CS 310

Parkinson's law:

"Work expands so as to fill the time available for its completion."

Common recurrence relations

INDUCTIVE EQUATION	$T(n)$
$T(n) = T(n-1) + bn^k$	$O(n^{k+1})$
$T(n) = cT(n-1) + bn^k$, for $c > 1$	$O(c^n)$
$T(n) = cT(n/d) + bn^k$, for $c > d^k$	$O(n^{\log_d c})$
$T(n) = cT(n/d) + bn^k$, for $c < d^k$	$O(n^k)$
$T(n) = cT(n/d) + bn^k$, for $c = d^k$	$O(n^k \log n)$

Basis equation is T (1) = a and k ≥ 0.

Finding the closest pair of points

Reference Reading:

Algorithm Design by Tardos et. al. 2006

Chapter 5: **§5.4** Finding the closest pair of points

Computing Fibonacci numbers recursively

$$F_n = F_{n-1} + F_{n-2}, F_0 = 1, F_1 = 1$$

Computing Fibonacci Numbers

$$F_n = F_{n-1} + F_{n-2}, F_0 = 1, F_1 = 1$$

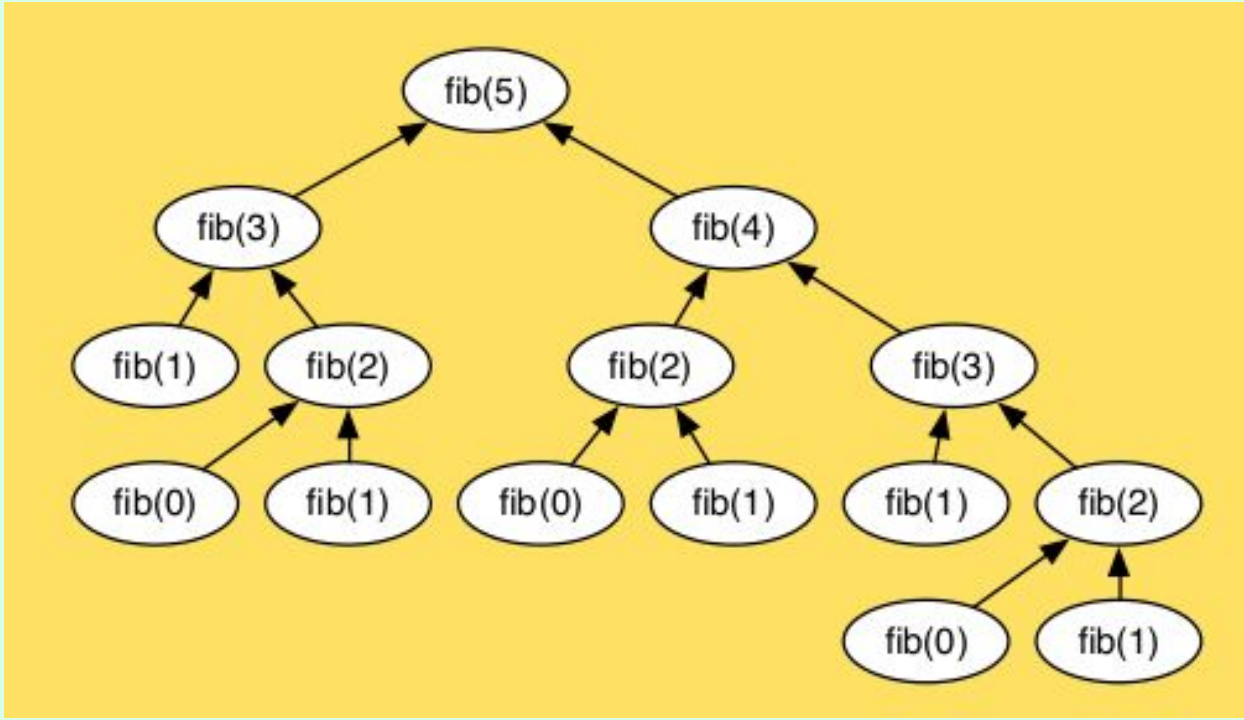
Implementing this as a *recursive* procedure is easy.

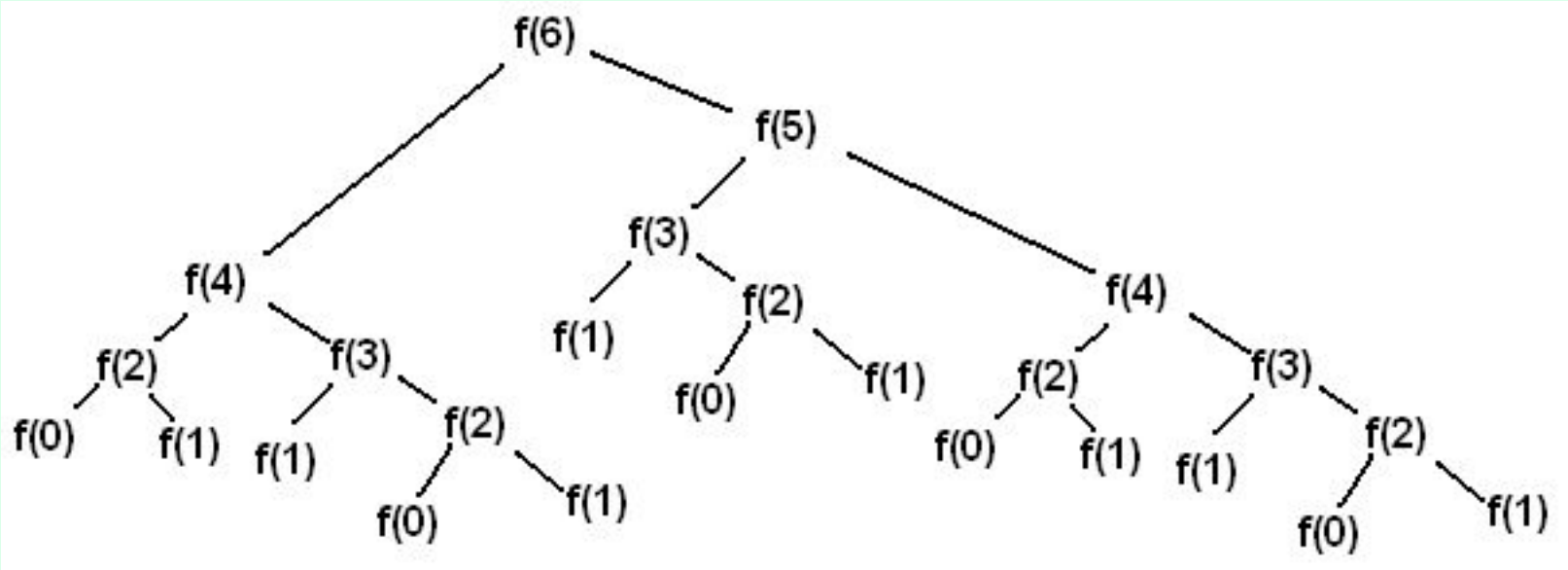
Computing Fibonacci Numbers

$$F_n = F_{n-1} + F_{n-2}, F_0 = 1, F_1 = 1$$

Implementing this as a *recursive* procedure is easy

```
fib(n) =
    if (n ≤ 1) then 1
    else fib(n − 1) + fib(n − 2)
```


fib(5)
fib(3)
fib(4)
fib(1)
fib(2)
fib(2)
fib(3)
fib(0)
fib(1)
fib(0)
fib(1)
fib(1)
fib(2)
fib(0)
fib(1)


f(6)
f(5)
f(4)
f(3)
f(4)
f(2)
f(3)
f(1)
f(2)
f(2)
f(3)
f(0)
f(1)
f(1)
f(2)
f(0)
f(1)
f(0)
f(2)
f(3)
f(1)
f(0)
f(1)
f(1)
f(2)
f(0)
f(1)

Computing Fibonacci Numbers

$$F_n = F_{n-1} + F_{n-2}\,,\; F_0 = 1,\; F_1 = 1$$

Implementing this as a recursive procedure is easy, but ...

Computing Fibonacci Numbers

$$F_n = F_{n-1} + F_{n-2}, F_0 = 1, F_1 = 1$$

Implementing this as a recursive procedure is easy, but slow

Computing Fibonacci Numbers

$$F_n = F_{n-1} + F_{n-2}, F_0 = 1, F_1 = 1$$

Implementing this as a recursive procedure is easy, but slow because we keep calculating the same value over and over.

Memoization

Solution: Trade space for time

Memoization

```
/* initialization */
for j= 1 to n
   Memo[j] = -1
Memo[0] = 1
Memo[1] = 1

fib(n){
   if (Memo[n] < 0)
      Memo[n] = fib(n-1) + fib (n-2)
   return Memo[n];
}
```

Memoization

Solution: Trade space for time

Running time?