

# COMP 250

INTRODUCTORY SCIENCE

Assignment Project Exam Help

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

Week 10-21 Re Add WeChat edu\_assist\_pro

Giulia Alberini, Fall 2020

# WHAT ARE WE GOING TO DO IN THIS VIDEO?



■ Recurrences      Assignment Project Exam Help

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

Add WeChat edu\_assist\_pro

# ALGORITHM ANALYSIS

- We would like to find a function  $T(n)$  that describes the running time of an algorithm given an input size  $n$ .

Assignment Project Exam Help

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

- It is relatively easy to determine  $T(n)$  for algorithms only have loops. (e.g. insertion sort)

Add WeChat edu\_assist\_pro

- But how do we determine  $T(n)$  for a recursive algorithm?

# ALGORITHM ANALYSIS

**Example:** Suppose a list has  $n$  elements, what is  $T(n)$  for the following algorithm?

Assignment Project Exam Help

```
reverse(list) {  
    if(list.size() == 1) {  
        return;  
    }  
    firstElement = list.removeFirst();  
    reverse(list); // now the list has n-1 elements  
    list.addLast(firstElement);  
}
```

# RECURRENCES

“A recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs”

e.g. Fibona  $F(n) = F(n-1) + F(n-2)$

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

Add WeChat edu\_assist\_pro

We use recurrences to express the overall running time  $T(n)$  of an algorithm with input size  $n$  in terms of the running time on smaller inputs.

Note that for Fibonacci number  $n$  is an input value. It is NOT the input size!

## EXAMPLE 1: REVERSING A LIST

```
reverse(list) {
```

```
    if (list.size() == 1) {  
        return;
```

```
    }
```

```
    firstElement
```

```
    reverse(list);
```

```
    list.addLast(firstElement);
```

```
}
```

Assignment Project Exam Help

Base case

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

Add WeChat edu\_assist\_pro

Recursive step

$$T(1) = b,$$

$$T(n) = c + T(n - 1)$$

## OBSERVATIONS

**Q:** What assumptions are we making about `removeFirst()` and `addLast()` ?

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

**A:** They can be executed in constant time.  
(Note that this is not true if we use an `ArrayList`.)

# HOW TO SOLVE A RECURRENCE

There are different methods used to try to solve a recurrence:

**Assignment Project Exam Help**

- Forward sub <https://eduassistpro.github.io/>
- Back substitution **Add WeChat edu\_assist\_pro**
- Recursion-tree method
- Master Theorem

:



# HOW TO SOLVE A RECURRENCE

There are different methods used to try to solve a recurrence:

**Assignment Project Exam Help**

- Forward sub <https://eduassistpro.github.io/>
- Back substitution Add WeChat edu\_assist\_pro
- Recursion-tree method
- Master Theorem

:

## SOLVING THE RECURRENCE USING BACK SUBSTITUTION

$$T(n) = c + T(n - 1)$$

$$= c + c + T(n - 2)$$

Assignment Project Exam Help

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

Add WeChat edu\_assist\_pro

## SOLVING THE RECURRENCE USING BACK SUBSTITUTION

$$T(n) = c + T(n - 1)$$

$$= c + c + T(n - 2)$$

$$= c$$

<https://eduassistpro.github.io/>  
Add WeChat edu\_assist\_pro

## SOLVING THE RECURRENCE USING BACK SUBSTITUTION

$$T(n) = c + T(n - 1)$$

$$= c + c + T(n - 2)$$

$$= c$$

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

$$= 3c + T(n - 3) =$$

Add WeChat [edu\\_assist\\_pro](#)

$$= kc + T(n - k) = \dots$$

$$= (n - 1)c + T(1)$$

$$= (n - 1)c + b = c \cdot n + (b - c)$$

which is  $\Theta(n)$ .

## EXAMPLE 2: SORTING A LIST

```
sort(list) {
```

```
    if (list.size() == 1) {  
        return;  
    }
```

Assignment Project Exam Help  
Base case

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

```
    minElement
```

```
    sort(list);
```

```
    list.addFirst(minElement);
```

Add WeChat edu\_assist\_pro  
Recursive step

```
}
```

$$T(1) = a, \quad T(n) = b + c \cdot n + T(n - 1)$$

## OBSERVATIONS

**Q:** What assumptions are we making about `addFirst()` ?

Assignment Project Exam Help

**A:** That can be executed <https://eduassistpro.github.io/>

Add WeChat edu\_assist\_pro

**Q:** It would be ok if this step uses time proportional to  $n$ . Why??

**A:** Because `removeMin()` already takes time proportional to  $n$ .

# SOLVING THE RECURRENCE USING BACK SUBSTITUTION

Let's solve the following slightly simpler recurrence:

$$T(n) = cn + T(n - 1)$$

$$= cn + c(n - 1) + T(n - 2)$$

Assignment Project Exam Help

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

Add WeChat edu\_assist\_pro

# SOLVING THE RECURRENCE USING BACK SUBSTITUTION

Let's solve the following slightly simpler recurrence:

$$T(n) = cn + T(n - 1)$$

$$= cn + c(n - 1) + T(n - 2)$$

$$= cn + c(n - 1) + \dots$$

Add WeChat edu\_assist\_pro



## SOLVING THE RECURRENCE USING BACK SUBSTITUTION

Let's solve the following slightly simpler recurrence:

$$T(n) = cn + T(n - 1)$$

$$= cn + c(n - 1) + T(n - 2)$$

$$= cn + c(n - 1) + T(n - 3)$$

...

$$= c[n + (n - 1) + (n - 2) + \dots + (n - k + 1)] + T(n - k + 1)$$

$$= c[n + (n - 1) + (n - 2) + \dots + 2] + T(1), \text{ when } k = n - 1$$

Assignment Project Exam Help

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

Add WeChat edu\_assist\_pro

# SOLVING THE RECURRENCE USING BACK SUBSTITUTION

Let's solve the following slightly simpler recurrence:

$$T(n) = cn + T(n - 1)$$

$$= cn + c(n - 1) + T(n - 2)$$

$$= cn + c(n - 1) + T(n - 3)$$

...

$$= c[n + (n - 1) + (n - 2) + \dots + 1] + T(1)$$

$$= c[n + (n - 1) + (n - 2) + \dots + 1] + T(1), \text{ when } k = n - 1$$

$$= \frac{1}{2}cn^2 + \frac{1}{2}cn - c + a$$

which is  $\Theta(n^2)$ .

$$\sum_{i=1}^n i = \frac{1}{2}n(n + 1)$$

## EXAMPLE 3: TOWER OF HANOI

```
tower(n, start, finish, other) {  
    if (n > 0) {  
        tower(n-1, other, start, finish)  
        move from start to finish  
        tower(n-1, start, finish, other)  
    }  
}
```

Base case is  $n=0$

Recursive step

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

$$T(0) = b, \quad T(n) = c + 2T(n - 1)$$

## SOLVING THE RECURRENCE USING BACK SUBSTITUTION

$$T(n) = c + 2T(n - 1)$$

$$= c + 2(c + 2T(n - 2))$$

$$= c(1 + 2) + 4T(n - 2)$$

Assignment Project Exam Help

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

Add WeChat edu\_assist\_pro

## SOLVING THE RECURRENCE USING BACK SUBSTITUTION

$$T(n) = c + 2T(n - 1)$$

$$= c + 2(c + 2T(n - 2))$$

$$= c(1 + 2) + 4T(n - 2)$$

$$= c(1 + 2) + 4[c + 2T(n - 3)]$$

$$= c(1 + 2 + 4) + 8T(n - 3)$$

Assignment Project Exam Help

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

Add WeChat edu\_assist\_pro

## SOLVING THE RECURRENCE USING BACK SUBSTITUTION

$$T(n) = c + 2T(n - 1)$$

$$= c + 2(c + 2T(n - 2))$$

$$= c(1 + 2) + 4T(n - 2)$$

$$= c(1 + 2) + 4[c + 2T(n - 3)]$$

$$= c(1 + 2 + 4) + 8T(n - 3)$$

...

$$= c[1 + 2 + 4 + \dots + 2^{k-1}] + 2^k T(n - k)$$

$$= c[1 + 2 + 4 + \dots + 2^{n-1}] + 2^n T(0), \text{ when } k = n$$

Assignment Project Exam Help

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

Add WeChat edu\_assist\_pro

## SOLVING THE RECURRENCE USING BACK SUBSTITUTION

$$T(n) = c + 2T(n - 1)$$

$$= c + 2(c + 2T(n - 2))$$

$$= c(1 + 2) + 4T(n - 2)$$

$$= c(1 + 2) + 4[c + 2T(n - 3)]$$

$$= c(1 + 2 + 4) + 8T(n - 3)$$

...

$$= c[1 + 2 + 4 + \dots + 2^{k-1}] + 2^k T(n - k)$$

$$= c[1 + 2 + 4 + \dots + 2^{n-1}] + 2^n T(0), \text{ when } k = n$$

$$= c(2^n - 1) + 2^n b = (c + b)2^n - c$$

which is  $\Theta(2^n)$ .

Assignment Project Exam Help

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

Add WeChat edu\_assist\_pro

$$\sum_{i=0}^n r^i = \frac{r^{n+1} - 1}{r - 1}$$

YOU SHOULD KNOW...

$$1 + 2 + 3 + \dots + k = ?$$

Assignment Project Exam Help

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

$$1 + 2 + 4 + 8 + \dots = ?$$

Add WeChat edu\_assist\_pro

$$1 + x + x^2 + x^3 + \dots + x^k = ?$$



## EXAMPLE 4: BINARY SEARCH

```
binarySearch(list, key, left, right){
    if(left <= right){
        mid = (left + right)/2
        if(list[mid]==key)
            return mid
        else {
            if(key<list[mid])
                return binarySearch(list, key, left, mid-1)
            else
                return binarySearch(list, key, mid+1, right)
        }
    }
    return -1
}
```

Assignment Project Exam Help

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

Add WeChat edu\_assist\_pro

$$T_w(1) = b, \quad T_w(n) = c + T\left(\frac{n}{2}\right)$$

## SOLVING THE RECURRENCE USING BACK SUBSTITUTION

To simplify our analysis let's assume  $n$  is a power of 2. This will not affect the order of growth of the solution.

$$\begin{aligned} T(n) &= c + T(n/2) \\ &= c + \end{aligned}$$

Assignment Project Exam Help

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

Add WeChat edu\_assist\_pro

## SOLVING THE RECURRENCE USING BACK SUBSTITUTION

To simplify our analysis let's assume  $n$  is a power of 2. This will not affect the order of growth of the solution.

$$T(n) = c + T(n/2)$$

$$= c +$$

$$= c +$$

Add WeChat edu\_assist\_pro

## SOLVING THE RECURRENCE USING BACK SUBSTITUTION

To simplify our analysis let's assume  $n$  is a power of 2. This will not affect the order of growth of the solution.

$$T(n) = c + T(n/2)$$

$$= c +$$

$$= c +$$

$$\dots$$

$$= c \cdot k + T(n/2^k)$$

$$= c \cdot \log_2 n + T(1), \text{ when } k = \log_2 n$$

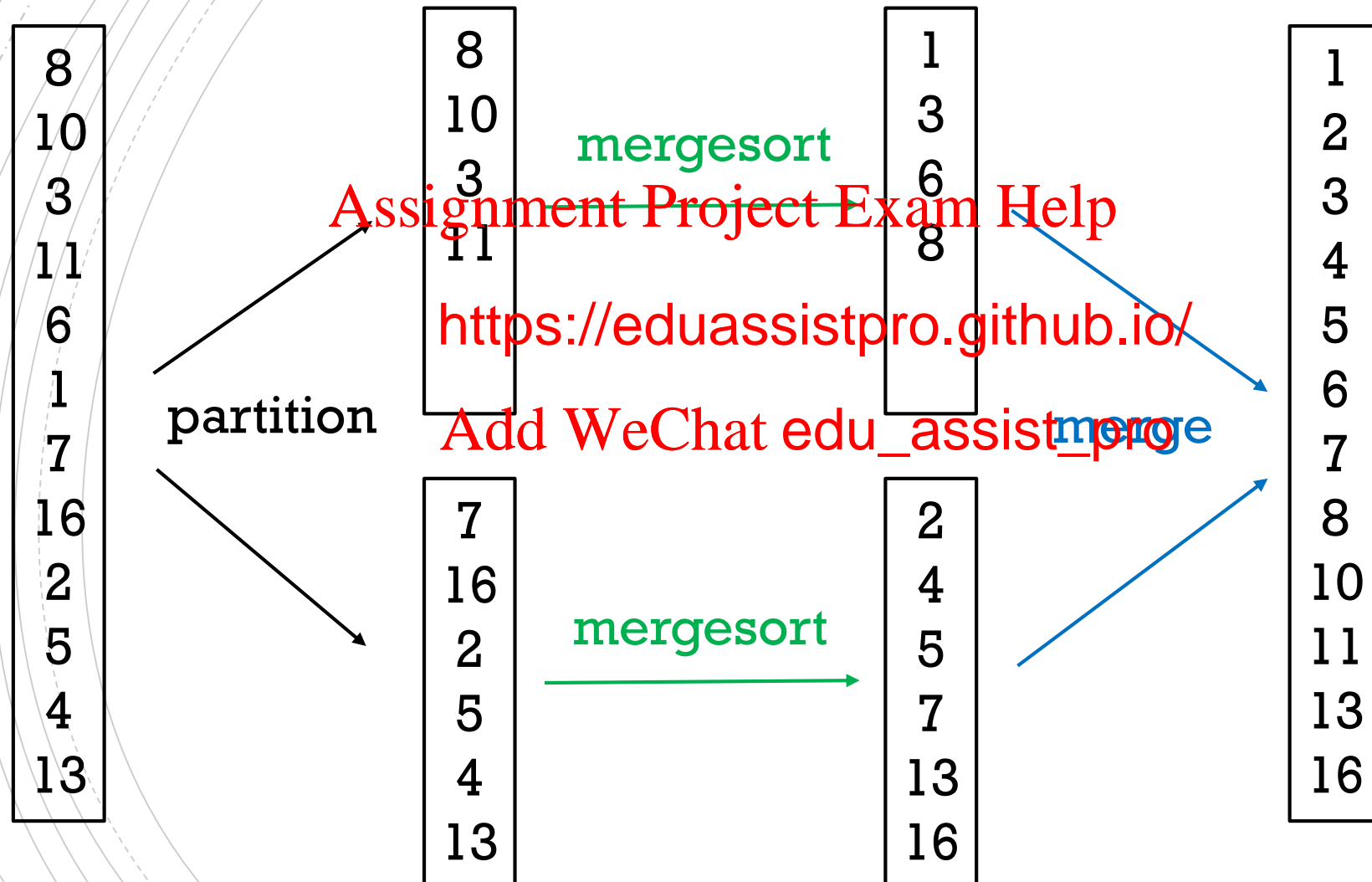
## SOLVING THE RECURRENCE USING BACK SUBSTITUTION

To simplify our analysis let's assume  $n$  is a power of 2. This will not affect the order of growth of the solution.

$$\begin{aligned} T(n) &= c + T(n/2) \\ &= c + \text{Assignment Project Exam Help} \\ &= c + \text{https://eduassistpro.github.io/} \\ &\dots \text{Add WeChat edu\_assist\_pro} \\ &= c \cdot k + T(n/2^k) \\ &= c \cdot \log_2 n + T(1), \text{ when } k = \log_2 n \\ &= c \cdot \log_2 n + b \end{aligned}$$

which is  $\Theta(\log_2 n)$ .

## RECALL MERGESORT



## EXAMPLE 5: MERGESORT

```
mergesort(list) {
```

```
    if (list.size() == 1)
        return list
```

Base case

```
    else {
```

```
        mid = (list.si
```

```
        list1 = list.g
```

```
        list2 = list.g
```

```
        list1 = mergesort(list1)
```

```
        list2 = mergesort(list2)
```

```
        return merge(list1, list2)
```

```
    }
```

```
}
```

Assignment Project Exam Help

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

Add WeChat edu\_assist\_pro

( ) - 1)

Recursive step

$$T(1) = a, \quad T(n) = b + c \cdot n + 2 \cdot T\left(\frac{n}{2}\right)$$

Let's ignore the constant term for simplicity

## WHAT IF $n$ IS NOT EVEN?

Example:  $t(13) = c * 13 + t(6) + t(7)$

Assignment Project Exam Help

In general, one should wr

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

$$T(n) = c n + T\left(\text{floor}\left(\frac{n}{2}\right)\right) + T\left(\text{ceiling}\left(\frac{n}{2}\right)\right)$$

In COMP250, one typically assumes  $n = 2^k$  for recurrences that involve  $T\left(\frac{n}{2}\right)$ .

The more general recurrence has roughly the same solution.



# SOLVING THE RECURRENCE USING BACK SUBSTITUTION

To simplify our analysis let's assume  $n$  is a power of 2.

$$T(n) = cn + 2T\left(\frac{n}{2}\right)$$

$$= cn + 2\left(c\frac{n}{2} + 2T\left(\frac{n}{4}\right)\right) = cn + cn + 4T\left(\frac{n}{4}\right)$$

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

Add WeChat edu\_assist\_pro

# SOLVING THE RECURRENCE USING BACK SUBSTITUTION

To simplify our analysis let's assume  $n$  is a power of 2.

$$T(n) = cn + 2T\left(\frac{n}{2}\right)$$

$$= cn + 2\left(c\frac{n}{2} + 2T\left(\frac{n}{4}\right)\right) = cn + cn + 4T\left(\frac{n}{4}\right)$$

$$= cn + cn + 4\left(c\frac{n}{4} + 2T\left(\frac{n}{8}\right)\right) = cn + 8T\left(\frac{n}{8}\right)$$

Assignment Project Exam Help

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

Add WeChat edu\_assist\_pro

# SOLVING THE RECURRENCE USING BACK SUBSTITUTION

To simplify our analysis let's assume  $n$  is a power of 2.

$$T(n) = cn + 2T\left(\frac{n}{2}\right)$$

$$= cn + 2\left(c\frac{n}{2} + 2T\left(\frac{n}{4}\right)\right) = cn + cn + 4T\left(\frac{n}{4}\right)$$

$$= cn + cn + 4\left(c\frac{n}{4} + 2T\left(\frac{n}{8}\right)\right) = cn + cn + 4cn + 8T\left(\frac{n}{8}\right)$$

...

$$= cn \cdot k + 2^k T\left(\frac{n}{2^k}\right)$$

$$= cn \cdot \log_2 n + 2^{\log_2 n} T(1), \text{ when } k = \log_2 n$$

Assignment Project Exam Help

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

Add WeChat edu\_assist\_pro

# SOLVING THE RECURRENCE USING BACK SUBSTITUTION

To simplify our analysis let's assume  $n$  is a power of 2.

$$T(n) = cn + 2T\left(\frac{n}{2}\right)$$

$$= cn + 2\left(c\frac{n}{2} + 2T\left(\frac{n}{4}\right)\right) = cn + cn + 4T\left(\frac{n}{4}\right)$$

$$= cn + cn + 4\left(c\frac{n}{4} + 2T\left(\frac{n}{8}\right)\right) = cn + cn + 4cn + 8T\left(\frac{n}{8}\right)$$

...

$$= cn \cdot k + 2^k T\left(\frac{n}{2^k}\right)$$

$$= cn \cdot \log_2 n + 2^{\log_2 n} T(1), \text{ when } k = \log_2 n$$

$$= cn \log_2 n + bn$$

which is  $\Theta(n \log_2 n)$ .

Assignment Project Exam Help

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

Add WeChat edu\_assist\_pro

## TODAY'S RECURRENCES

$$T(n) = c + T(n - 1)$$

Assignment Project Exam Help

$$T(n) = c + 2T\left(\frac{n}{2}\right)$$

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

Add WeChat [edu\\_assist\\_pro](#)

$$T(n) = c + T\left(\frac{n}{2}\right)$$

$$T(n) = cn + T\left(\frac{n}{2}\right)$$

An orange paint roller with a red handle, positioned horizontally. The roller is partially covered in orange paint, which is dripping down the left side. The text "Coming Soon" is written in white on the orange surface of the roller.

# Coming Soon

## Assignment Project Exam Help

In the next

- Trees

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

Add WeChat edu\_assist\_pro