Programación
Recursividad & Ordenamiento

Dr. Wenceslao Palma
wenceslao.palma@ucv.cl

Recursividad

- La funciones en lenguaje C se pueden emplear recursivamente, es decir, una función puede llamarse a si misma ya sea directa o indirectamente.
- La recursividad es una técnica del tipo "dividir y conquistar" en la cual el problema a resolver se expresa (define) en términos de pequeños problemas (subproblema) del mismo tipo.
- Ejemplo: el factorial de un número entero. Se sabe que:

```
0! = 1
1! = 1
2! = 2x1
3! = 3x2x1
4! = 4x3x2x1
5! = 5x4x3x2x1
.......
```

De este modo podemos definir una función no recursiva de la sgte manera:

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \times (n-1) \times (n-2) \times \cdots \times 3 \times 2 \times 1 & \text{if } n \geq 1 \end{cases}$$

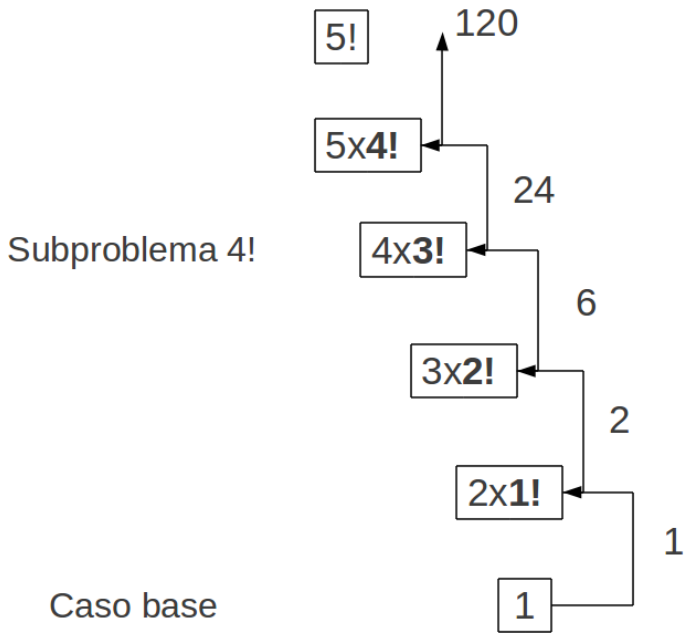
y podemos escribir una función que nos permita calcular $n!$:

```
int factorial(int n){
  int i, fact=1;

  for(i=1;i<=n;i++)
     fact *= i;

   return fact;
}
```

Sin embargo, $n!$ puede ser definido en forma recursiva:

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \times (n-1)! & \text{if } n \geq 1 \end{cases}$$

$n!$ es expresado en función del subproblema $(n-1)!$, y éste último en función del subproblema $(n-2)!$.....



Ahora, podemos escribir una función recursiva en lenguaje C:

```
int factorial(int n){
    int fact;

    if (n==0)
       return 1;

    fact = n*factorial(n-1);
    return fact;
}
```

La recursividad no proporciona un ahorro en almacenamiento ya que se debe mantener una pila de los valores procesados. Ni será más rápida. Pero el código recursivo es más compacto y frecuentemente más fácil de escribir y de entender que su equivalente no recursivo.

Recursividad

En la secuencia de Fibonacci $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...$ cada elemento es la suma de los dos precedentes. Si definimos fib(0)=0, fib(1)=1 y asi sucesivamente, entonces puede definirse la secuencia de Fibonacci en forma recursiva:

$$fib(n) = \begin{cases} n & \text{if } n = 0 || n = 1 \\ fib(n-2) + fib(n-1) & \text{if } n \geq 2 \end{cases}$$

la función en lenguaje C:

```
int fibonacci(int n){
int fib;

if ((n==0) || (n==1))
   return n;

fib = fibonacci(n-2)+fibonacci(n-1);

return fib;
}
```

Proponga un método iterativo para calcular fib(n) y compárelo con el método recursivo usando fib(6).

Ordenamiento

- Insertion sort
- Quicksort

Ordenamiento

Insertion sort:

- va tomando cada uno de los elementos del conjunto de datos a ordenar y los deja (inserta) en la posición correcta.
- En un ordenamiento ascendente, para insertar un elemento k se realiza un corrimiento hacia la derecha de todos los elementos que son mayores o iguales a k.
- Si la secuencia es $1 - 9 - 4 - 8 - 7 - 6 - 12 - 13 - 15 - 11$:

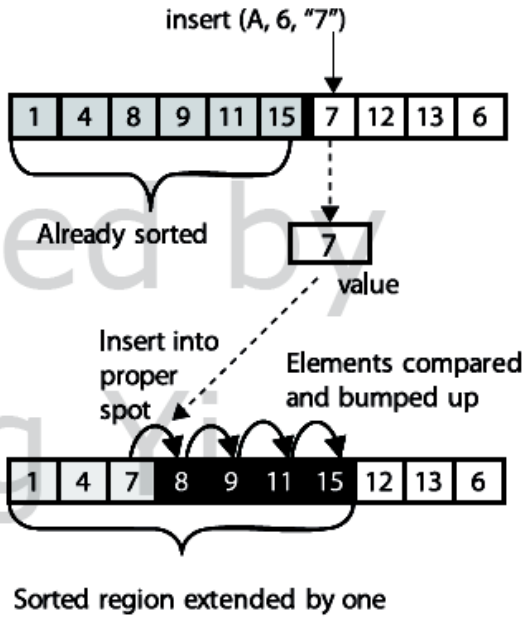
```
1-9-4-8-7-6-12-13-15-11
1-4-9-8-7-6-12-13-15-11
1-4-8-9-7-6-12-13-15-11
1-4-7-8-9-6-12-13-15-11
1-4-6-7-8-9-12-13-15-11
1-4-6-7-8-9-11-12-13-15
```

INSERTION SORT

Best	Average	Worst
$\Omega(n)$	$O(n^2)$	$O(n^2)$

Array

```
sort (A)
1.   for i = 1 to n − 1 do
2.       insert (A, i, A[i])
end

insert (A, pos, value)
1.   i = pos − 1
2.   while (i ≥ 0 and A[i] > value) do
3.         A[i + 1] = A[i]
4.         i = i − 1
5.   A[i + 1] = value
end
```



Otro ejemplo:

15	09	08	01	04	11	07	12	13	06	05	03	16	02	10	14

09	15	08	01	04	11	07	12	13	06	05	03	16	02	10	14
08	09	15	01	04	11	07	12	13	06	05	03	16	02	10	14
01	08	09	15	04	11	07	12	13	06	05	03	16	02	10	14
01	04	08	09	15	11	07	12	13	06	05	03	16	02	10	14
01	04	08	09	11	15	07	12	13	06	05	03	16	02	10	14
01	04	07	08	09	11	15	12	13	06	05	03	16	02	10	14
01	04	07	08	09	11	12	15	13	06	05	03	16	02	10	14
01	04	07	08	09	11	12	13	15	06	05	03	16	02	10	14
01	04	06	07	08	09	11	12	13	15	05	03	16	02	10	14
01	04	05	06	07	08	09	11	12	13	15	03	16	02	10	14
01	03	04	05	06	07	08	09	11	12	13	15	16	02	10	14
01	03	04	05	06	07	08	09	11	12	13	15	16	02	10	14
01	02	03	04	05	06	07	08	09	11	12	13	15	16	10	14
01	02	03	04	05	06	07	08	09	10	11	12	13	15	16	14
01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16

Ordenamiento

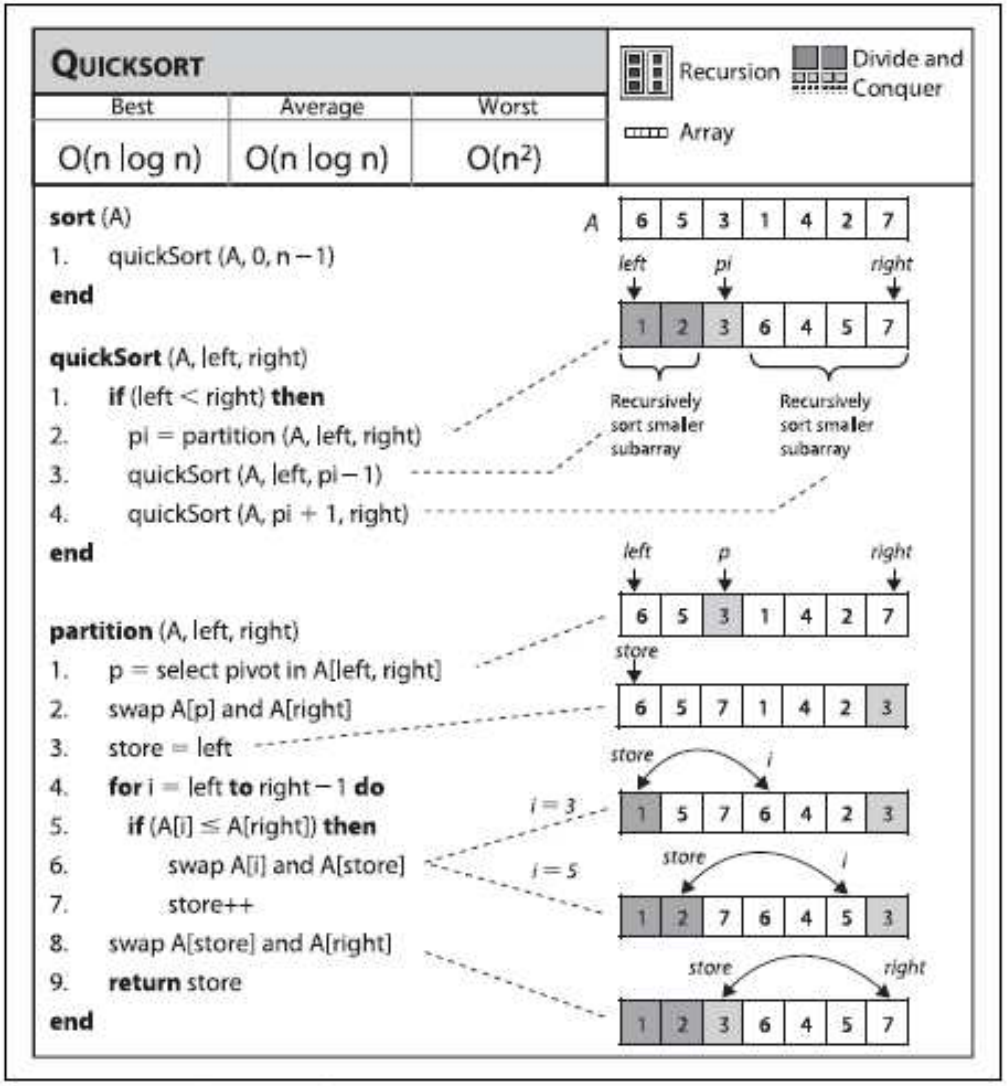
- El mejor caso se presenta cuando el conjunto de datos ya se encuentra ordenado. $O(n)$
- El peor caso se presenta cuando el conjunto de datos se encuentra ordenado en orden inverso. $O(n^2)$

Ordenamiento

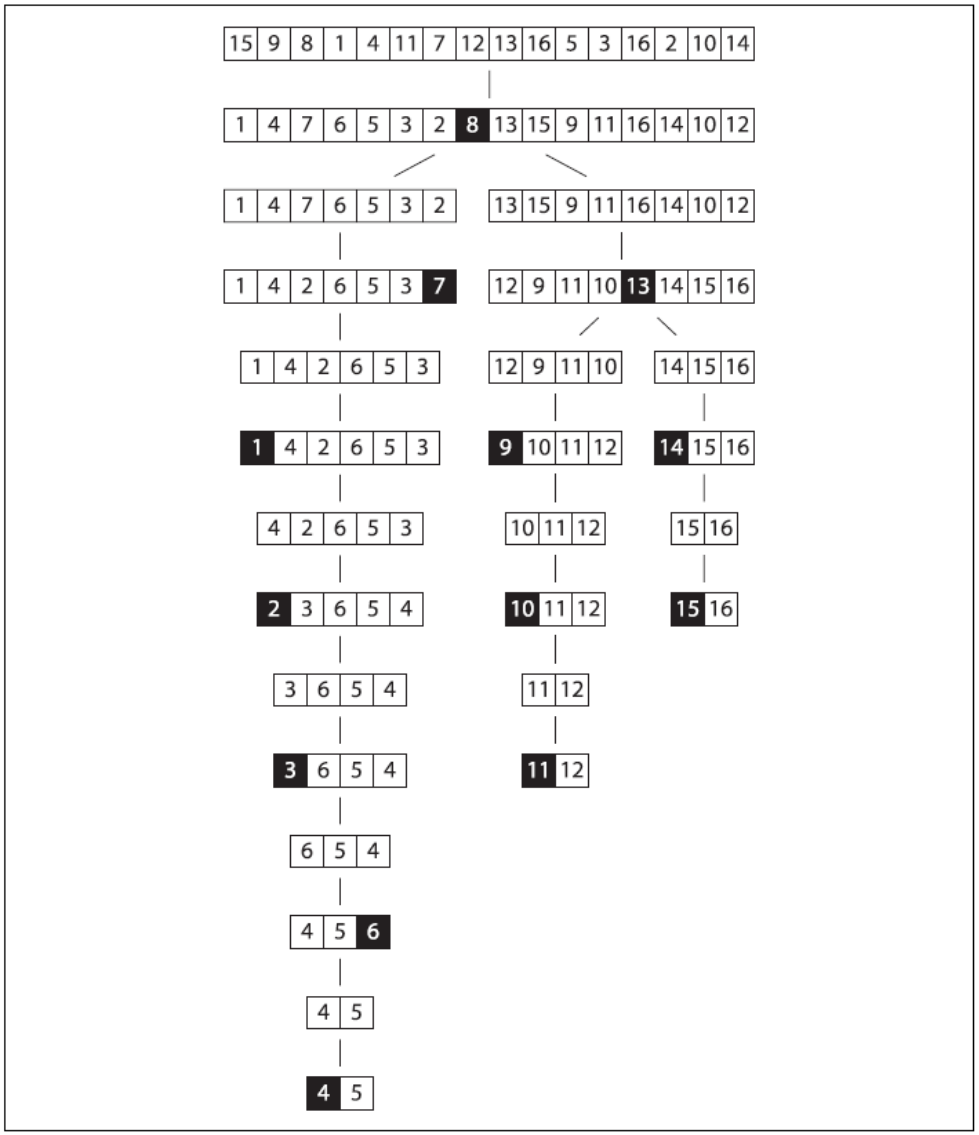
Quicksort:

- utiliza un elemento (pivote) para particionar el conjunto de elementos a ordenar en dos subconjuntos.
- un subconjunto contiene los elementos menores al pivote y el otro contiene los elementos mayores al pivote.
- luego cada subconjunto es ordenado en forma recursiva.

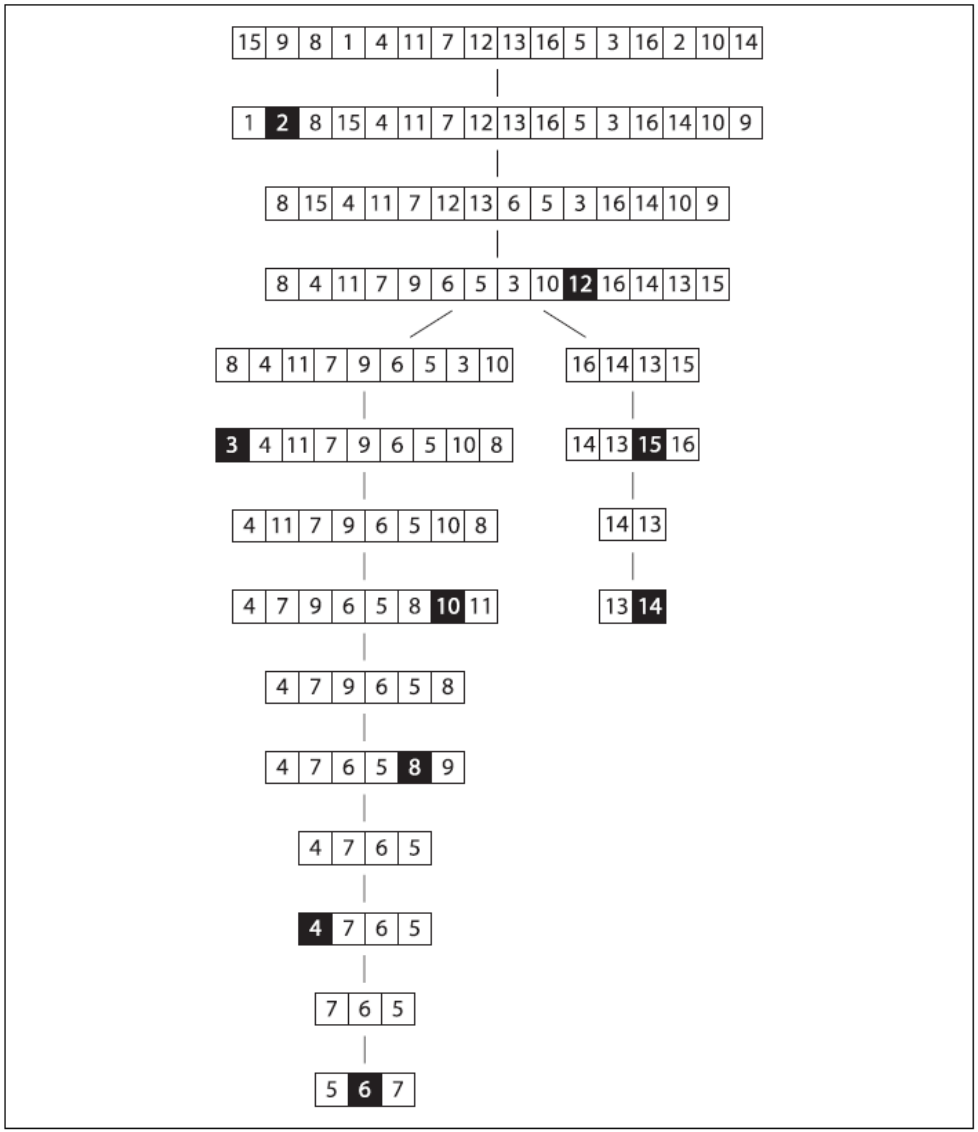
Ordenamiento



Ordenamiento



Ordenamiento



Quicksort en lenguaje C:

```
#include <stdlib.h>
void qsort( void *buf, size_t num, size_t size, int (*cmpInteger)(const void *, const void *) )
```

Un ejemplo:

```
#include <stdlib.h>

main(){
   int a[20]={2,1,0,23,12,45,30,7,10,29,9,5,40,38,33,20,11,60,55,4};

  int i=0;

 qsort(a,20,sizeof(int),(void *)cmpInteger);

 for(i=0;i<20;i++)
     printf("%d ",a[i]);
 printf("\n");
}

int cmpInteger(const int *v1, const int *v2){
   return (*v1 - *v2);
}
```