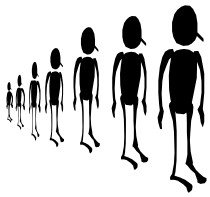
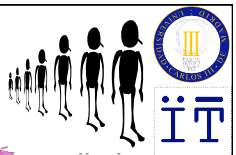
Recursion



Carlos Delgado Kloos
Dep. Ingeniería Telemática
Univ. Carlos III de Madrid

cdk@it.uc3m.es

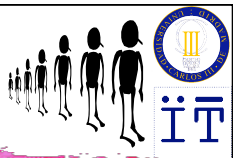
Recursive methods



- A method is called recursive, if it calls itself (directly or indirectly)
- (For a recursive method to define a terminating computation)
  the recursive call(s) have to be simpler (according to some metric)

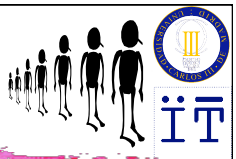
cdk@it.uc3m.es

Example 1



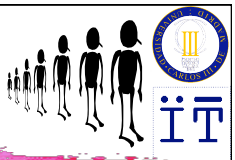
```
public static long s (int n)
{if (n==1)
      return 1;
 else
      return s(n-1)+n;
}
```


Example 1

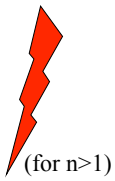


- `s(3) =` (recursive call)
- `s(2)+3 =` (recursive call)
- `(s(1)+2)+3 =` (recursive call)
- `(1+2)+3 =` (sum)
- `(3+3)=` (sum)
- `6`

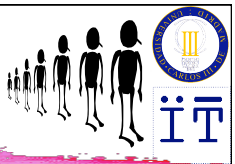
Example 2



```
public static long s (int n)
{if (n==1)
      return 1;
 else
      return s(n+1)+n;
}
```

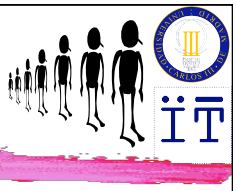


Example 2



- `s(3) =` (rec. call)
- `s(4)+3 =` (rec. call)
- `(s(5)+4)+3 =` (rec. call)
- `((s(6)+5)+4)+3 =` (rec. call)
- `(((s(7)+6)+5)+4)+3 =` (rec. call)
- `((((s(8)+7)+6)+5)+4)+3 =`
- `...`
- Non-termination

How does a recursive method look like?



- Conditional
  - Base case (non-recursive) `1`
  - Recursive case `s(n-1)+n`
    (approximates towards the base case condition) `(n==1)`

Exercise: countBack

```
void countBack (int counter)
{if(counter == 0)
   return;
 else {
   System.out.println(""+counter);
   countBack(--counter);
   return;
   }
}
```

Exercise: square

- $(N-1)^2 = N^2 - 2N + 1$
- $N^2 = (N-1)^2 + 2N - 1$

- square(1) = 1
- square(N) = square(N-1) + 2N -1

Exercise: square

```
int square(int n)
{if (n == 1)
  return 1;
 else
   return square(n-1)+2*n-1;
}
```

Exercise: mystery

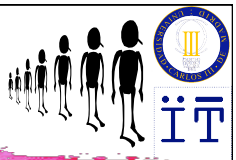
- mystery(0,Q) = Q
- mystery(P,Q) = mystery(P-1, Q+1)

- What is the value of mystery(2,4)?

Kinds of recursion: Linear recursion

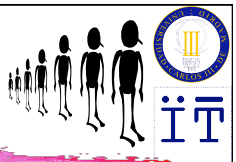
- Linear recursion
  (in each conditional branch at most one recursive call)
  - Tail recursion
    (last operation in branch:
    recursive call)
  - Non-tail recursion
    (pending operation)

Kinds of recursion: Non-linear recursion



- Non-linear recursion
  - Cascading recursion (op(f...,f...))
  - Nested recursion (f(...f...))
  - ...

Example 3: Factorial



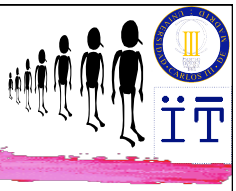
`fac(n) = n!`

`fac(5)= 5*4*3*2*1`

`fac(5)= 5*fac(4)= 5*24=120`

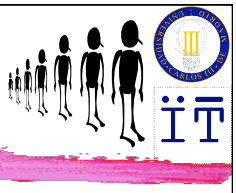
n	fac(n)
0	1
1	1
2	2
3	6
4	24
5	120
...	...

Non-tail recursion: Factorial



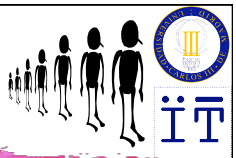
```
public static long fac (int n)
{if (n<=1)
      return 1;
 else
      return n*fac(n-1);
}
```


Tail recursion: Factorial



```
public static long fact (int n,m)
{if (n<=1)
      return m;
 else
      return fact(n-1,n*m);
}
public static long fac (int n)
{return fact(n,1);}
```

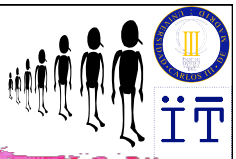
Example 4: Fibonacci



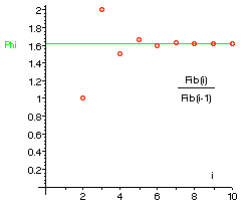
n	fib(n)
0	1
1	1
2	2
3	3
4	5
5	8
...	...

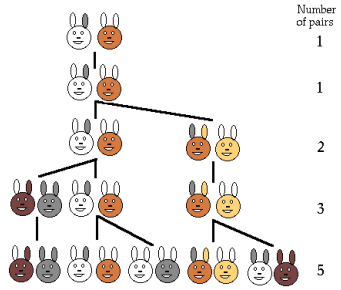
```
fib(5)=
fib(4)+fib(3)
= 5+3
```

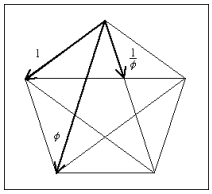

Example 4: Fibonacci

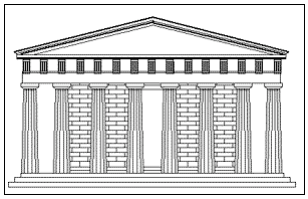


- **Exercise:** Search for applications of *Fibonacci* with *google*

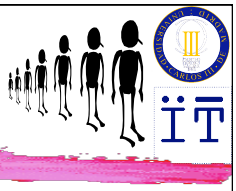






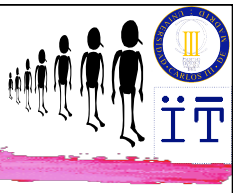


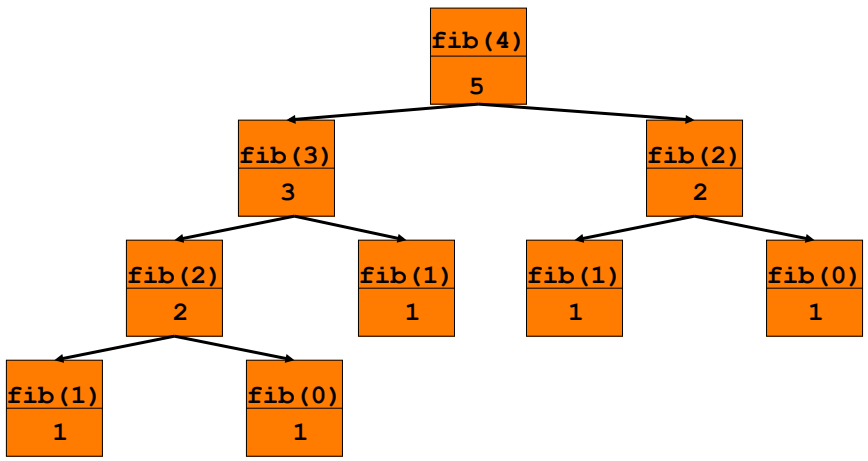
Cascading recursion: Fibonacci



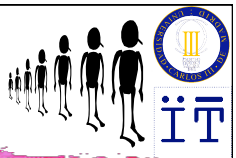
```
public static long fib (int n)
{if (n<=1)
      return 1;
 else
      return fib(n-1)+fib(n-2);
}
```


Fibonacci



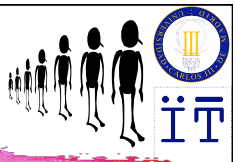


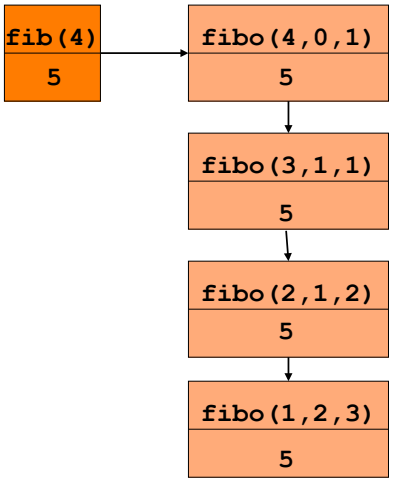
Linear Fibonacci



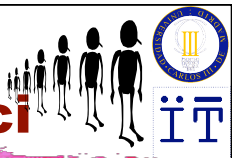
```
public static long fibo (int n,x,y)
{if (n<=1)
      return x+y;
 else
      return fibo(n-1,y,x+y);
}
public static long fib (int n)
{return fibo(n,0,1);}
```


Fibonacci



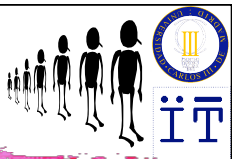


Non-recursive Fibonacci



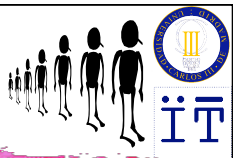
$$fib(n)=\frac{(1+\sqrt{5})^{n+1} - (1-\sqrt{5})^{n+1}}{(2^{n+1}\cdot\sqrt{5})}$$

Nested recursion: Morris



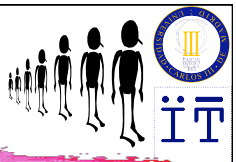
```
public static long mor(int n, m)
{if (n==m)
      return (m+1);
 else
      return mor(n,mor(n-1,m+1));
}
```

Morris



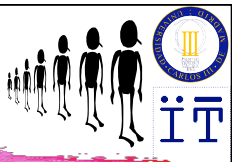
- mor(4,0) =
- mor(4,mor(3,1)) =
- mor(4,mor(3,mor(2,2)) =
- mor(4,mor(3,3)) =
- mor(4,4) =
- 5

Nested recursion: Ackermann

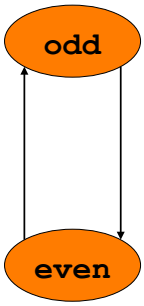


```
public static long ack (int n, m)
{if (n==0)
      return (m+1);
 else if (m==0)
      return ack(n-1,1);
 else
      return ack(n-1,ack(n,m-1));
}
```

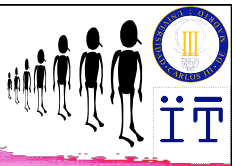
Mutual recursion



```
public static boolean odd (int n)
{if (n==0)
      return false;
 else
      return even(n-1);
}
public static boolean even(int n)
{if (n==0)
      return true;
 else
      return odd(n-1);
}
```



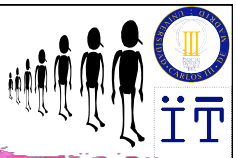
Recursion vs. Iteration



- Tail recursion can be immediately converted into iteration (loop).
- Other forms of recursion require program transformation techniques and possibly more complex data structures.

"The transformation from recursion to iteration is one of the most fundamental concepts of computer science." -- D. Knuth 1974

Factorial



```
public static long
fact (int n,m)
{if (n<=1)
      return m;
 else
      return
      fact(n-1,n*m);
}
```

```
public static long
fact (int n,m)
{private int N,M;
 N=n; M=m;
 while !(N<=1)
   {M=N*M; N=N-1;}
 return M;
}
```


cdk@it.uc3m.es