Análisis de algoritmos recursivos

Pedro O. Pérez M., MTI

Análisis y diseño de algoritmos
Tecnológico de Monterrey

pperezm@tec.mx

01-2019

Contenido

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 utilizan el proceso inductivo.
- ¿Qué recuerdas del proceso inductivo? ¿Qué pasos hay que considerar? ¿Cómo se realiza?

Algoritmos conocidos

Procedure 1 POW - Calculate x^n using a recursive algorithm.

Input: $x : Real, n : Integer$
  if $n = 0$ **then**
    return 1
  else
    return $x * POW(x, n - 1)$
  end if

Procedure 2 ENIGMA - Unknown

Input: n : *Integer*

```
if n <= 0 then
    return 1
else
    return ENIGMA(n − 1) + ENIGMA(n − 1)
end if
```

Procedure 3 BINARY_SEARCH - Find the position of an element within an array using a recursive algorithm.

Input: $A : Array, low : Index, high : Index, key : Object$

```
if (high − low) = 1 then
  if A[low] = key then
    return low
  else
    return −1
  end if
else
  NEXTSLIDE
end if
return −1
```

$mid \leftarrow (high + low)/2$
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 if

Procedure 4 SEARCH_BST - Find the position of an element within a binary search tree.

Input: $node : BSTNode, key : Object$

```
if value = node → value then
    return true
else if value < node → value then
    return SEARCH_BST(node → left)
else if value > node → value then
    return SEARCH_BST(node → right)
end if
```

Ejercicios de revisión I

Procedure 5 POW2 - $y = x^n$, using a recursive algorithm

Input: $x : Real, n : Integer$

```
if n < 0 then
    return POW2(1/x, −n)
else if n == 0 then
    return 1
else if n == 1 then
    return x
else if n mód 2 = 0 then
    return POW2(x ∗ x, n/2)
else if n mód 2 = 1 then
    return x ∗ POW2(x ∗ x, (n − 1)/2)
end if
```

Procedure 6 SUM_LIST - Calculate the sum of all the elements in a list of integers.

Input: n : *node_list*

```
if n <> NIL then
    return 0
else
    return n → value + SUM_LIST(n → next)
end if
```

Merge Sort

Procedure 7 COPY

Input: $A, B : Array; low, high : Index$

for $i \leftarrow low$ **to** $high$ **do**

$A[i] \leftarrow B[i]$

end for

Procedure 8 MERGE

Input: $A, B : Array; low, mid, high : Index$

```
i ← low
j ← mid + 1
k ← low
while i ≤ mid and j ≤ high do
    if A[i] < A[j] then
        B[k] = A[i]
        i ← i + 1
    else if A[i] ≥ A[j] then
        B[k] = A[j]
        j ← j + 1
    end if
end while
```

```
while j ≤ high do
    B[k] = A[j]
    k ← k + 1
    j ← j + 1
end while
while i ≤ mid do
    B[k] = A[i]
    k ← k + 1
    i ← i + 1
end while
```

Procedure 9 SPLIT

Input: $A, B : Array; low, high : Index$

if $(high - low) \geq 1$ **then**
  $mid \leftarrow (high + low)/2$
  $SPLIT(A, B, low, mid)$
  $SPLIT(A, B, mid + 1, high)$
  $MERGE(A, B, low, mid, high)$
  $COPY(A, B, low, high)$
end if