Análisis de algoritmos recursivos

ANÁLISIS Y DISEÑO DE ALGORITMOS
MTI. PEDRO O. PÉREZ M.
TECNOLÓGICO DE MONTERREY, CAMPUS QUERÉTARO

Algoritmos recursivos

- Este tipo de algoritmos no se pueden analizar como se hace con los algoritmos iterativos.
- El análisis de este tipo de algoritmos se hace utilizando el proceso inductivo:
  - ¿Qué recuerdan del proceso inductivo?
  - ¿Qué pasos hay que considerar?
  - ¿Cómo se realiza?

Ejemplo 1

```
procedure EXAMPLE1(n):
if n == 0 then
    return 1
else
    return 1 + EXAMPLE1(n - 1)
```

Ejemplo 2

```
procedure EXAMPLE2(n):
if n == 0 then
    return 1
else
    return EXAMPLE2(n - 1) + EXAMPLE2(n - 1)
```

Ejemplo 3

```
procedure EXAMPLE3(n):
if n == 1 then
    return 1
else
    return 1 + EXAMPLE3(n / 2)
```

Ejemplo 4

```
procedure EXAMPLE4(n):
if n == 1 then
    return 1
else
    return 1 + EXAMPLE4(n / 3)
```

Quick Sort

```
procedure FIND_PIVOT(A, low, high):
    if low == high then
        return -1
    else
        return A[low]
```

Quick Sort

```
procedure MAKE_PARTITION(A, low, high, pivot):
i ← low
j ← high

while i < j do
begin
    SWAP(A, i, j)

    while A[i] < pivot do
        i ← i + 1

    while A[j] >= pivot do
        j ← j - 1
end
return i
```

$$\sum_{1}^{n} c = n * c$$

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

$$\sum_{1}^{n} i^2 = \frac{2n^3+3n^2+n}{6}$$

Quick Sort

```
procedure QUICKSORT(A, low, high):
pivot ← FIND_PIVOT(A, low, high)

if pivot <> -1 then
begin
    pos ← MAKE_PARTITION(A, low, high, pivot)
    QUICKSORT(A, low, pos - 1)
    QUICKSORT(A, pos, high)
end
```

Búsqueda binaria

```
procedure BINARY_SEARCH(A, low, high, key):
if low <= high then
begin
    mid ← (high + low) / 2
    if key == A[mid] then
        return mid
    else if key < A[mid] then
        BINARY_SEARCH(A, low, mid − 1, key)
    else
        BINARY_SEARCH(A, mid + 1, high, key)
end
return -1
```

Ejercicios colaborativos

```
procedure TERNARY_SEARCH(A, low, high, key):
if low <= high then
begin
    aux ← high + low
    mid1 ← aux / 3
    mid2 ← (2 * aux) / 3
    if key == A[mid1] then
        return mid1
    else if key == A[mid2] then
        return mid2
    else if key < A[mid1] then
        TERNARY_SEARCH(A, low, mid1 - 1, key)
    else if key > A[mid1] and key < A[mid2] then
        TERNARY_SEARCH(A, mid1 + 1, mid2 - 1, key)
    else
        TERNARY_SEARCH(A, mid2 + 1, high, key)
end
return -1
```