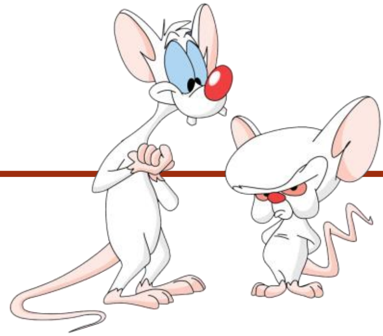
COMP 250

INTRODUC TER SCIENCE

Week 10-2 : Re

Giulia Alberini, Fall 2020

WHAT ARE WE GOING TO DO IN THIS VIDEO?



- Recurrences

ALGORITHM ANALYSIS

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

- It is relatively easy to determine $T($ r algorithms only have loops. (e.g. insertion sort fro

- 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?


```
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 ( ) ( ) $F(n-2)$

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 ← list.removeFirst();
    reverse(list);
    list.addLast(firstElement);
}
```

Base case

Recursive step

$$T(1) = b, \qquad T(n) = c + T(n-1)$$

OBSERVATIONS

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

A: They can be executed in consta

(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:

- Forward sub
- Back substitution
- Recursion-tree method
- Master Theorem

:

HOW TO SOLVE A RECURRENCE

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

- Forward sub

- Back substitution

- Recursion-tree method
- Master Theorem

:

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

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

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

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

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

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

$$= c$$

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

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

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

$$= c$$

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

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

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

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

which is $\Theta(n)$.

EXAMPLE 2: SORTING A LIST

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

Base case

Recursive step

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

OBSERVATIONS

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

A: That can be execu

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:

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

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

Let's solve the following slightly simpler recurrence:

$$\begin{aligned} T(n) &= cn + T(n-1) \\ &= cn + c(n-1) + T(n-2) \\ &= cn + c(n-1) \end{aligned}$$

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

Let’s solve the following slightly simpler recurrence:

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

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

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

$$\cdots$$

$$= c[n + (n-1) + (n-2) + \cdots \quad (\qquad)] + T(n-k-1)$$

$$= c[n + (n-1) + (n-2) + \cdots + 2] + T(1), \quad when\ k = n-2$$

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

Let's solve the following slightly simpler recurrence:

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

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

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

$$\cdots$$

$$= c[n + (n-1) + (n-2) + \cdots \quad (\qquad)] + T(n-k-1)$$

$$= c[n + (n-1) + (n-2) + \cdots + 2] + T(1), \quad when\ k = n-2$$

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

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

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

EXAMPLE 3: TOWER OF HANOI

```
tower(n, start, finish, other) {
    if(n>0) {
        tower(n-                              sh)
        move fro
        tower(n-1, other, fin              t)
    }
}
```

Base case is n=0

Recursive step

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

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

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

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

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

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

$$T(n) = c + 2T(n-1)$$
$$= c + 2\big(c + 2T(n-2)\big)$$
$$= c(1+2) + 4T(n-2)$$
$$= c(1+2) + 4[c + 2T(n-3)]$$
$$= c(1+2+4) + 8T(n-3)$$

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

$$
\begin{aligned}
T(n) &= c + 2T(n-1) \\
&= c + 2\big(c + 2T(n-2)\big) \\
&= c(1+2) + 4T(n-2) \\
&= c(1+2) + 4[c \qquad (\qquad)] \\
&= c(1+2+4) + 8T(n-3) \\
&\ldots \\
&= c[1+2+4+\cdots+2^{k-1}] + 2^k T(n-k) \\
&= c[1+2+4+\cdots+2^{n-1}] + 2^n T(0), \quad when\ k = n
\end{aligned}
$$

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

$$T(n) = c + 2T(n-1)$$
$$= c + 2\big(c + 2T(n-2)\big)$$
$$= c(1+2) + 4T(n-2)$$
$$= c(1+2) + 4[c + 2T(n-3)]$$
$$= c(1+2+4) + 8T(n-3)$$

…

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

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

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

YOU SHOULD KNOW…

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

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

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

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<li
                return binarySearc          y, left, mid-1)
            else
                return binarySearch(list, key, mid+1, right)
        }
    }
    return -1
}
```


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

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

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

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

$$= c +$$

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

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

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

$$= c +$$

$$= c +$$

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

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

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

$$= c +$$

$$= c +$$

$$\dots$$

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

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

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

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

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

$$= c +$$

$$= c +$$

$$\dots$$

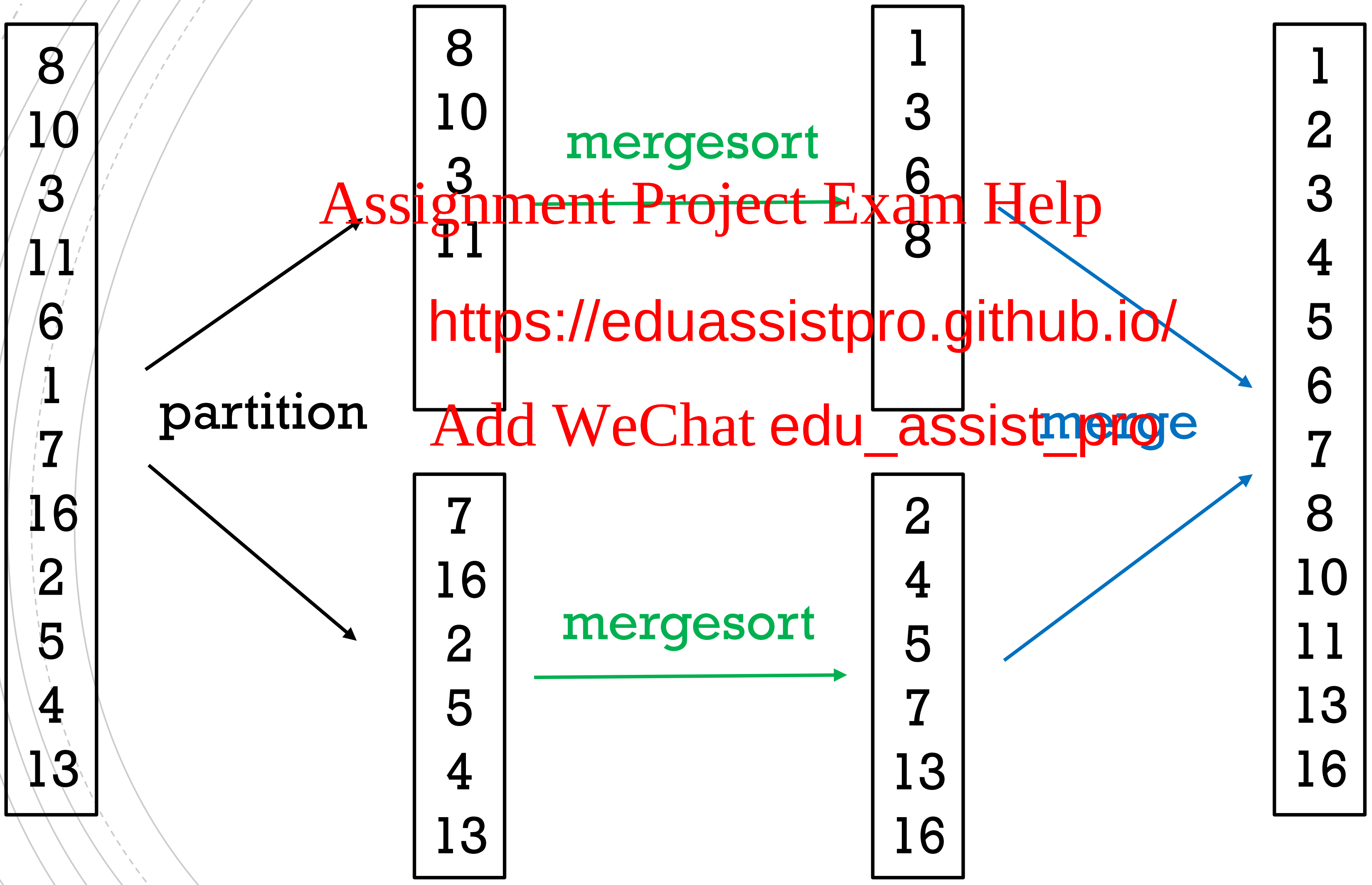
$$= c \cdot k + T(n/2^k)$$

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

$$= c \cdot \log_2 n + b$$

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

RECALL MERGESORT



EXAMPLE 5: MERGESORT

```
mergesort(list){
    if (list.size() == 1)
        return list
    else {
        mid = (list.si
        list1 = list.g
        list2 = list.g                    ()-1)
        list1 = mergesort(list1)
        list2 = mergesort(list2)
        return merge(list1, list2)
    }
}
```

Base case

Recursive step

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

Let's ignore the constant term for simplicity

WHAT IF n IS NOT EVEN?

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

In general, one should wr

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

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

The more general recurrence has roughly the same solution.

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

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

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

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

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

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

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

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

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

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

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

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

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

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

…

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

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

SOLVING THE RECURRENCE USING BACK SUBSTITUTION

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

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

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

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

$$\ldots$$

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

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

$$= cn \log_2 n + bn$$

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

TODAY'S RECURRENCES

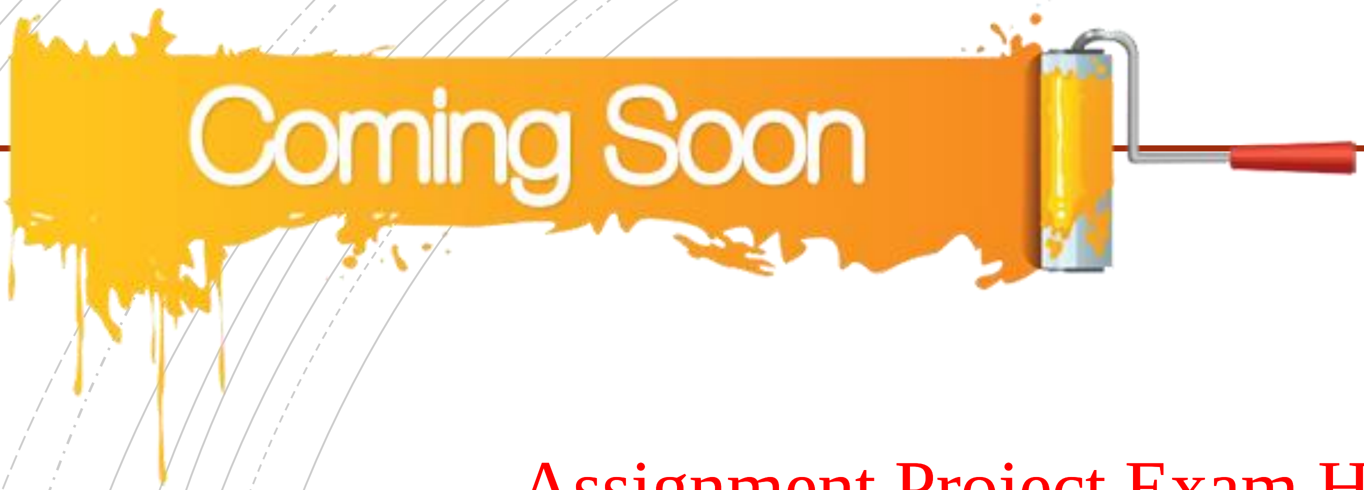
$$T(n) = c + T(n-1)$$

$$T(\ ) \qquad (\qquad)$$

$$T(n) = c + 2 \ (\qquad)$$

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

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



In the next

- Trees