

# COMP 250

INTRODUCTORY SCIENCE

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Week 12-2:  
Add WeChat: edu\_assist\_pro

Giulia Alberini, Fall 2020

Slides adapted from Michael Langer's

# WHAT ARE WE GOING TO DO IN THIS VIDEO?



- **Heaps**

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

# PRIORITY QUEUE

Assume a set of comparable elements or “keys”.

Assignment Project Exam Help

Like a queue, but now <https://eduassistpro.github.io/> definition of which  
element to remove next, namely the best priority.

Add WeChat edu\_assist\_pro

e.g. hospital emergency room

# PRIORITY QUEUE ADT

- `add(key)`

- `removeMin()`

“highest” priority =

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

- `peek()`

- `contains(element)`

- `remove(element)`

# HOW TO IMPLEMENT A PRIORITY QUEUE ?

- sorted list ?

Assignment Project Exam Help

- binary search tree (la

<https://eduassistpro.github.io/>

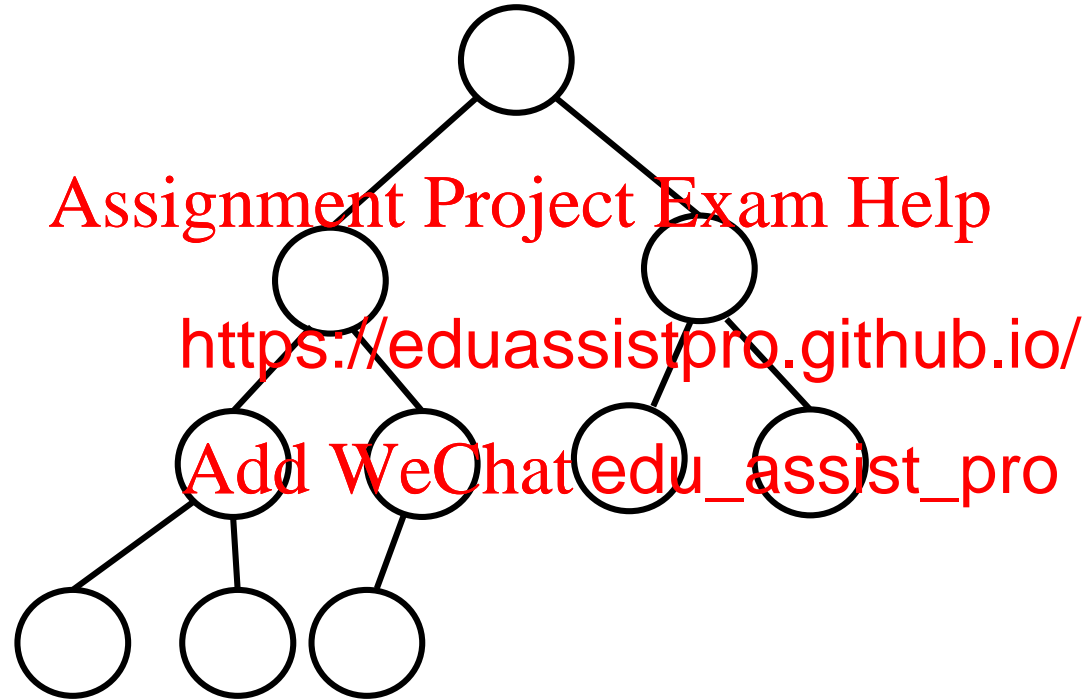
- balanced binary search tree (COMP

Add WeChat edu\_assist\_pro

- heap (next 2 lectures)

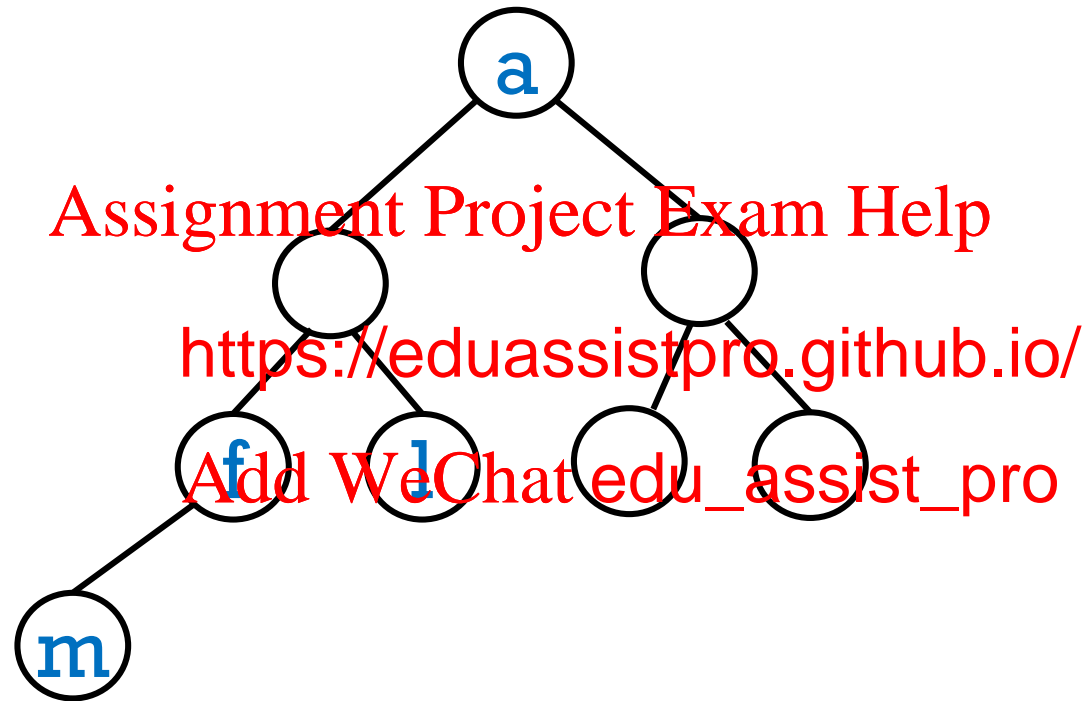
Not the same “heap” you hear about in COMP 206.

## COMPLETE BINARY TREE (DEFINITION)



Binary tree of height  $h$  such that every level less than  $h$  is full, and all nodes at level  $h$  are as far to the left as possible

## MIN HEAP (DEFINITION)

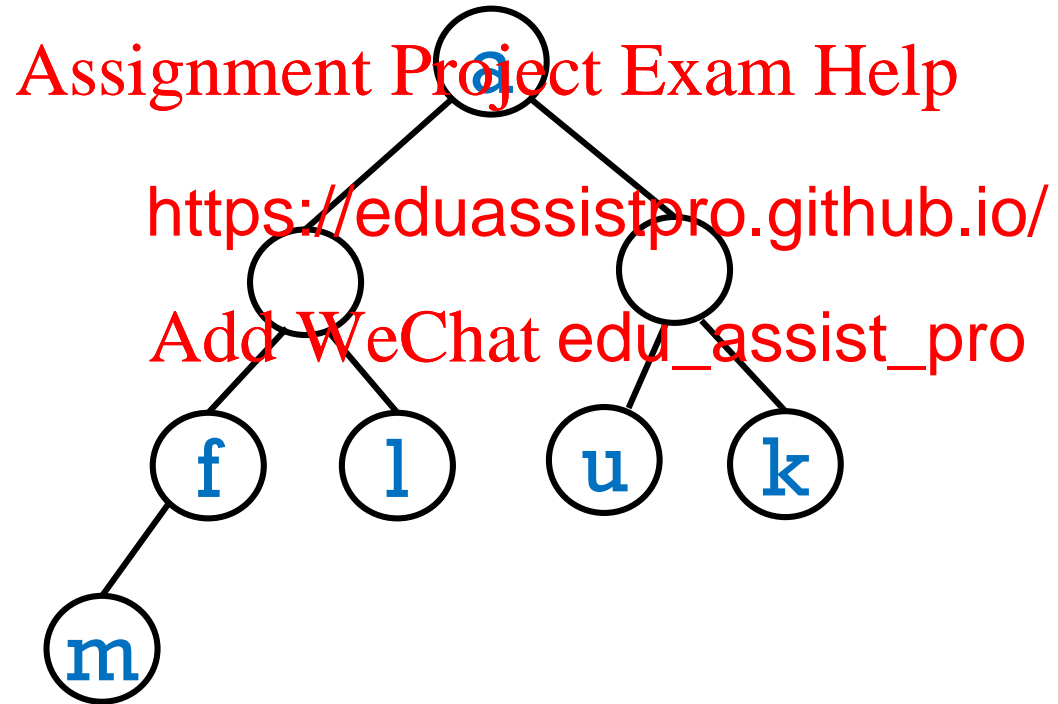


Complete binary tree with unique comparable elements, such that each node's element (key) is less than its children's element (key).



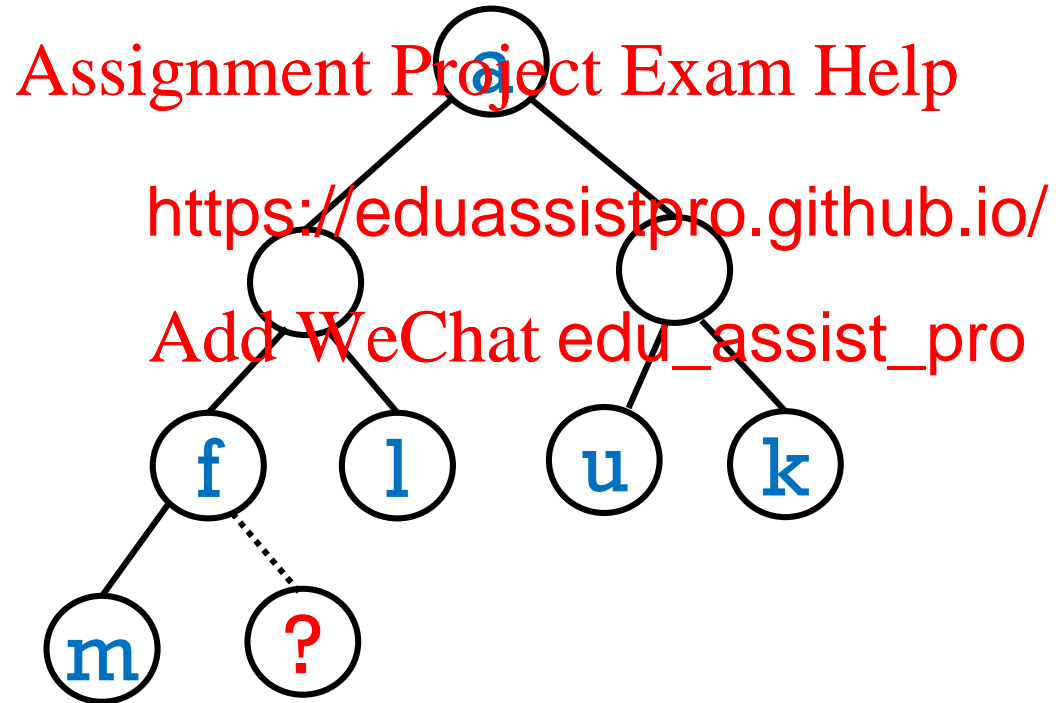
ADD()

For example, add(c)



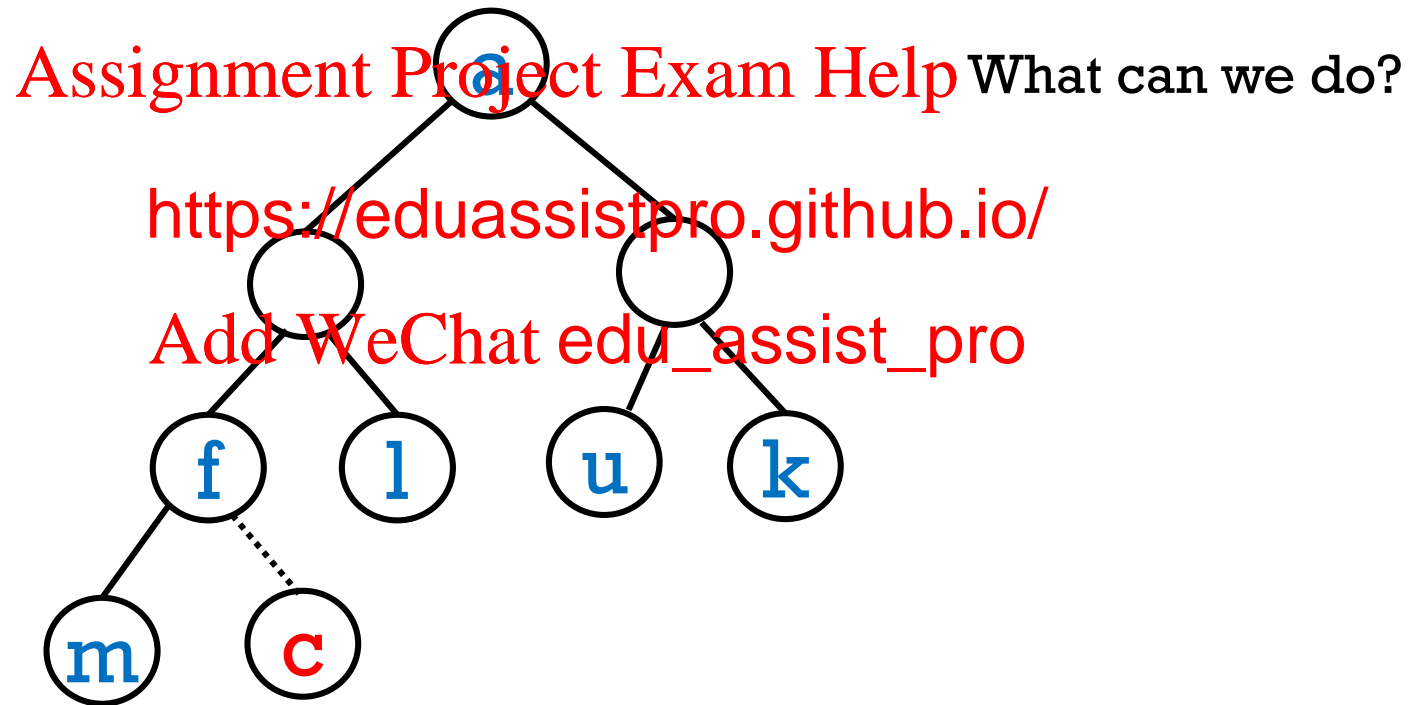
ADD()

For example, add(**c**)



ADD()

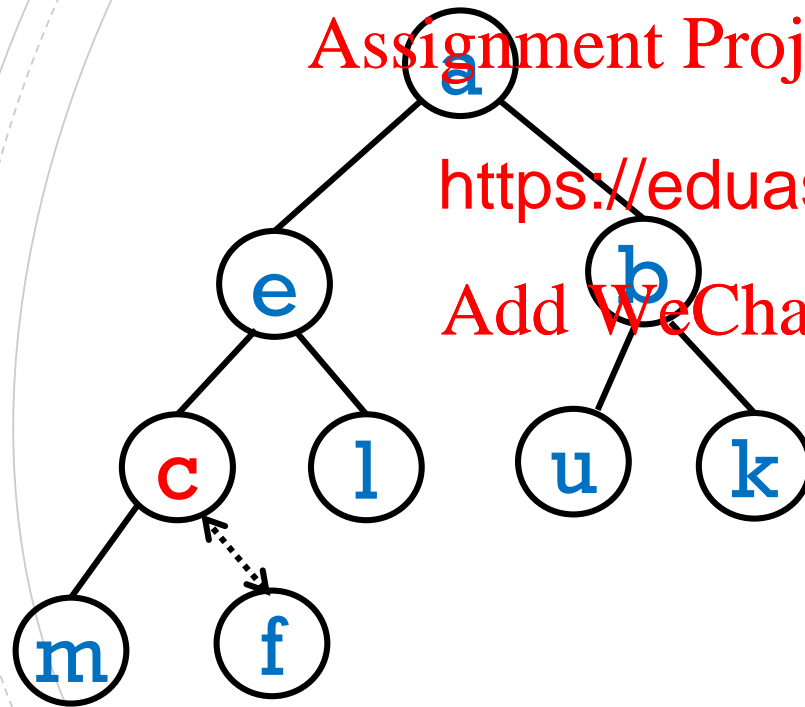
For example, `add(c)`



Problem : adding at the next available slot typically destroys the heap property.

ADD()

For example, add(**c**)



Assignment Project Exam Help

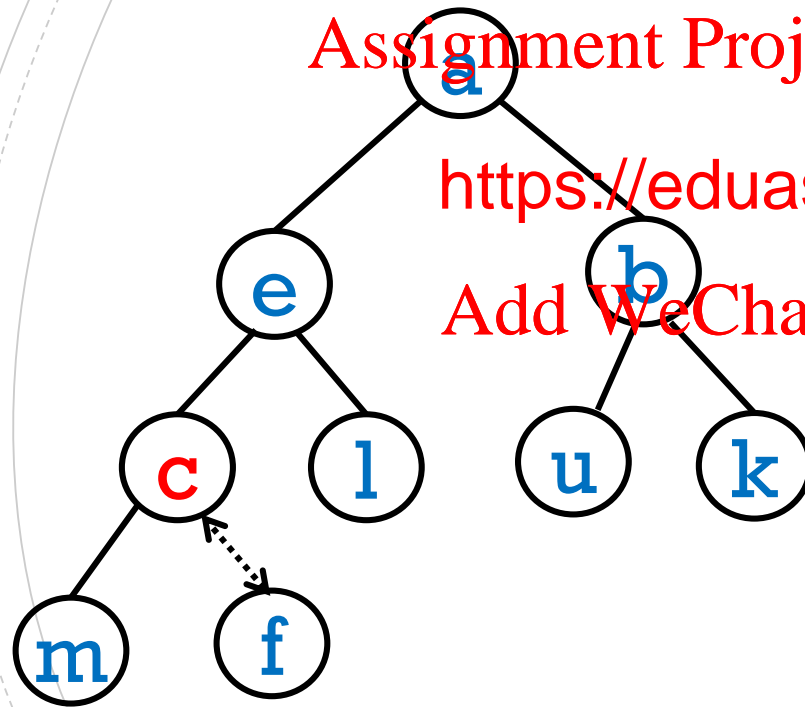
What can we do?

<https://eduassistpro.github.io/> What can we do with its parent f.

Add WeChat edu\_assist\_pro is create a problem with **c**'s former sibling, who is now **c**'s child?

ADD()

For example, add(**c**)



Assignment Project Exam Help

What can we do?

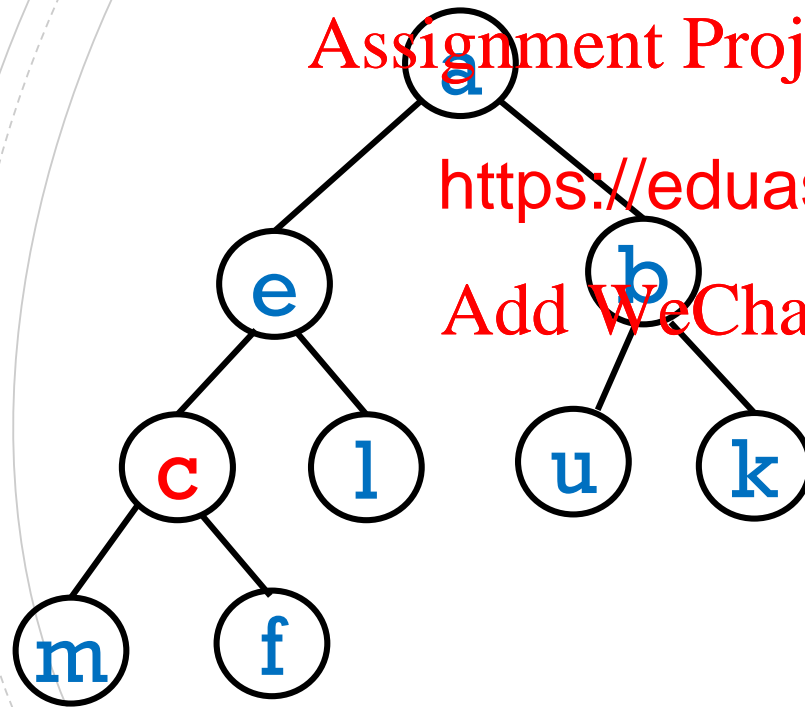
<https://eduassistpro.github.io/> What can we do with its parent f.

Add WeChat edu\_assist\_pro is create a problem with c's former sibling, who is now c's child?

A: No. Why?

ADD()

For example, add(**c**)



Assignment Project Exam Help

Q: Are we done?

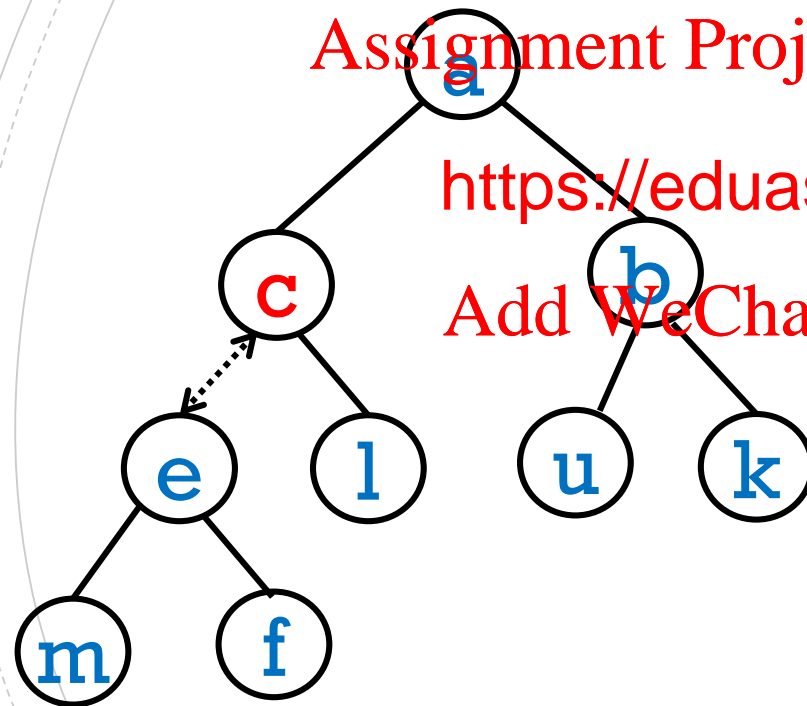
<https://eduassistpro.github.io/>

unnecessarily. What about **c**'s

Add WeChat edu\_assist\_pro

ADD()

For example, add(**c**)

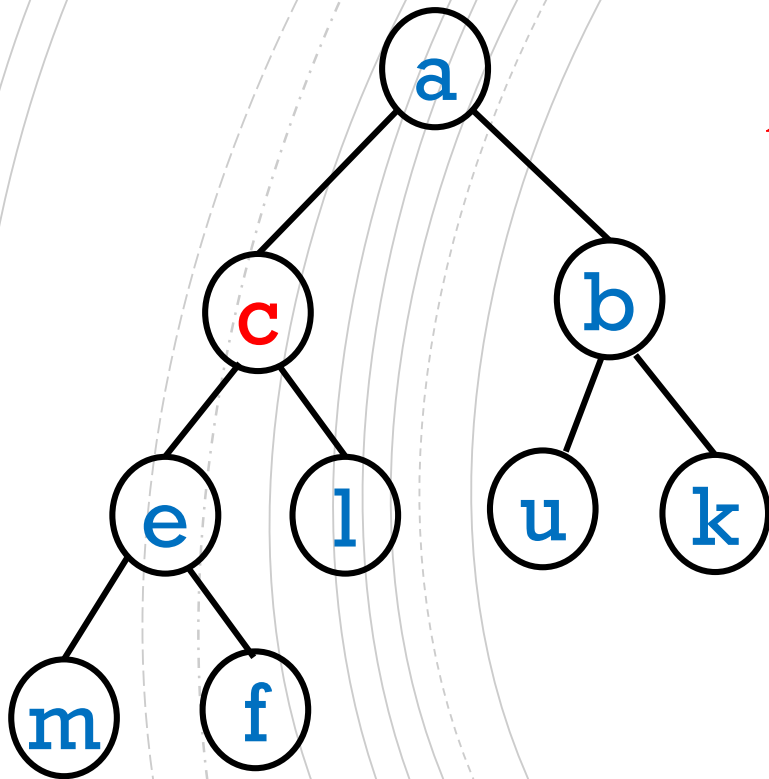


We swap **c** with its (new) parent **e**.

are done because **c** is greater than its parent **a**

Add WeChat edu\_assist\_pro

## ADD() - IMPLEMENTATION



```
add(key) {  
    cur = new node at next available leaf position  
    cur.key = key
```

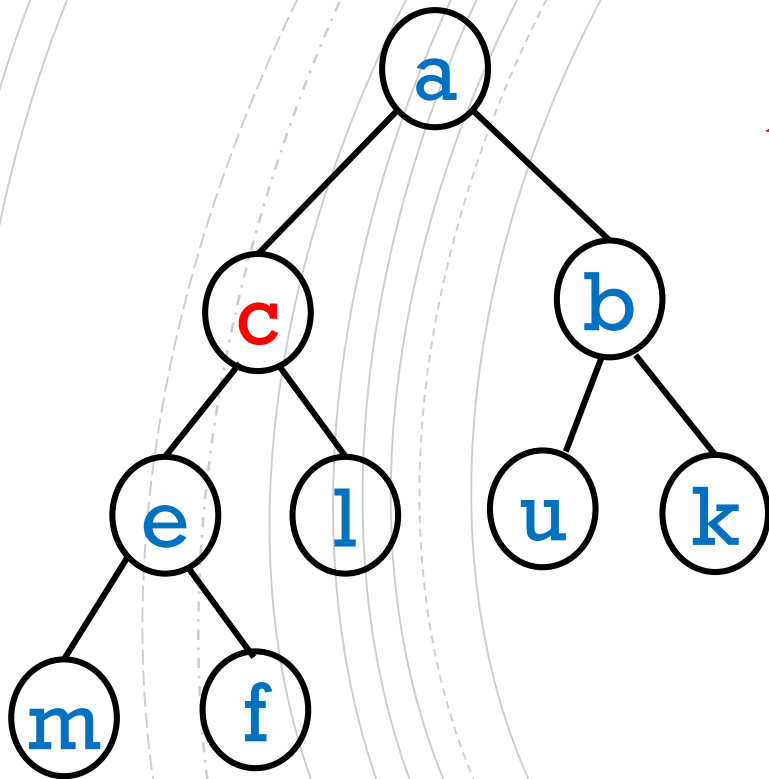
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

```
}
```



## ADD() - IMPLEMENTATION



```
add(key) {
```

Assignment Project Exam Help

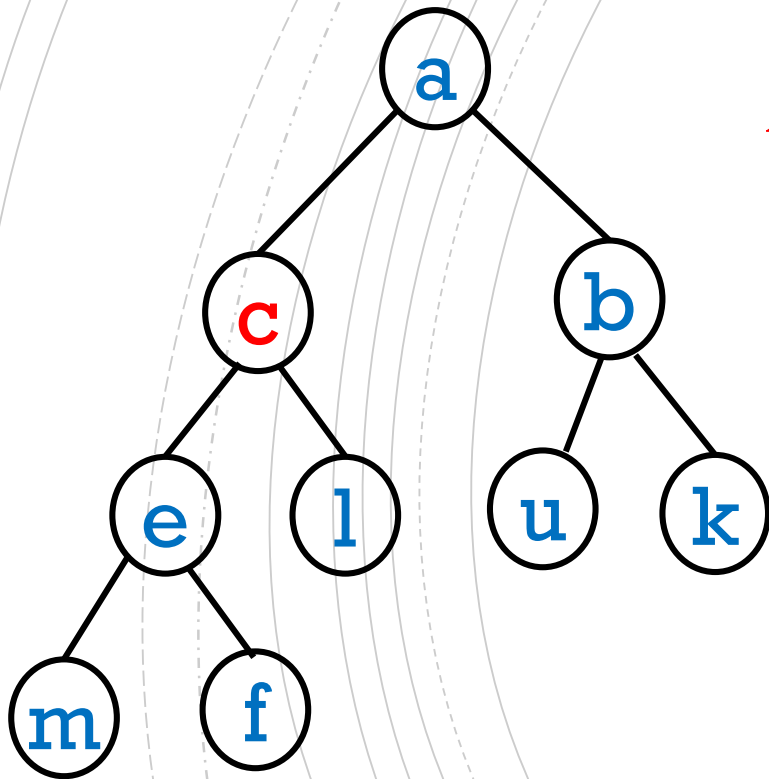
```
    cur = new node at next available leaf position  
    cur.key = key
```

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

```
}
```

## ADD() - IMPLEMENTATION



```
add(key) {
```

```
    cur = new node at next available leaf position
```

```
    cur.key = key
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

```
    else {
```

Add WeChat edu\_assist\_pro

```
        while (c & cur.key < cur.parent.key) {
```

```
            swapKeys(cur, cur.parent)
```

```
            cur = cur.parent
```

```
        }
```

```
    }
```

```
}
```

## HOW TO BUILD A HEAP?

add( **k** ) Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

## HOW TO BUILD A HEAP?

add( **k** )  
add( **f** )

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

## HOW TO BUILD A HEAP?

add( **k** )  
add( **f** )

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat  edu\_assist\_pro

## HOW TO BUILD A HEAP?

add( **k** )

add( **f** )

add( **e** )

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat  edu\_assist\_pro

## HOW TO BUILD A HEAP?

add( **k** )

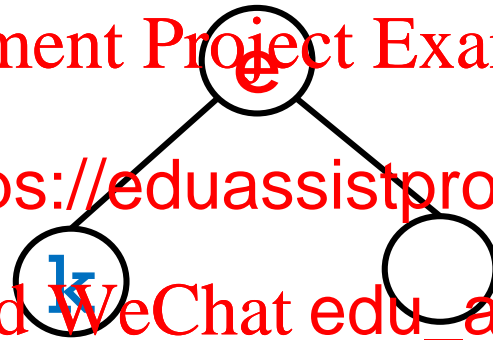
add( **f** )

add( **e** )

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro



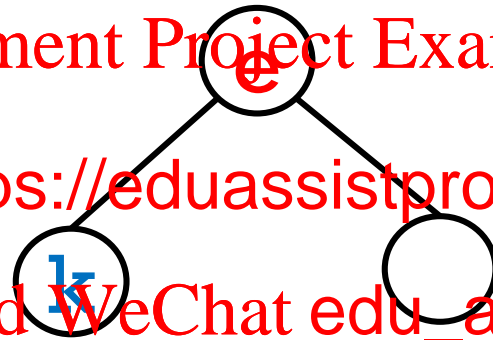
## HOW TO BUILD A HEAP?

add( **k** )  
add( **f** )  
add( **e** )  
add( **a** )

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro





## HOW TO BUILD A HEAP?

add( **k** )

add( **f** )

add( **e** )

add( **a** )

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

**k**



## HOW TO BUILD A HEAP?

add( **k** )  
add( **f** )  
add( **e** )  
add( **a** )  
add( **g** )

Assignment Project Exam Help

<https://eduassistpro.github.io/>

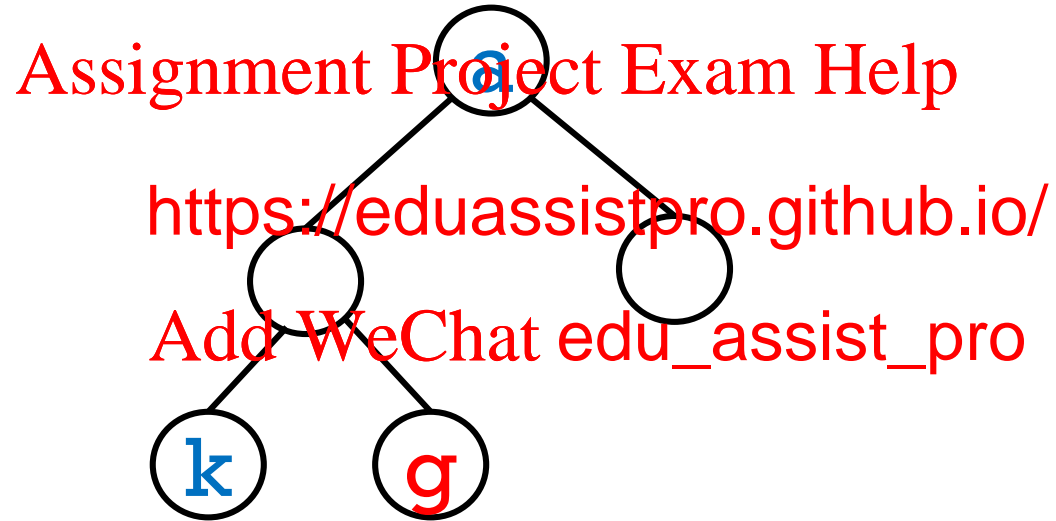
Add WeChat edu\_assist\_pro

**k**



## HOW TO BUILD A HEAP?

add( **k** )  
add( **f** )  
add( **e** )  
add( **a** )  
add( **g** )

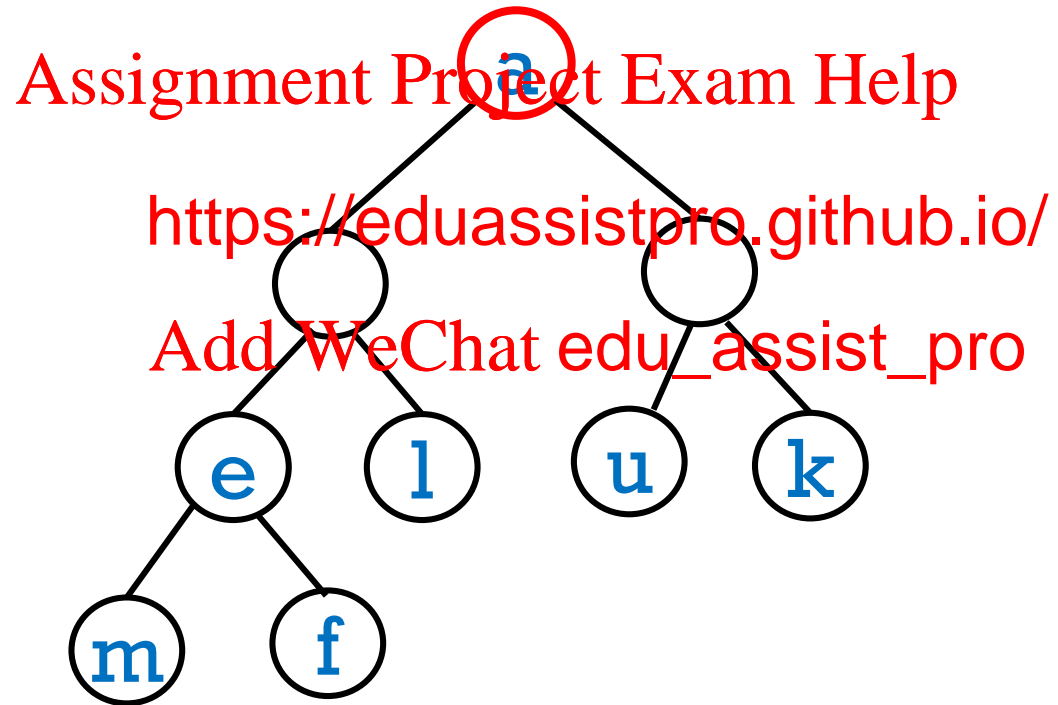


This method of building a heap is slow.

We will see a faster method next video.

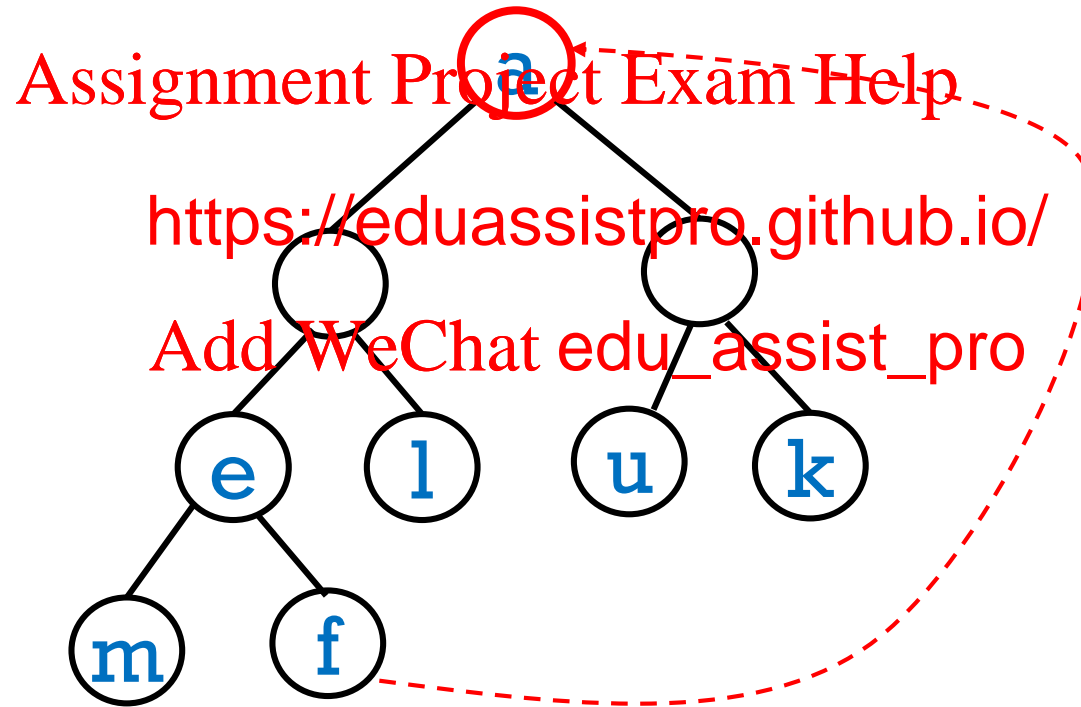
## REMOVEMIN()

returns root element



## REMOVEMIN()

returns root element



## REMOVEMIN()

a

Claim: if the root has two children, then the new root *will* be greater than at least one of its children.

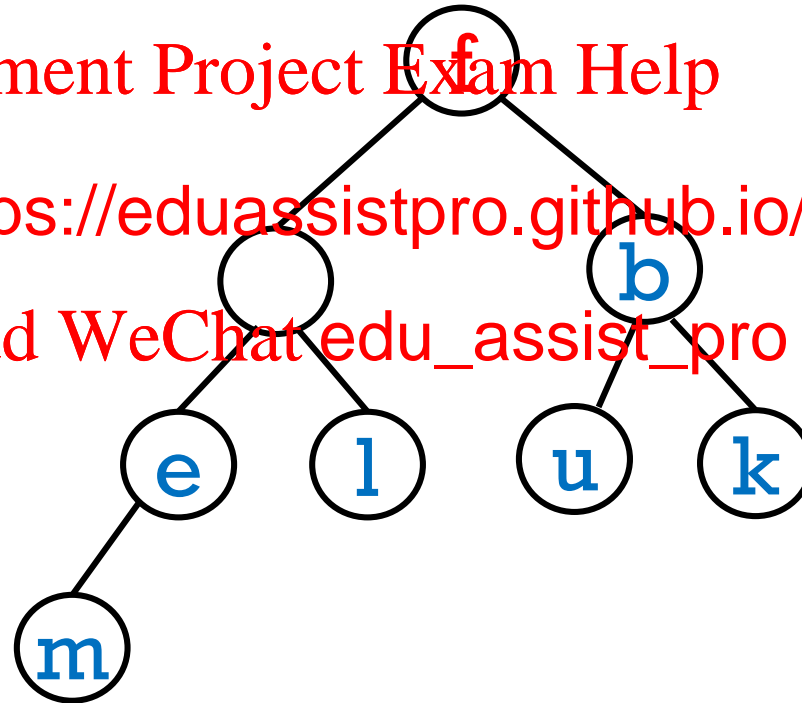
Why?

How to solve this problem?

Assignment Project Exam Help

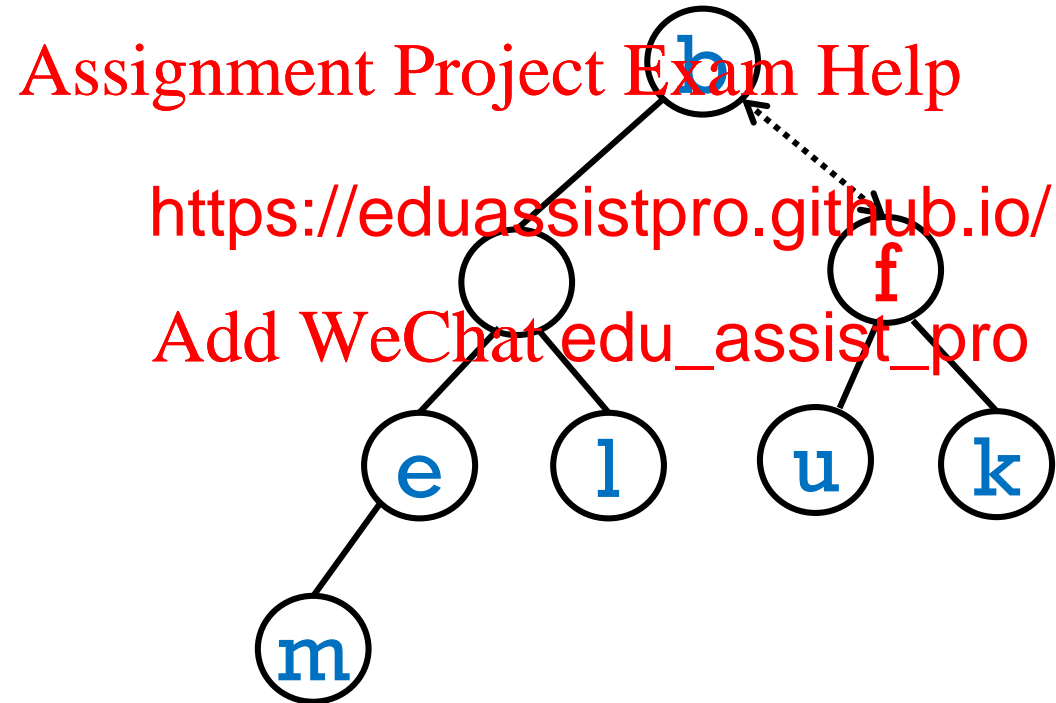
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro



## REMOVEMIN()

Swap keys with the smaller child!



## REMOVEMIN()

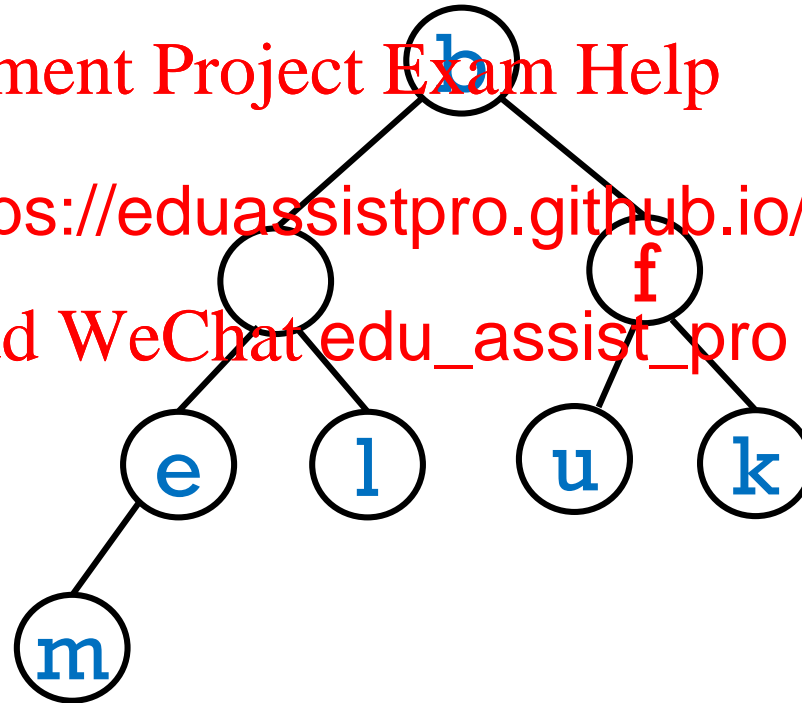
Swap keys with the smaller child!

Keep swapping with keys with the smaller child until it's necessary.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

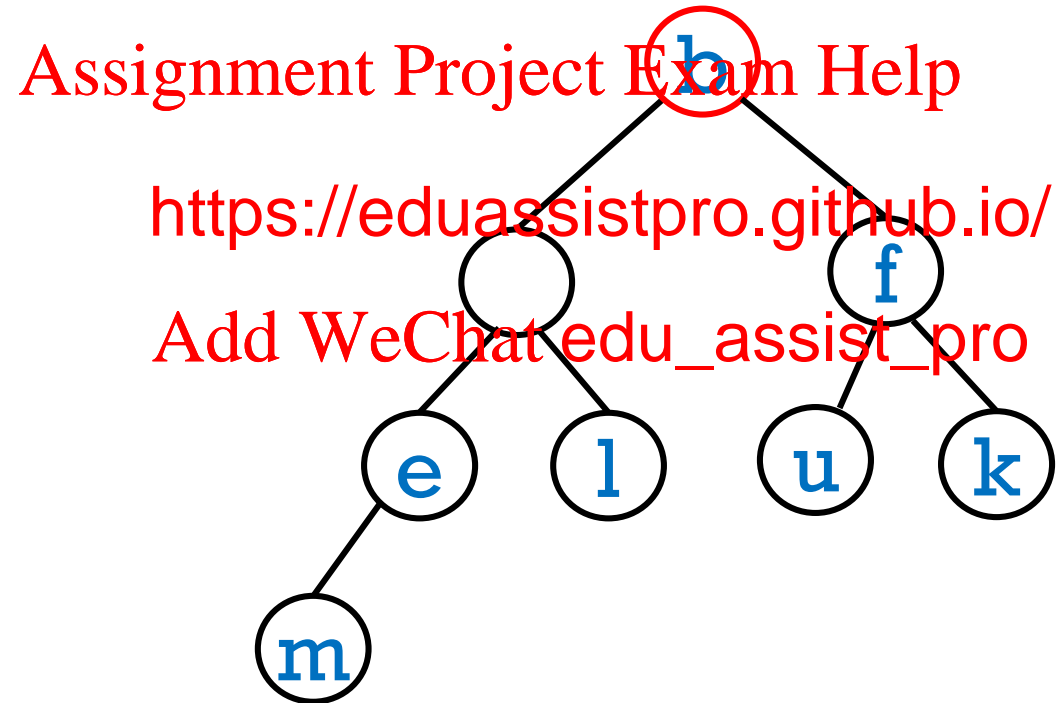
Add WeChat edu\_assist\_pro





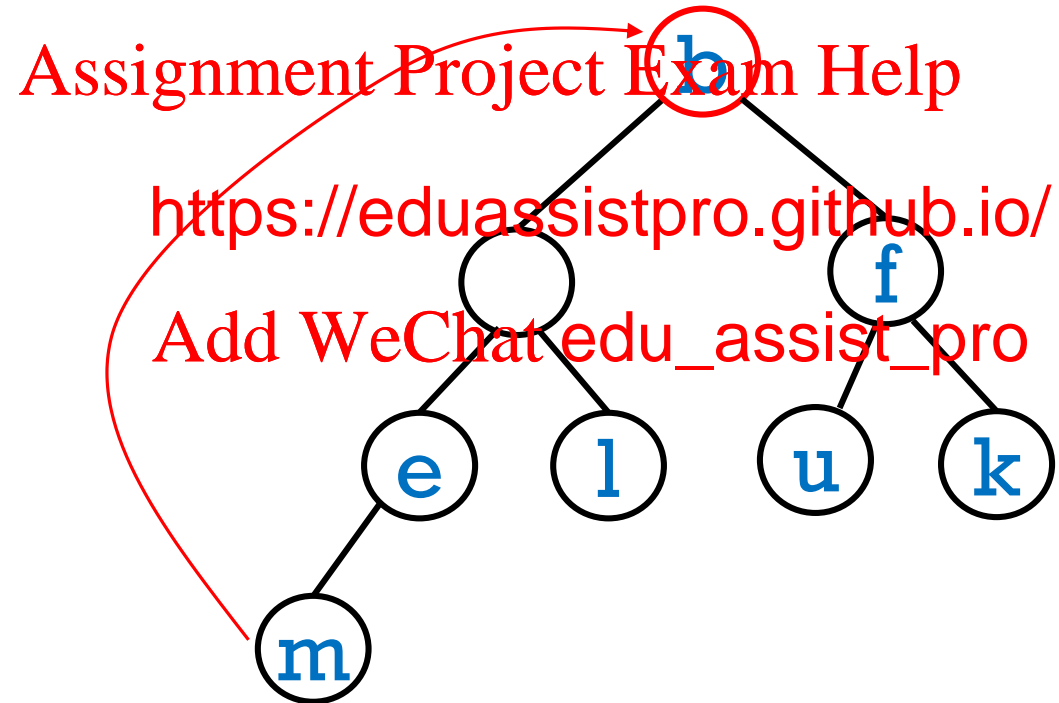
## REMOVEDMIN()

Let's removeMin() again!



## REMOVEDMIN()

Let's removeMin() again!



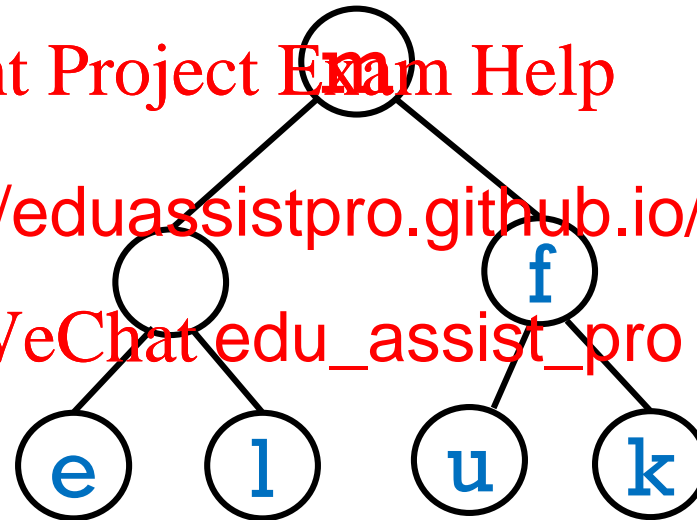
## REMOVEDMIN()

Let's removeMin() again!

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro



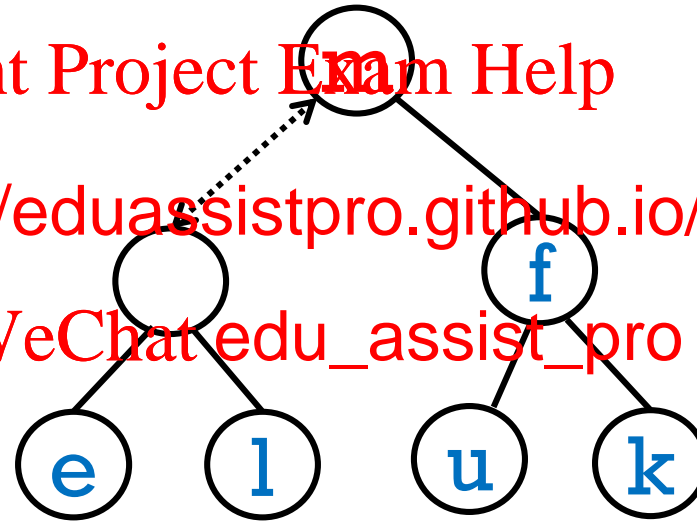
## REMOVEDMIN()

Now swap with smaller child, if necessary, to preserve heap property.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro



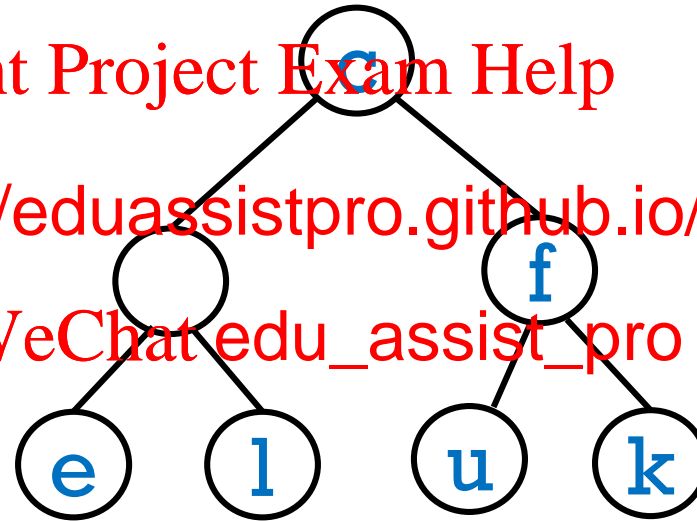
## REMOVEDMIN()

Now swap with smaller child, if necessary, to preserve heap property.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro



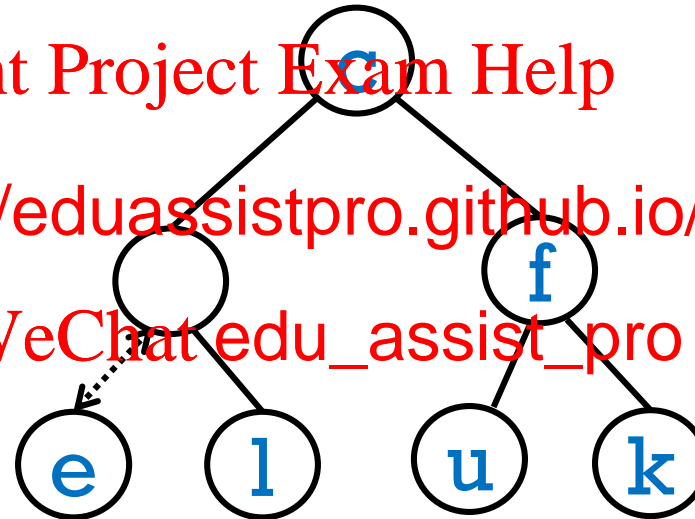
## REMOVED MIN()

Keep swapping with  
smaller child, if necessary.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro



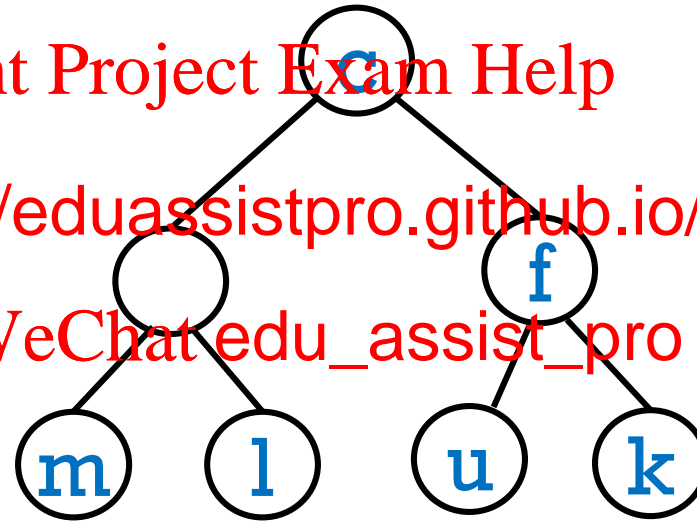
## REMOVED MIN()

Keep swapping with  
smaller child, if necessary.

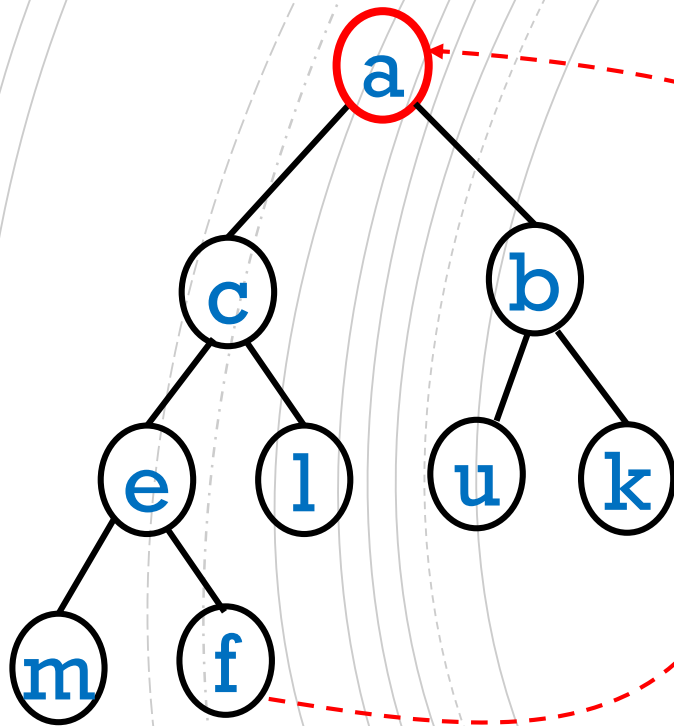
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro



## REMOVEDMIN() - IMPLEMENTATION

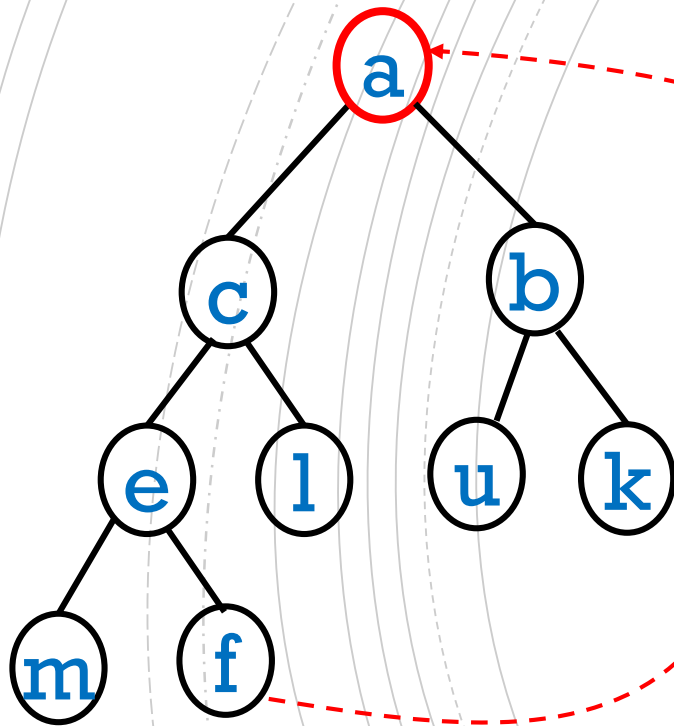


**Assignment Project Exam Help**  
<https://eduassistpro.github.io/>  
Add WeChat edu\_assist\_pro

```
removeMin () {  
    temp = root.key  
    remove the last leaf node and  
           o the root  
  
    return temp  
}
```



## REMOVEDMIN() - IMPLEMENTATION



```
removeMin () {
```

```
    temp = root.key  
    remove the last leaf node and  
           o the root
```

<https://eduassistpro.github.io/>

```
    while ((cur != null) && cur.key > cur.left.key)  
        cur = cur.left
```

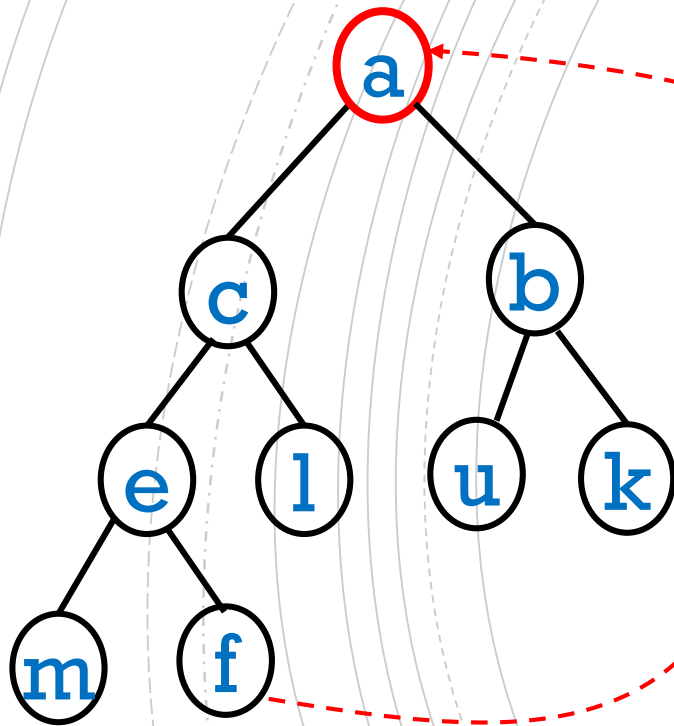
Add WeChat [edu\\_assist\\_pro](https://eduassistpro.github.io/)

```
    if (cur != null) {  
        cur.key = cur.right.key  
        cur.right = removeMin()  
    }
```

```
    }  
    return temp
```

```
}
```

## REMOVEDMIN() - IMPLEMENTATION



```
removeMin () {
```

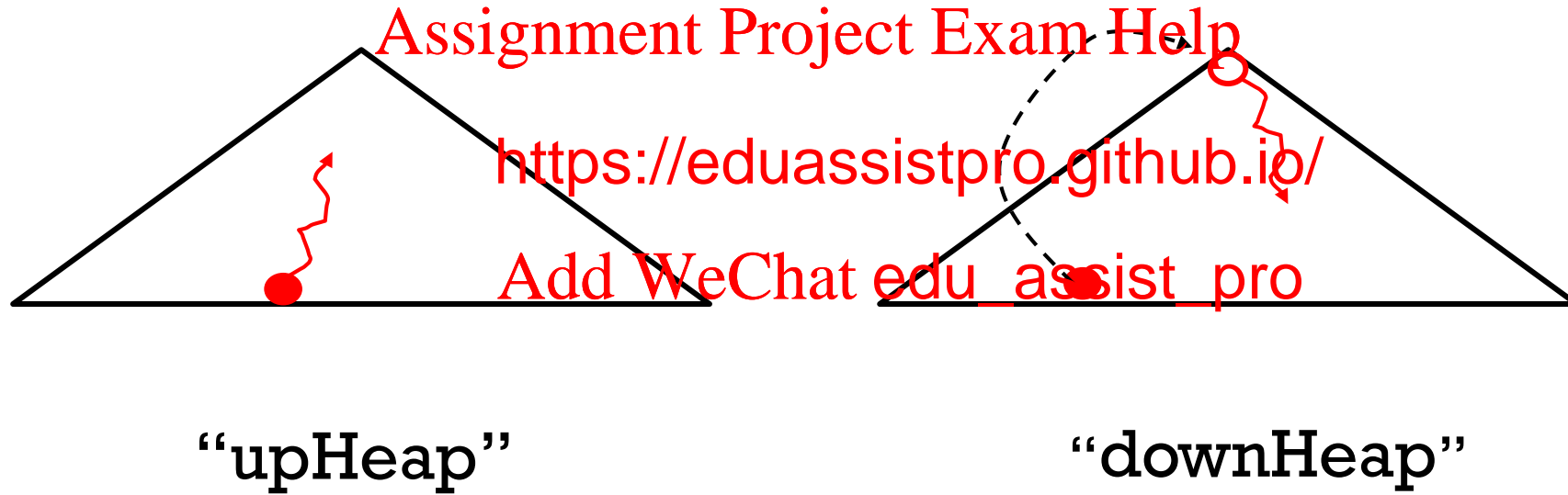
*temp = root key  
remove the last leaf node and  
o the root*

<https://eduassistpro.github.io/>

```
while ((cur != null && cur.key > cur.left.key)
|| (cur.right != null & cur.key > cur.right.key)) {
    minChild = child with smaller key
    swapKeys(cur, minChild)
    cur = minChild
}
return temp
}
```

add()

removeMin()



# REMOVE()

---

Q: What about remove(key) ?

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

## REMOVE()

Q: What about remove(key) ?

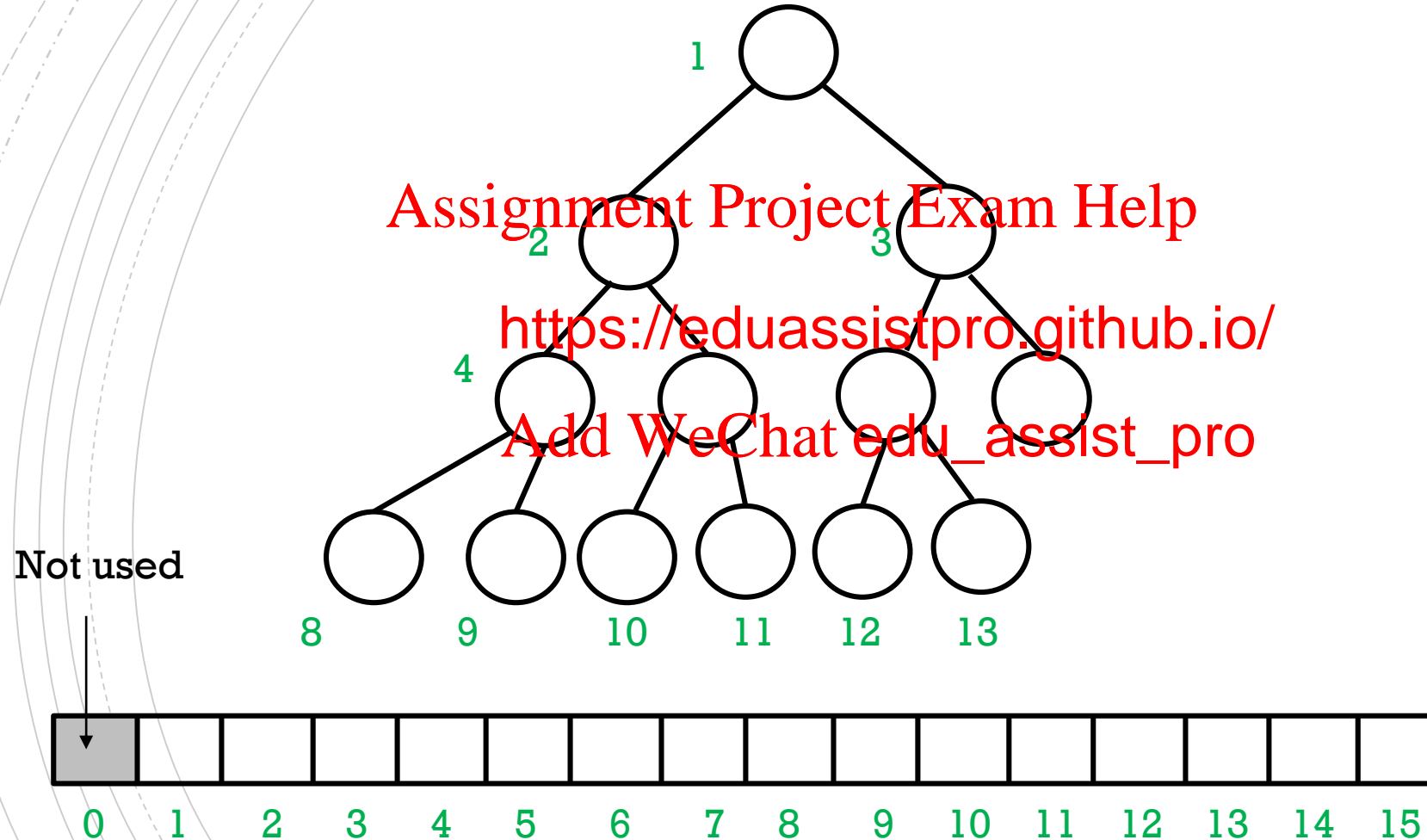
Assignment Project Exam Help

A: Worst case  $O()$

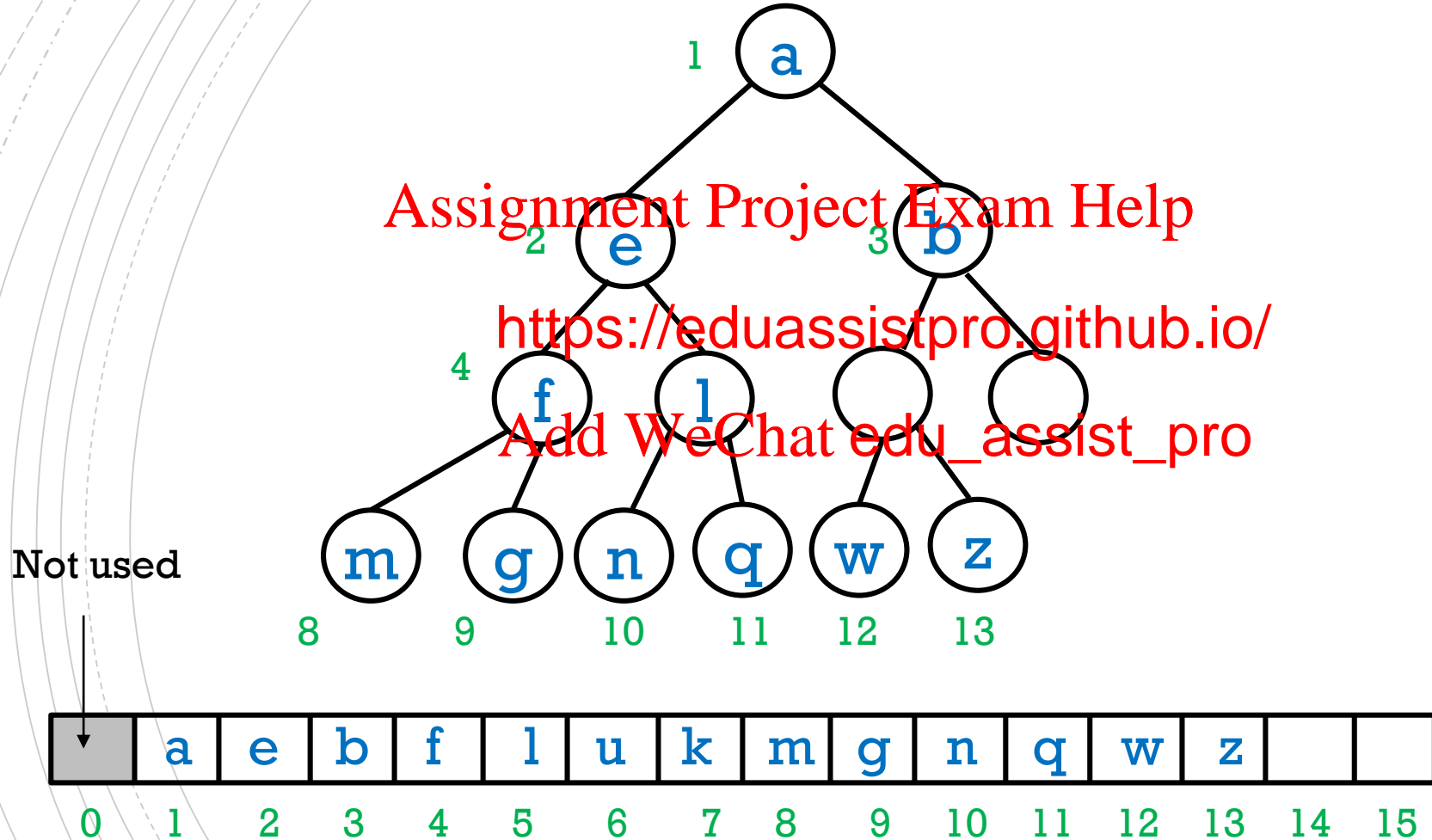
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

# HEAP (ARRAY IMPLEMENTATION)

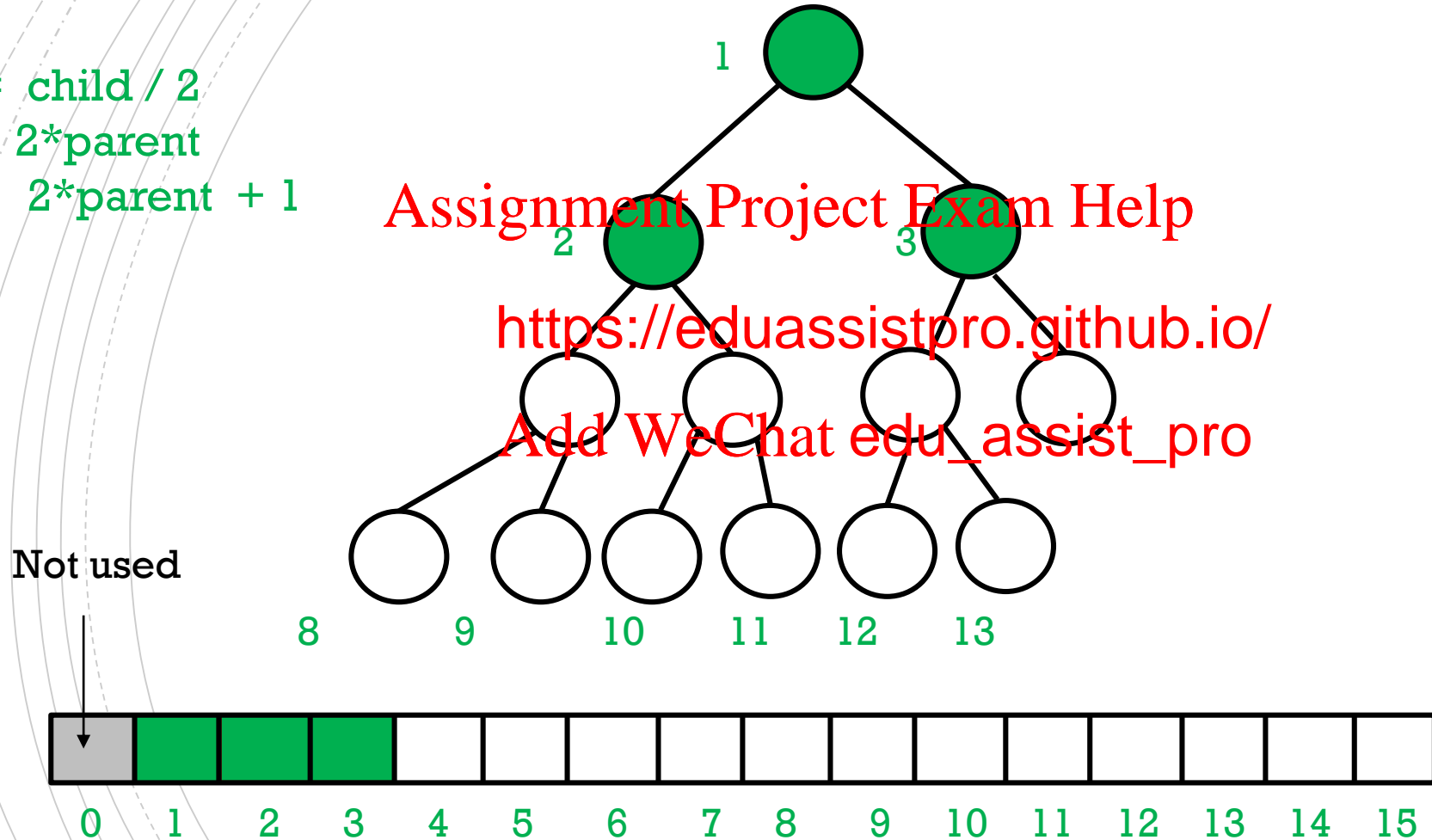


# HEAP (ARRAY IMPLEMENTATION)



# HEAP INDEX RELATIONS

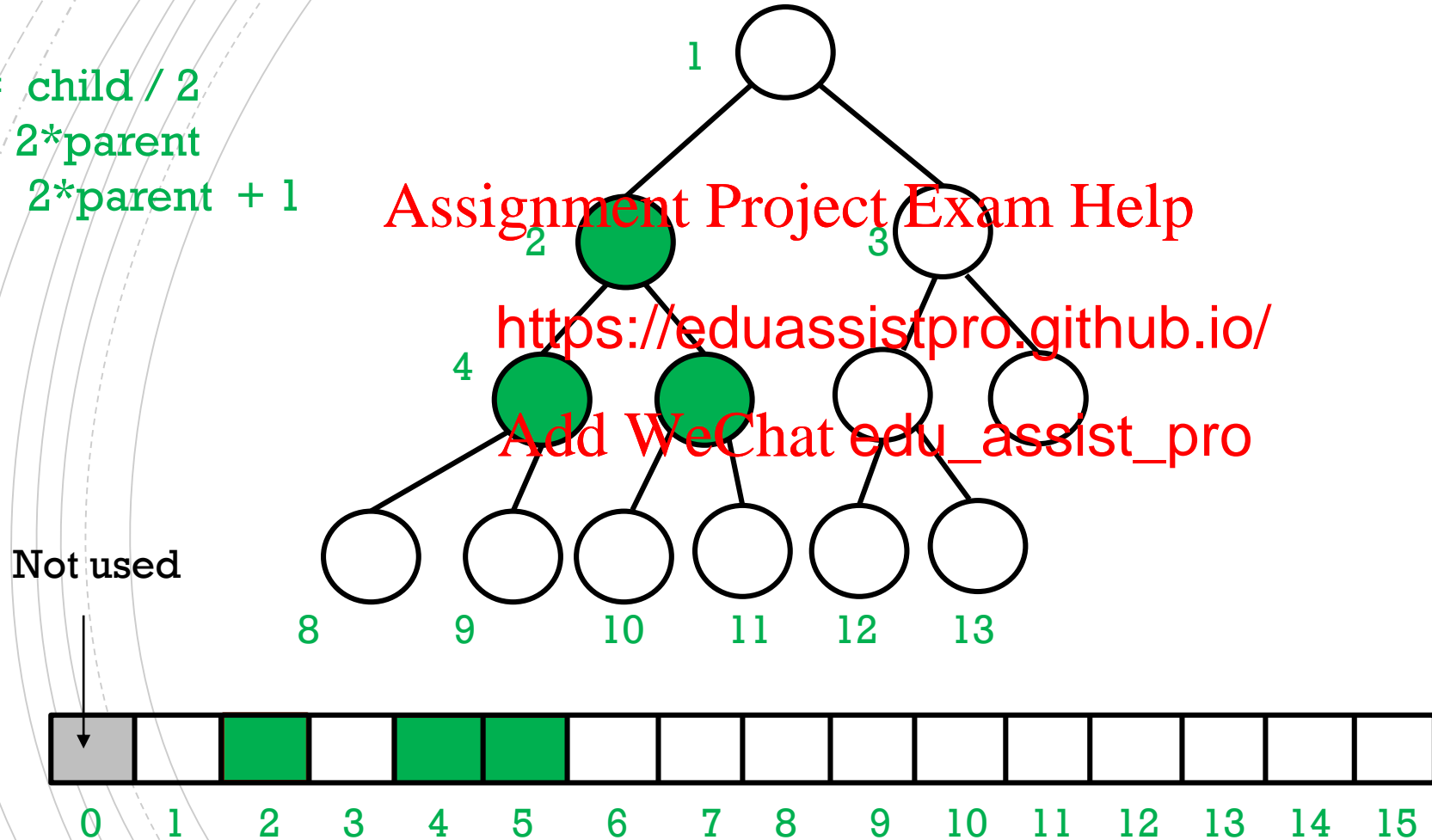
parent = child / 2  
left = 2\*parent  
right = 2\*parent + 1





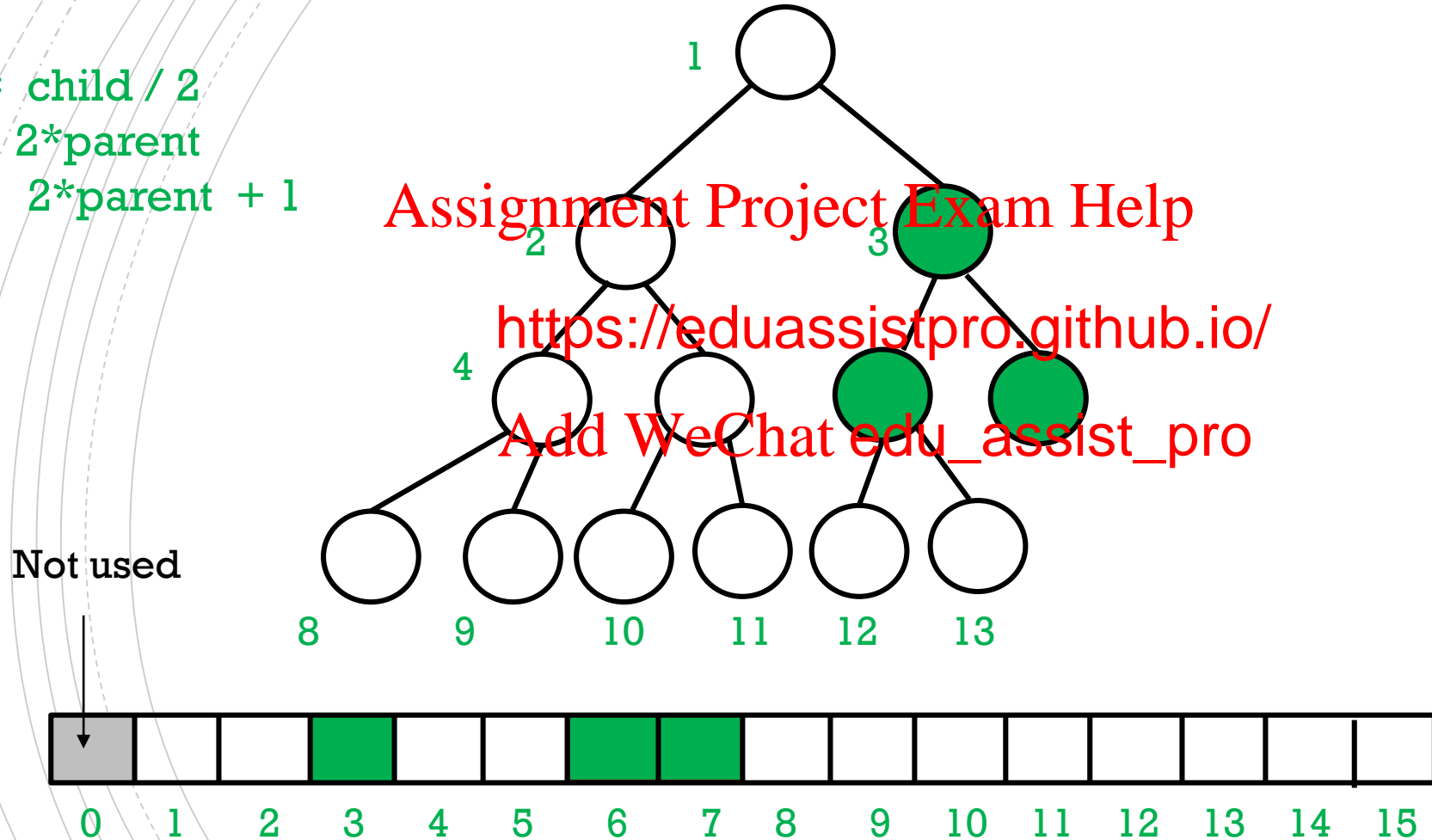
# HEAP INDEX RELATIONS

parent = child / 2  
left = 2\*parent  
right = 2\*parent + 1



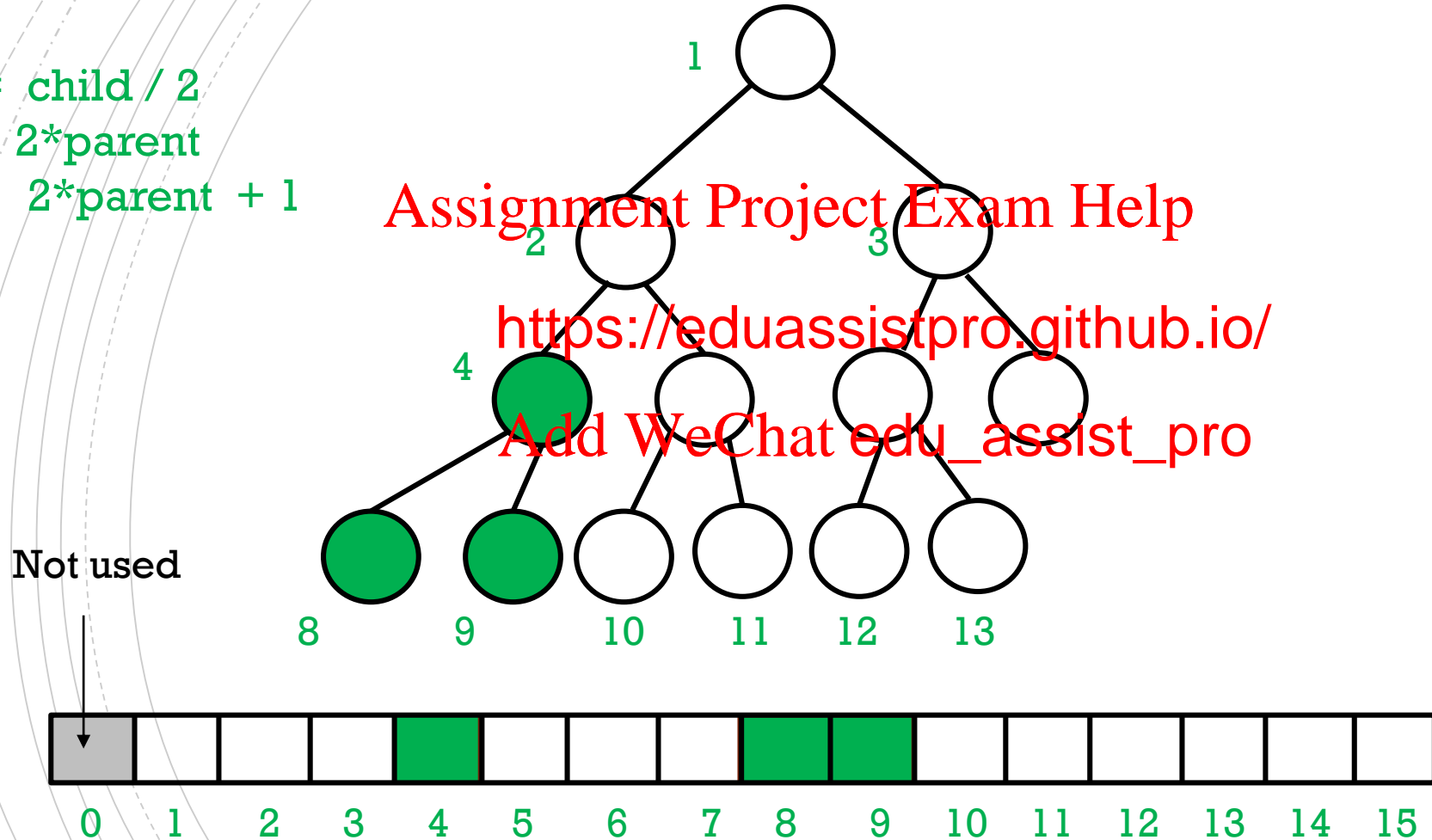
# HEAP INDEX RELATIONS

```
parent = child / 2
left   = 2*parent
right  = 2*parent + 1
```

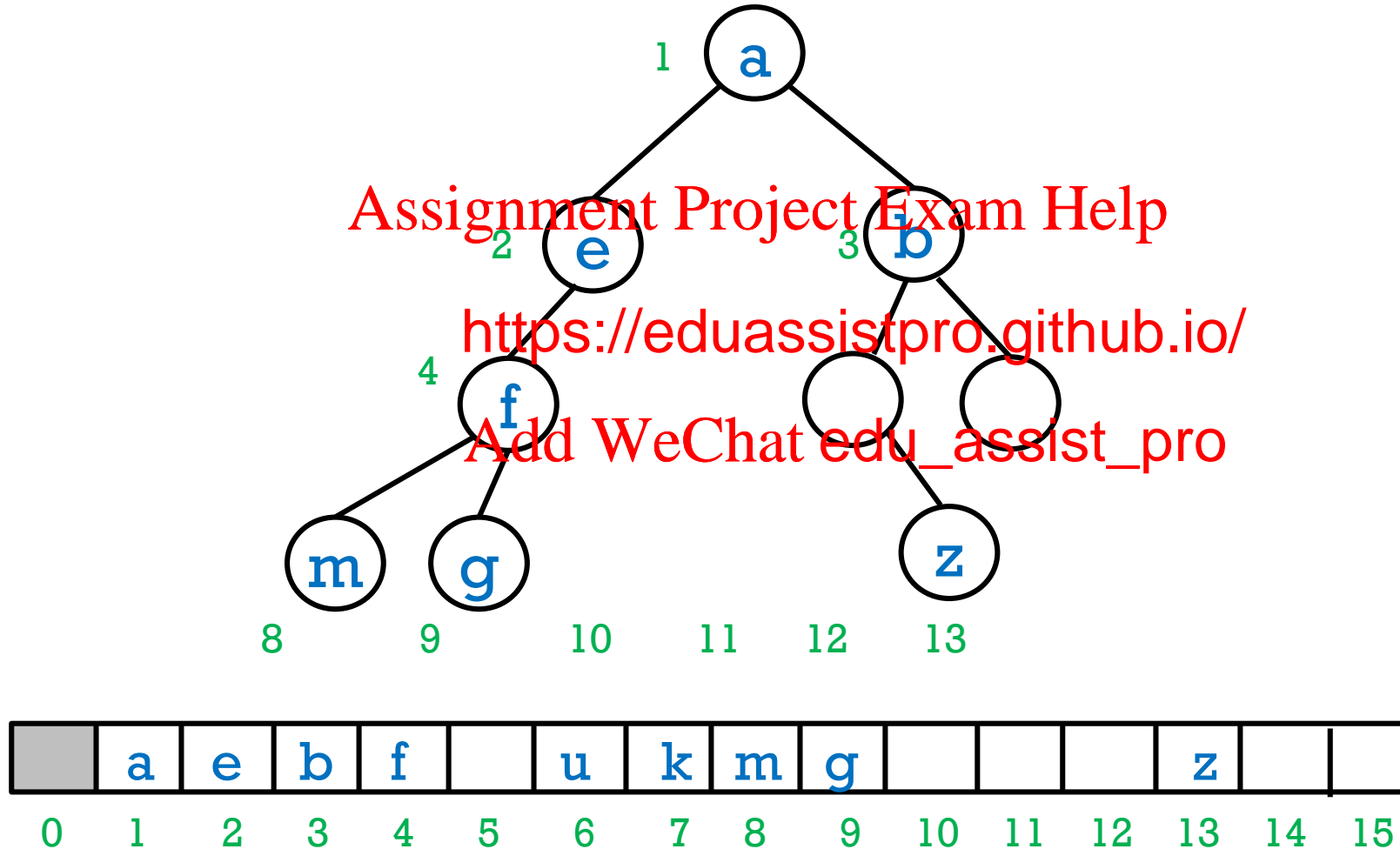


# HEAP INDEX RELATIONS

parent = child / 2  
left = 2\*parent  
right = 2\*parent + 1



ASIDE: an array data structure can be used for *any* binary tree. But this is uncommon and often inefficient.



## ADD() - IMPLEMENTATION

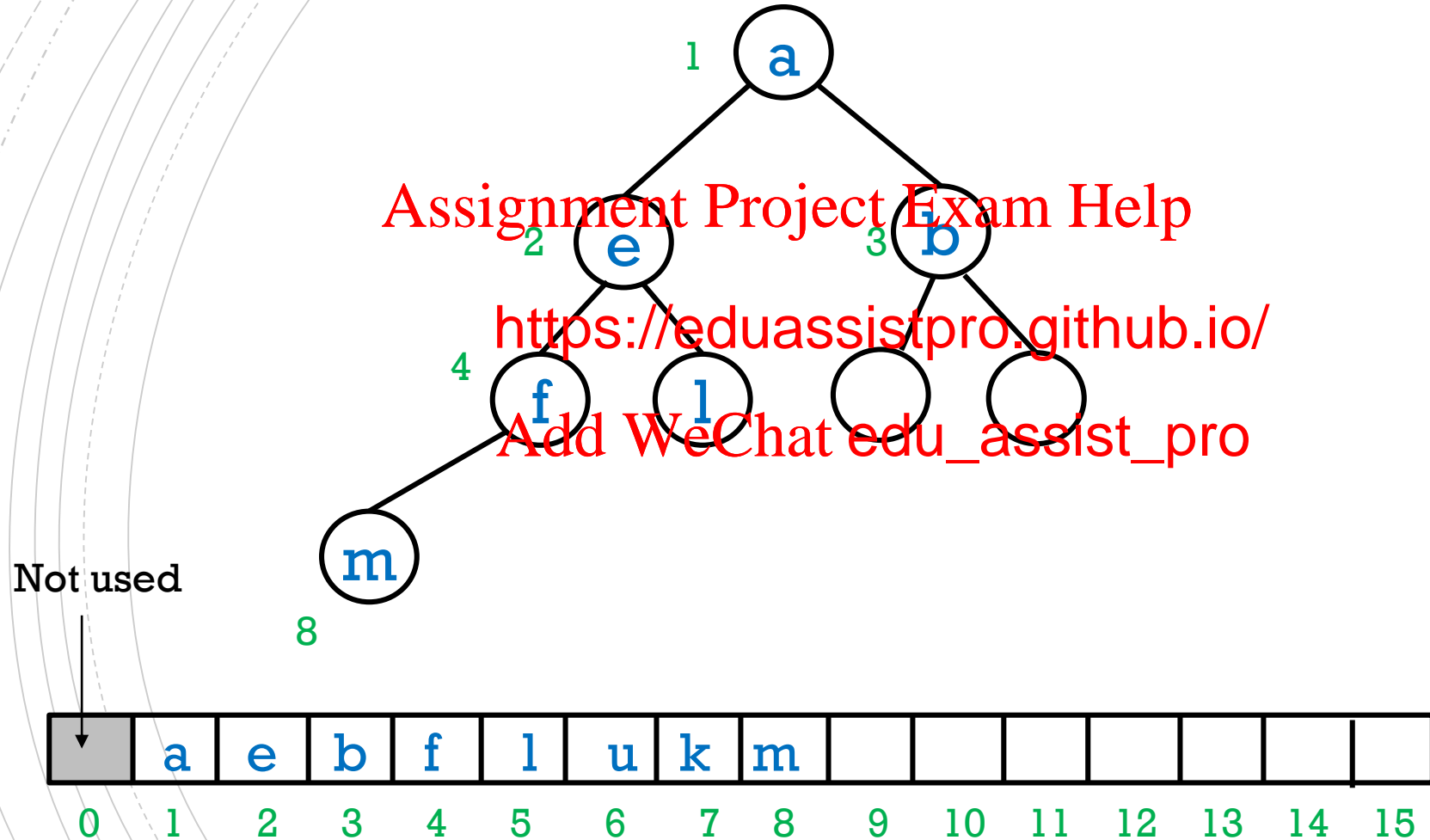
```
add(key) {  
    size = size + 1 // number of elements in heap  
    // assuming no other element  
    heap[ size ] = key  
    i = size  
    // the following is sometimes called "upHeap"  
    while ( i > 1 && heap[i] < heap[ i/2 ] ){  
        swapElements( i, i/2 )  
        i = i/2  
    }  
}
```

Assignment Project Exam Help

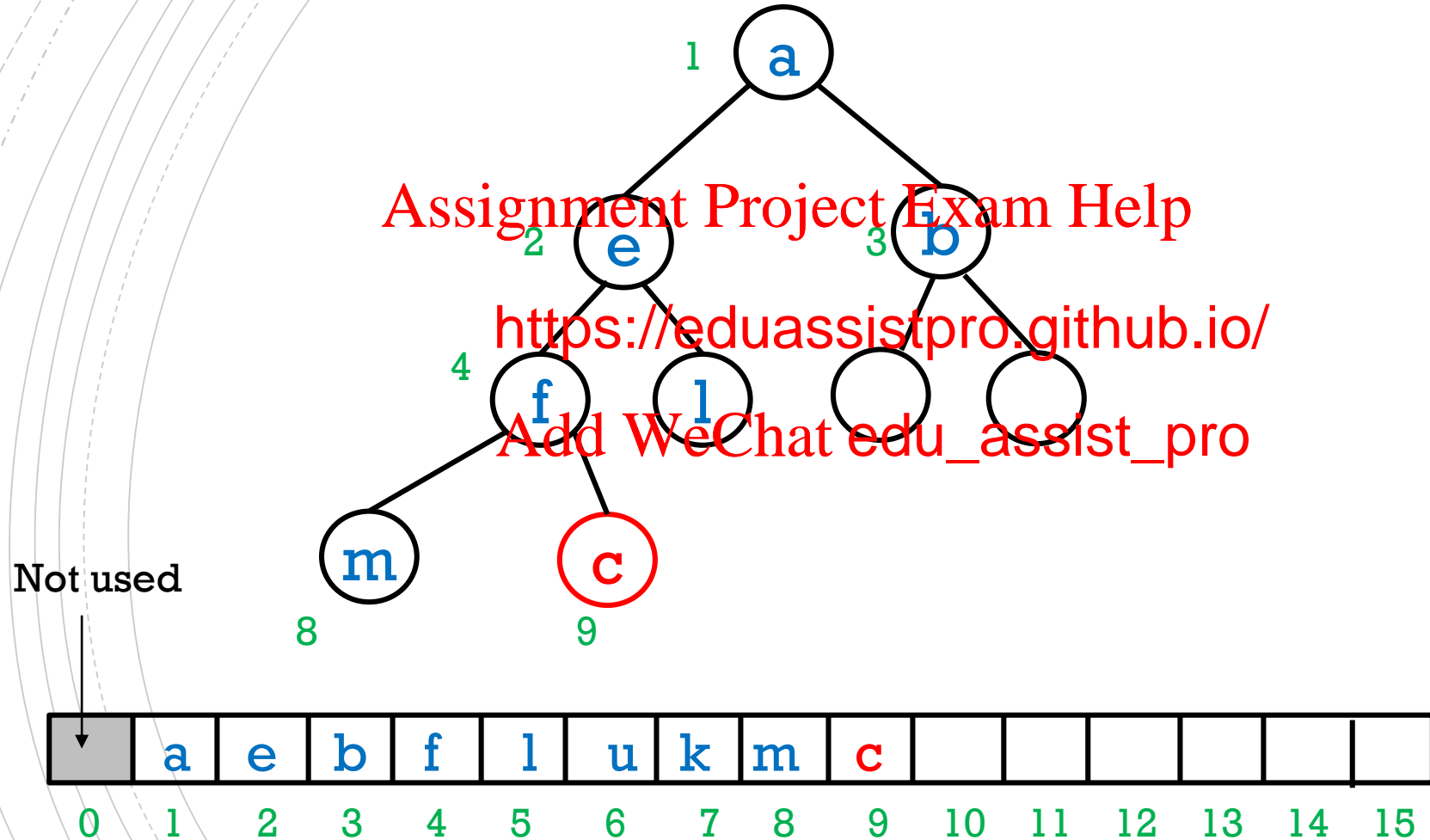
<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

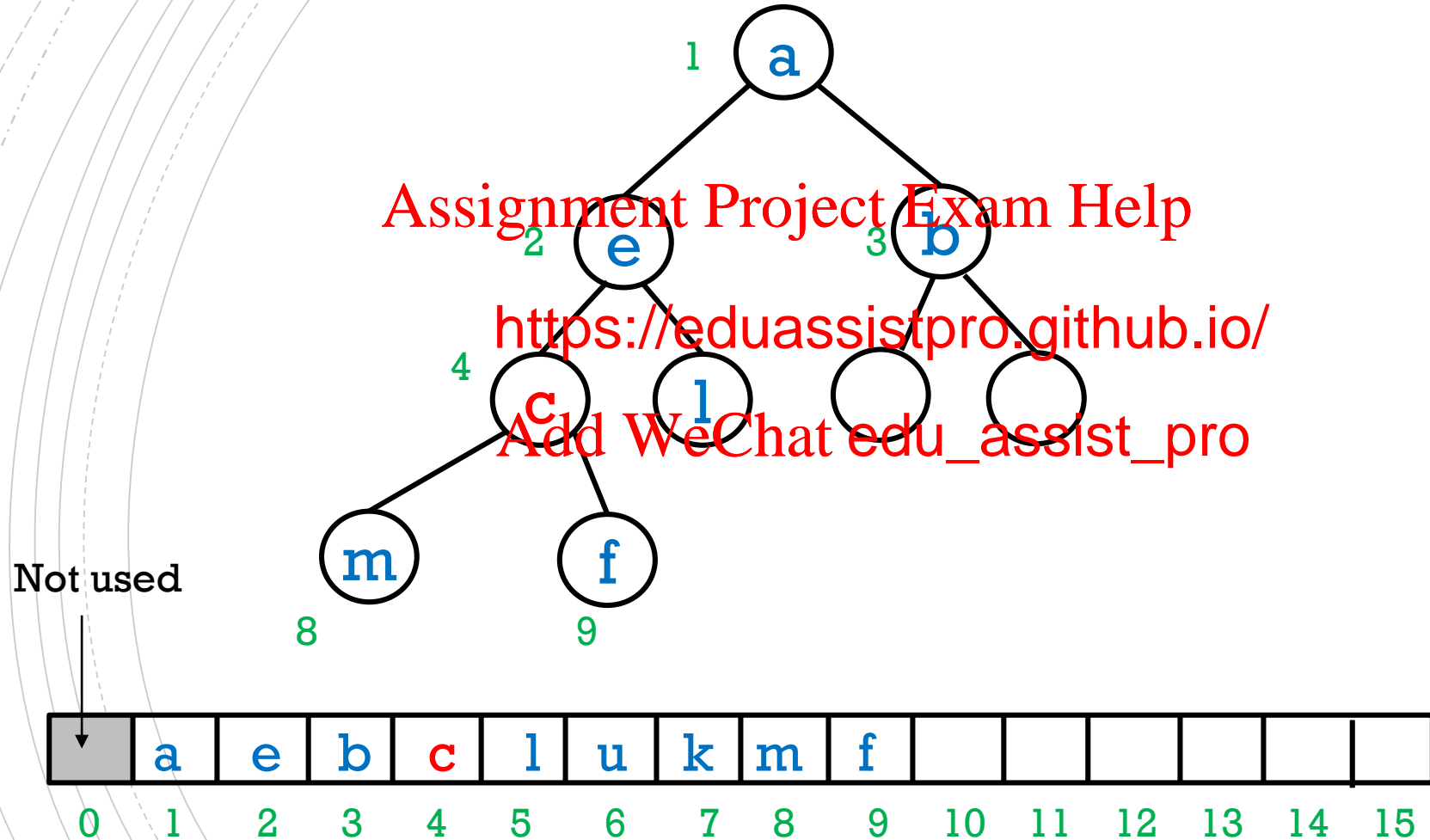
E.G. add (c)



E.G. add (c)

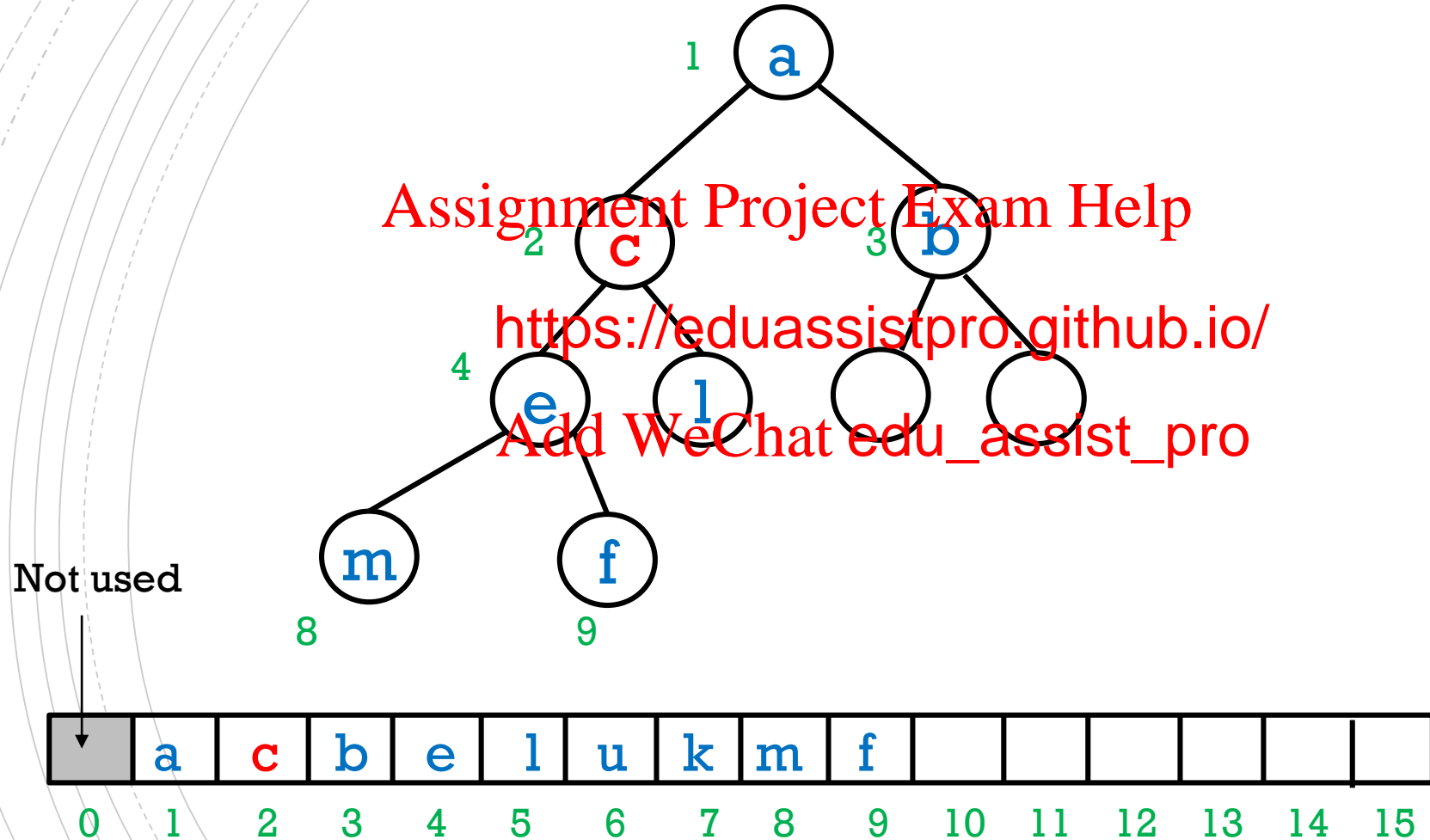


E.G. add (c)





E.G. add (c)



## NEXT VIDEO

---

- write `removeMin()` using array indices

Assignment Project Exam Help

- discuss best and worst case

<https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

- faster algorithm for building a heap



# Coming Soon

## Assignment Project Exam Help

In the next

■ More on <https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro