

# lecture 15

## recurrences

- factorial
- Towers of Hanoi
- binary search

### Example 1

```
factorial (n) { // n ≥ 1
  if n == 1
    return 1
  else
    return n * factorial (n-1)
}
```

$$t(1) = 1$$

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

Solving a recurrence using "back substitution"

$$\begin{aligned} t(n) &= c + t(n-1) \\ &= c + c + t(n-2) \\ &= c + c + c + t(n-3) \\ &= (n-1)c + t(1) \end{aligned}$$

Often we write "1" instead of  $c$ , namely one unit.

$$\begin{aligned} t(n) &= 1 + t(n-1) \\ &= 1 + 1 + t(n-2) \\ &= 1 + 1 + 1 + t(n-3) \\ &= n-1 + t(1) \end{aligned}$$

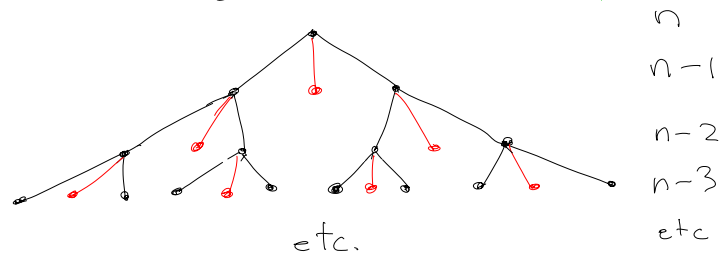
### Example 2 - towers of Hanoi

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

$$t(0) = 0$$

$$t(n) = 1 + 2t(n-1)$$

How many disk moves?



By inspection:

$$1 + 2 + 4 + 8 + \dots + 2^{n-1} = 2^n - 1$$

$$\begin{aligned}
t(n) &= 1 + 2t(n-1) \\
&= 1 + 2(1 + 2t(n-2)) \\
&= 1 + 2 + 4t(n-2) \\
&= 1 + 2 + 4(1 + 2t(n-3)) \\
&= 1 + 2 + 4 + 8t(n-3) \\
&= 1 + 2 + 4 + 8 + 2^{n-1} + 2^n t(0) \\
&= 2^n - 1, \text{ where } t(0) = 0
\end{aligned}$$

### Example 3

$$\begin{aligned}
t(n) &= 1 + c t(n-1) \\
&= 1 + c(1 + c t(n-2)) \\
&= 1 + c + c^2 t(n-2) \\
&= 1 + c + c^2(1 + t(n-3)) \\
&= 1 + c + c^2 + \dots + c^{n-1} + c^n t(0) \\
&= \frac{c^n - 1}{c - 1} + c^n t(0)
\end{aligned}$$

### Careful

$$\begin{aligned}
t(n) &= c + t(n-1) \\
&= cn + t(0) \quad O(n)
\end{aligned}$$

versus

$$\begin{aligned}
t(n) &= 1 + c t(n-1) \\
&= \frac{c^n - 1}{c - 1} + c^n t(0) \quad O(c^n)
\end{aligned}$$

### Example 4

$$\begin{aligned}
t(n) &= n + t(n-1) \\
&= n + (n-1) + t(n-2) \\
&= n + (n-1) + \dots + 2 + 1 + t(0) \\
&= \frac{n(n+1)}{2}
\end{aligned}$$

### Example 5

$$\begin{aligned}
t(n) &= cn + t(n-1) \\
&= cn + c(n-1) + t(n-2) \\
&= c[n + (n-1) + \dots + 2 + 1 + t(0)] \\
&= \frac{cn(n+1)}{2} \quad O(n^2) \\
&\quad \Omega(n^2)
\end{aligned}$$

### Example 6

```

decimalToBinary(n) {
  if (n > 0) {
    print n % 2
    decimalToBinary(n/2)
  }
}

binarySearch(a, e, low, high) {
  if (low < high) {
    mid ← (low + high) / 2
    if (e ≤ a[mid])
      return binarySearch(a, e, low, mid)
    else
      return binarySearch(a, e, mid+1, high)
  }
  else {if e == a[low]
    return low
  else return -1}
}

```

// base case n=0

// base case n=1

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

$$\begin{aligned}
 t(n) &= C + t\left(\frac{n}{2}\right) \\
 &= C + C + t\left(\frac{n}{4}\right) \\
 &= C + C + C + \dots + C + t(1) \\
 &= C \cdot \text{floor}(\log n) + t(1) \\
 &\leq C \cdot \log n + t(1)
 \end{aligned}$$

$t(n)$  is  $O(\log n)$

```

power(x, n) {
  if n == 0
    return 1
  else {
    tmp ← power(x, n/2)
    if n % 2 == 0
      return tmp * tmp
    else
      return tmp * tmp * x
  }
}

```

$C_1$  - one multiplication

$C_2$  - two multiplications

$$t(n) \leq C_2 + t\left(\frac{n}{2}\right) \Rightarrow O(\log n)$$

$$t(n) \geq C_1 + t\left(\frac{n}{2}\right) \Rightarrow \Omega(\log n)$$

Some related stuff  
(which we will use  
in future)

- floor and ceiling
- logs

Notation: floor vs ceiling

Let  $x$  be a real number.

$\lfloor x \rfloor \equiv$  floor ( $x$ ) is the largest integer that is less than or equal to  $x$

$\lceil x \rceil \equiv$  ceil ( $x$ ) is the smallest integer that is greater than or equal to  $x$ .

Let  $n$  be an integer.

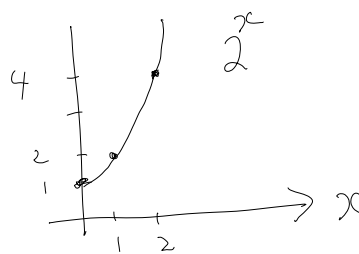
$$n = \lfloor \frac{n}{2.0} \rfloor + \lceil \frac{n}{2.0} \rceil$$

Let  $l$  be such that

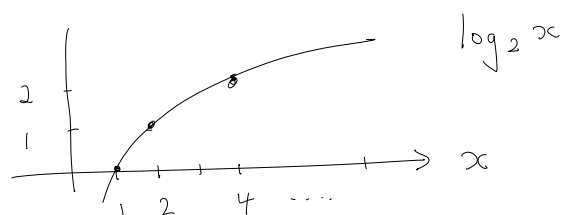
$$2^l \leq n < 2^{l+1}$$

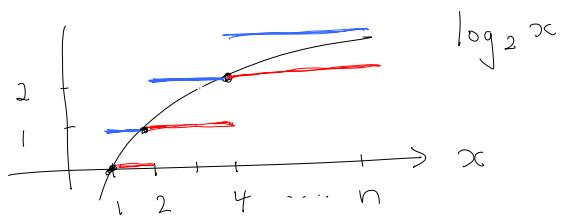
Then,

$$l = \lfloor \log n \rfloor$$



Don't forget:  
one is the inverse of the other!





$\lfloor \log_2 x \rfloor$  is the largest integer that is less than or equal to  $\log_2 x$ .

$\lceil \log_2 x \rceil$  is the smallest integer that is greater than or equal to  $\log_2 x$ .

$2^{\lfloor \log_2 x \rfloor}$

is the largest power of 2 that is less than or equal to  $x$ .

$2^{\lceil \log_2 x \rceil}$

is the smallest power of 2 that is greater than or equal to  $x$ .

- Quiz 1 (10 points)

→  $2 + 2 + 3 + 3$   
                               
     simple      fill in missing  
     problems   code

→ A-L Leacock 2b  
 M-2 here

- "All missing" issue

- Questions? Try WeBC T