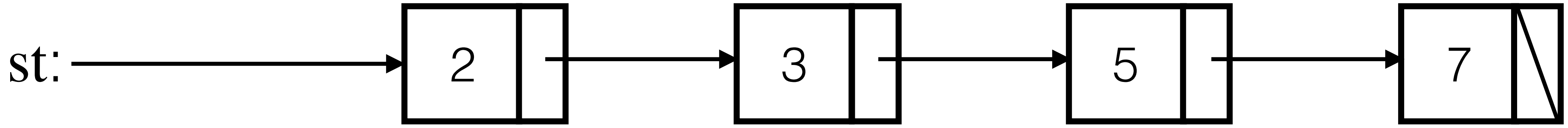
https://eduassistpro.github.io/

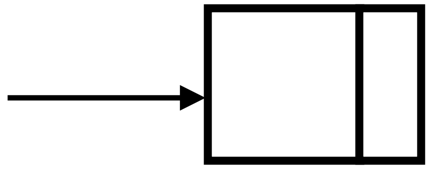
Add WeChat edu_assist_pro

```
function push(st,x)
    ew    node
              x
              st
    st ← elt
    return st
```

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List





Push(5)

Assignment Project Exam Help

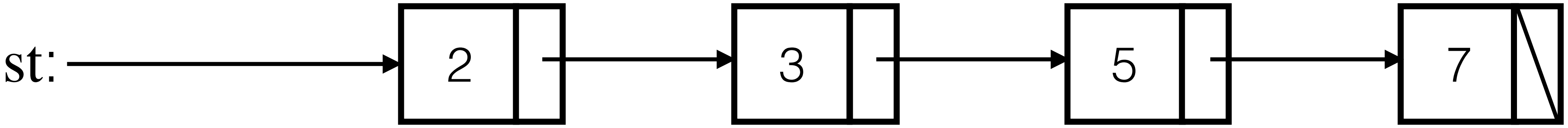
https://eduassistpro.github.io/

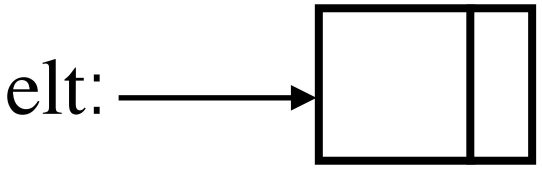
Add WeChat edu_assist_pro

```
function push(st,x)
            ew    node
                 x
                  st
   st ← elt
   return st
```

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List





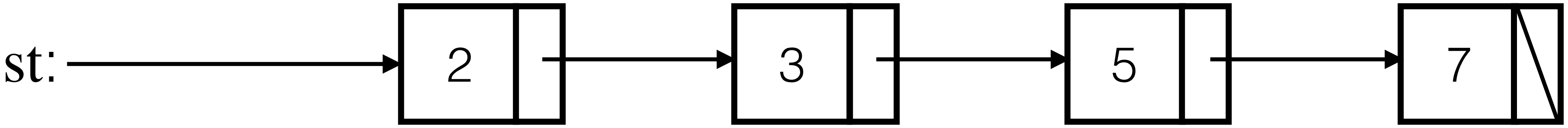
Push(5)

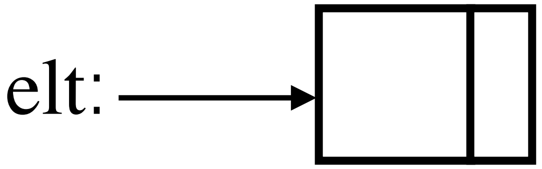
```
function push(st,x)
    ew    node
                x
                st
    st ← elt
    return st
```

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List





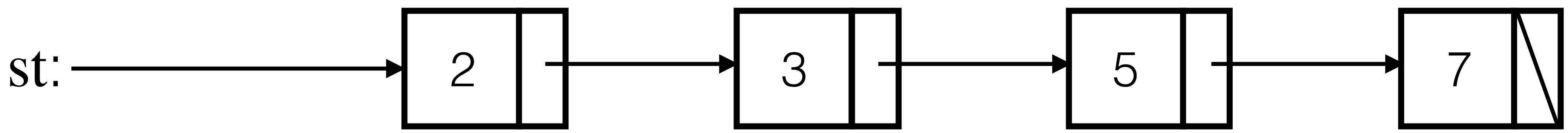
Push(5)

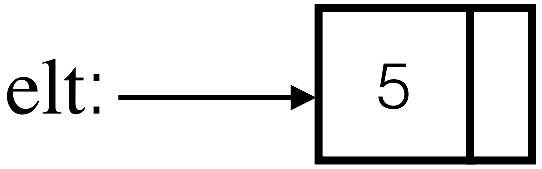
Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

```
function push(st,x)
    ew node
    x
    st
    st ← elt
    return st
```

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List





Push(5)

Assignment Project Exam Help

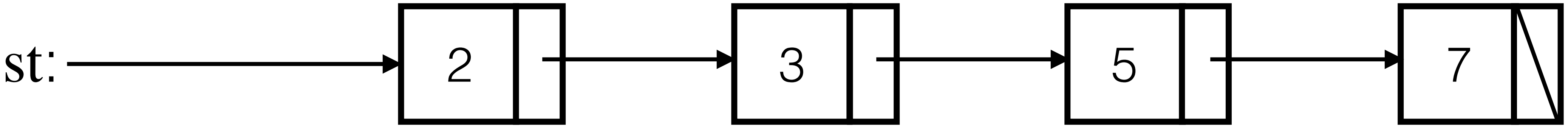
https://eduassistpro.github.io/

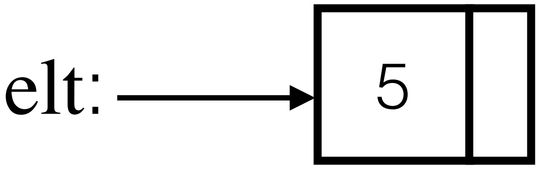
Add WeChat edu_assist_pro

```
function push(st,x)
    ew      node
                x
                st
    st ← elt
    return st
```

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List





Push(5)

Assignment Project Exam Help

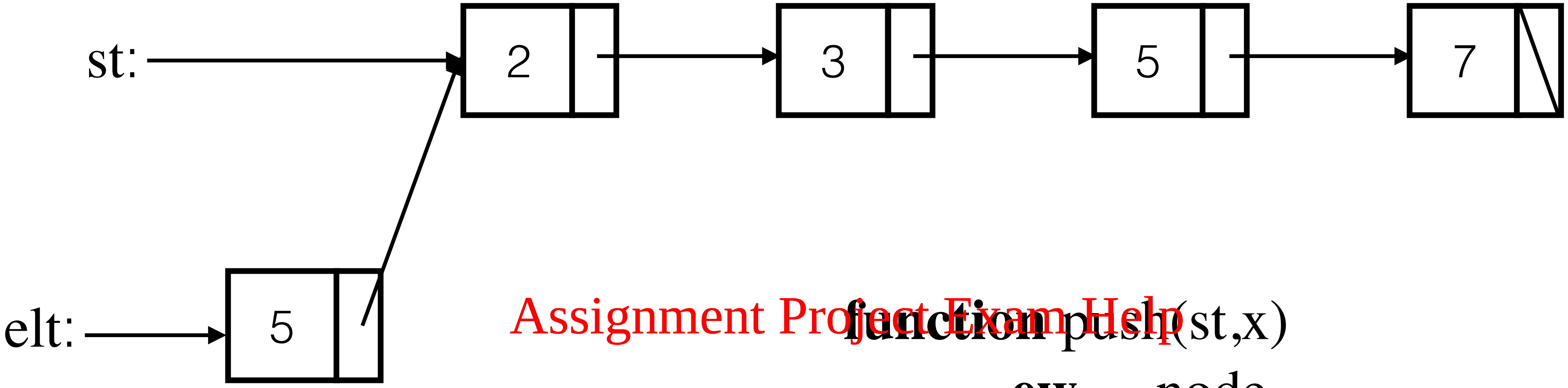
https://eduassistpro.github.io/

Add WeChat edu_assist_pro

```
function push(st,x)
    ew     node
            x
             st
    st ← elt
    return st
```

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



Push(5)

Assignment Project Exam Help

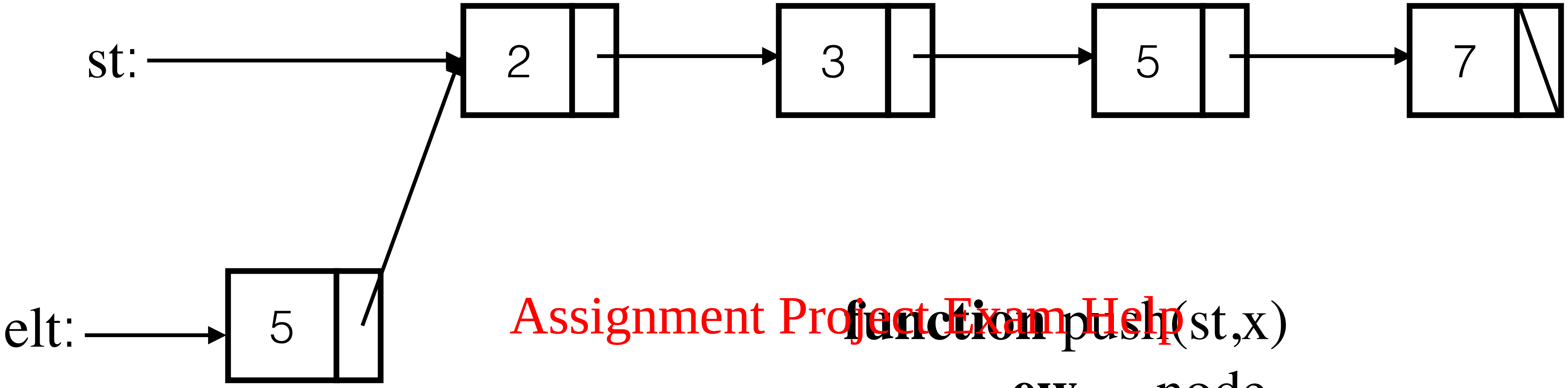
https://eduassistpro.github.io/

Add WeChat edu_assist_pro

```
function push(st,x)
    ew     node
           x
            st
    st ← elt
    return st
```

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



Push(5)

Assignment Project Exam Help

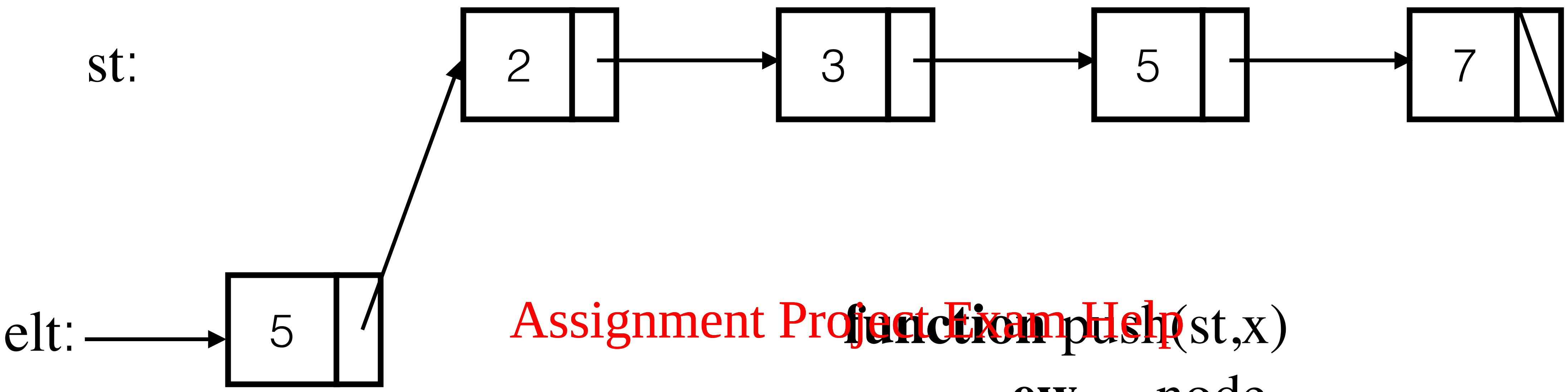
https://eduassistpro.github.io/

Add WeChat edu_assist_pro

```
function push(st,x)
    ew   node
              x
               st
    st ← elt
    return st
```

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



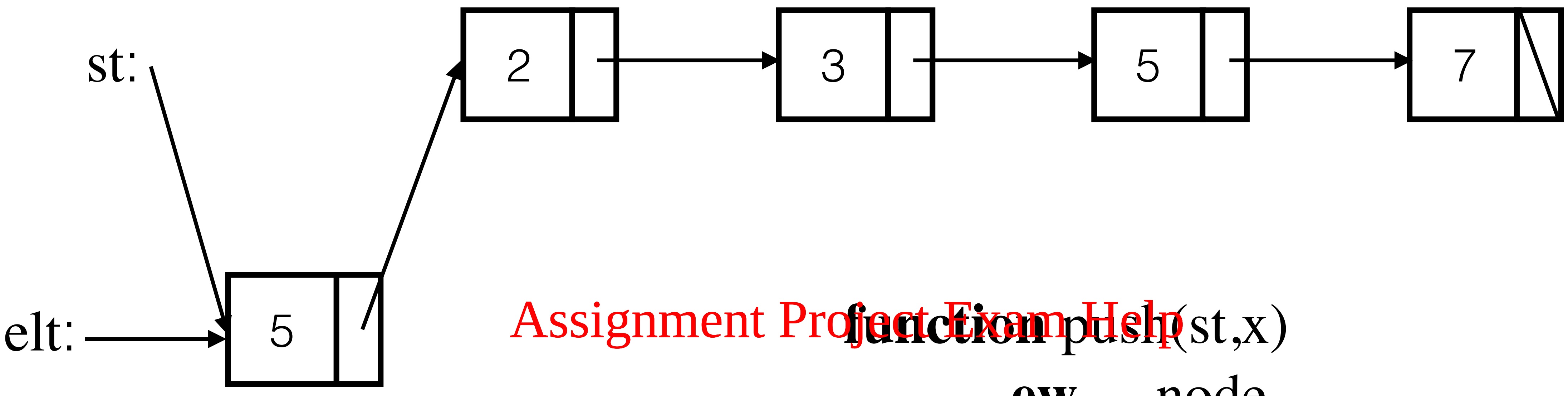
Push(5)

```
function push(st,x)
    ew      node
            x
             st
    st ← elt
    return st
```

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



Push(5)

Assignment Project Exam Help

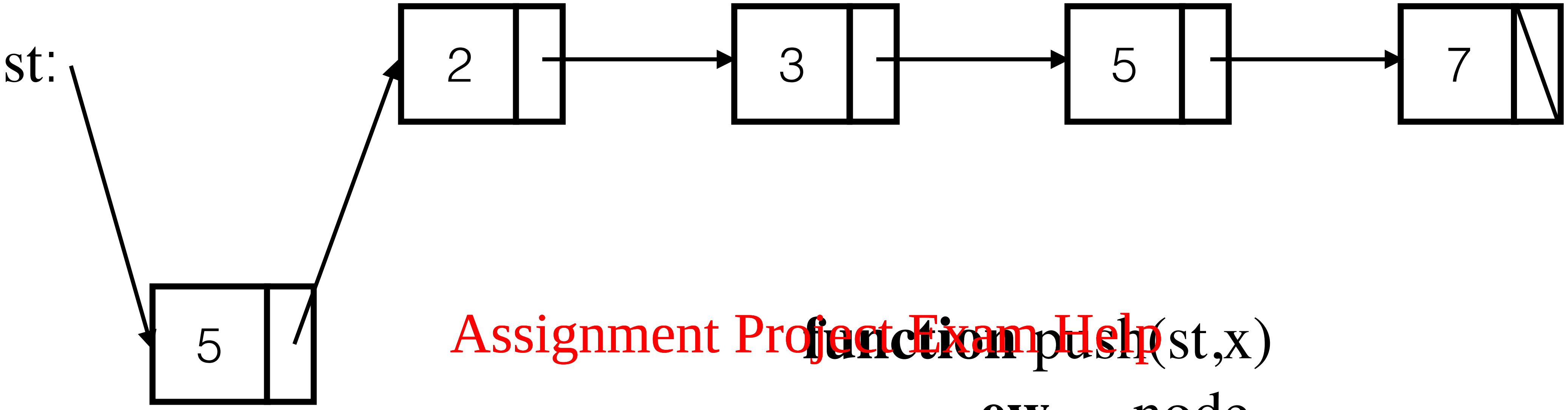
https://eduassistpro.github.io/

Add WeChat edu_assist_pro

```
function push(st,x)
    ew node
    x
    st
    st ← elt
    return st
```

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



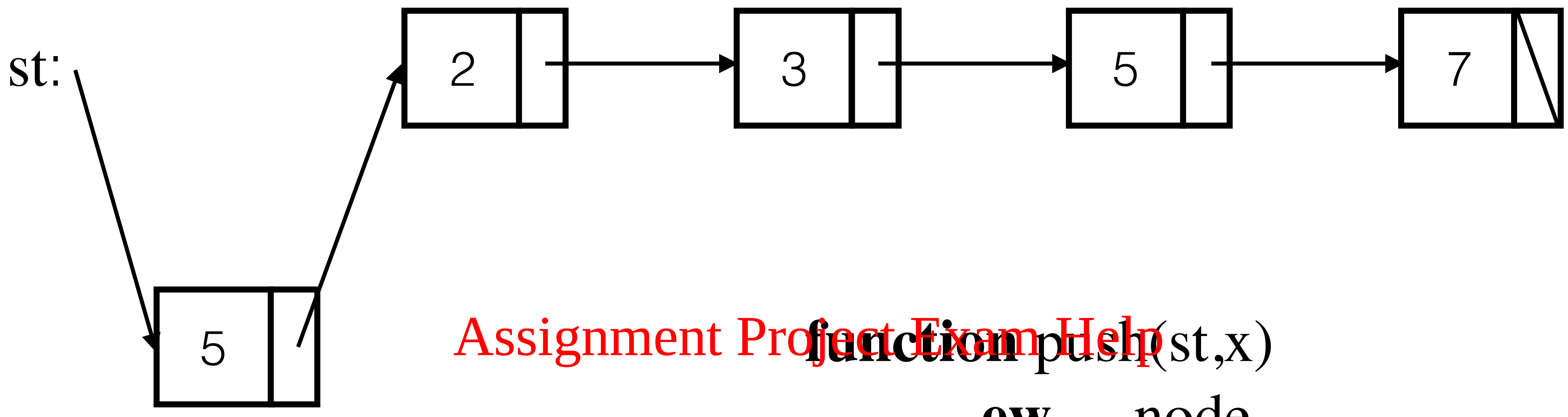
Push(5)

```
function push(st,x)
    ew      node
             x
              st
    st ← elt
    return st
```

Assignment Project Exam Help
https://eduassistpro.github.io/
Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Stack Implementation: Linked List



Push(5)

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

```
function push(st,x)
    ew    node
          x
           st
    st ← elt
    return st
```

See
https://www.cs.usfca.edu/~galles/visualization/Algorithms.html
for more visualisations

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Pseudo Code

- On the previous slide, we assumed that a “node” has two attributes: a “val” which is its value, and a “next” which points to the rest of the list.

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

- There is no standard for pseudo-code. mples in Levitin as a guide. Cormen et al. pages 20–22 (in Re urces) has a list of standard conventions used with pseudo-code which are good to follow, except we use ← as the assignment operator.

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Fundamental Data Structure: Queues

- First-In-First-Out (FIFO)
- Operations:
  - CreateQueue
  - Enqueue
  - Dequeue
  - Head
  - EmptyQueue?
  - …

Assignment Project Exam Help

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Other Data Structures

- We will meet many other (abstract) data structures, e.g.
  - The priority queue Assignment Project Exam Help
  - Various types of “tree” https://eduassistpro.github.io/
  - Various types of “graphAdd WeChat edu_assist_pro
- If you check out algorithm animation tools or advanced algorithm books, you will meet exotic data structures such as splay trees and skip lists.

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License

Next Week

Assignment Project Exam Help

- Algorithm analysis—how t rithm's resource consumption.

https://eduassistpro.github.io/

Add WeChat edu_assist_pro

Copyright University of Melbourne 2016, provided under Creative Commons Attribution License