Introducción a la Computación (Matemática)

Primer Cuatrimestre de 2018

Recursión Algorítmica

Recursión algorítmica

Es uno de los conceptos centrales en Computación.

La solución a un problema depende de la solución a instancias de menor tamaño del mismo problema.

Recursión algorítmica: ejemplo (factorial)

Encabezado: *Fact* $: n \in \mathbb{Z} \rightarrow \mathbb{Z}$
Precondición: $\{n = n_0 \wedge n_0 \geq 0\}$
Poscondición: $\{RV = n_0!\}$

Recursión algorítmica: ejemplo (factorial)

Encabezado: *Fact* : $n \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{n = n_0 \wedge n_0 \geq 0\}$
Poscondición: $\{RV = n_0!\}$

// Algoritmo iterativo
$RV \leftarrow 1$
while $(n > 0)$ {
    $RV \leftarrow RV * n$
    $n \leftarrow n - 1$
}

Recursión algorítmica: ejemplo (factorial)

Encabezado: *Fact* $: n \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{n = n_0 \wedge n_0 \geq 0\}$
Poscondición: $\{RV = n_0!\}$

```
// Algoritmo iterativo
RV ← 1
while (n > 0) {
        RV ← RV * n
        n ← n − 1
}
```

```
// Algoritmo recursivo
if (n = 0) {
   RV ← 1
} else {
   RV ← Fact(n − 1) * n
}
```

Recursión algorítmica: ejemplo (producto)

Encabezado: *Prod* $: n \in \mathbb{Z} \times m \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{n = n_0 \wedge n_0 \geq 0 \wedge m = m_0 \wedge m_0 \geq 0\}$
Poscondición: $\{RV = n_0 * m_0\}$

Recursión algorítmica: ejemplo (producto)

Encabezado: *Prod* $: n \in \mathbb{Z} \times m \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{n = n_0 \wedge n_0 \geq 0 \wedge m = m_0 \wedge m_0 \geq 0\}$
Poscondición: $\{RV = n_0 * m_0\}$

// Algoritmo iterativo
$RV \leftarrow 0$
while $(m > 0)$ {
    $RV \leftarrow RV + n$
    $m \leftarrow m - 1$
}

Recursión algorítmica: ejemplo (producto)

Encabezado: *Prod* $: n \in \mathbb{Z} \times m \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{n = n_0 \wedge n_0 \geq 0 \wedge m = m_0 \wedge m_0 \geq 0\}$
Poscondición: $\{RV = n_0 * m_0\}$

// Algoritmo iterativo
$RV \leftarrow 0$
while $(m > 0)$ {
    $RV \leftarrow RV + n$
    $m \leftarrow m - 1$
}

// Algoritmo recursivo
if $(m = 0)$ {
    $RV \leftarrow 0$
} else {
    $RV \leftarrow Prod(n, m-1) + n$
}

Recursión algorítmica

1. Resolver el problema para los casos base.
2. Suponiendo que se tiene resuelto el problema para instancias de menor tamaño, modificar dichas soluciones para obtener una solución al problema original.

La recursión ofrece otra forma de ciclar o repetir código.

Recursión algorítmica

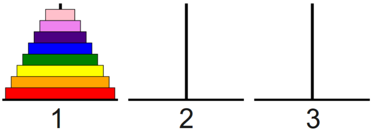
Herramienta poderosa para encontrar algoritmos para problemas no triviales, mediante técnicas como Divide and conquer o Backtracking.

Recursión algorítmica

Herramienta poderosa para encontrar algoritmos para problemas no triviales, mediante técnicas como Divide and conquer o Backtracking.

Por ej., ¿se acuerdan del problema de las Torre de Hanoi?

- Mover N discos de la estaca 1 a la 3.
- Mover de a un disco por vez.
- No se puede colocar un disco sobre otro de menor tamaño.



Hoy vamos a ver cómo resolverlo usando D&C.

Divide and conquer. 1

- Táctica político-militar de dudoso origen, frecuentemente atribuida a Julio César.
- Consiste en dividir al enemigo, de modo que cada una de las partes sea más fácil de derrotar que el todo.

Divide and conquer. 1

- Táctica político-militar de dudoso origen, frecuentemente atribuida a Julio César.
- Consiste en dividir al enemigo, de modo que cada una de las partes sea más fácil de derrotar que el todo.

En Computación, la técnica de D&C tiene tres etapas:

1. **Divide:** Dividir el problema en varios subproblemas de menor tamaño.

Divide and conquer. 1

- Táctica político-militar de dudoso origen, frecuentemente atribuida a Julio César.
- Consiste en dividir al enemigo, de modo que cada una de las partes sea más fácil de derrotar que el todo.

En Computación, la técnica de D&C tiene tres etapas:

1. **Divide:** Dividir el problema en varios subproblemas de menor tamaño.
2. **Conquer:** Resolver cada subproblema recursivamente. Si un subproblema es lo suficientemente pequeño (un caso base), resolverlo en forma directa.

Divide and conquer. 1

- Táctica político-militar de dudoso origen, frecuentemente atribuida a Julio César.
- Consiste en dividir al enemigo, de modo que cada una de las partes sea más fácil de derrotar que el todo.

En Computación, la técnica de D&C tiene tres etapas:

1. **Divide:** Dividir el problema en varios subproblemas de menor tamaño.
2. **Conquer:** Resolver cada subproblema recursivamente. Si un subproblema es lo suficientemente pequeño (un caso base), resolverlo en forma directa.
3. **Combine:** Combinar las soluciones de los subproblemas en una solución del problema original.

Divide and conquer. 2

Requisitos para aplicar divide y vencerás:

- Necesitamos un método (más o menos directo) de resolver los problemas de tamaño pequeño.
- El problema original debe poder dividirse fácilmente en un conjunto de subproblemas, del mismo tipo que el problema original pero con una resolución más sencilla (menos costosa).
- Los subproblemas deben ser disjuntos: la solución de un subproblema debe obtenerse independientemente de los otros.
- Es necesario tener un método de combinar los resultados de los subproblemas. En general que no sea muy oneroso.

Ejemplo 1 de D&C: Fibonacci de **F(n)**

Divide & Conquer: Dividir el problema en **F(n-1)** y **F(n-2)**.
Casos base: **F(1)= F(0)=1**.

Ejemplo 1 de D&C: Fibonacci de **F(n)**

Divide & Conquer: Dividir el problema en **F(n-1)** y **F(n-2)**.
Casos base: **F(1)= F(0)=1**.

Combine: Sumar las soluciones: **F(n)= F(n-1) + F(n-2)** .

Ejemplo 2 de D&C: Mergesort

Problema: Ordenar un arreglo A de enteros de tamaño n.

1. **Divide:**

Ejemplo 2 de D&C: Mergesort

Problema: Ordenar un arreglo A de enteros de tamaño n.

1. **Divide:** Dividir A en 2 subarreglos de tamaño $\sim \frac{n}{2}$.
2. **Conquer:**

Ejemplo 2 de D&C: Mergesort

Problema: Ordenar un arreglo A de enteros de tamaño n.

1. **Divide:** Dividir A en 2 subarreglos de tamaño $\sim \frac{n}{2}$.
2. **Conquer:** Ordenar cada subarreglo recursivamente.
   Si un subarreglo tiene tamaño 1 (caso base), no hacer nada.
3. **Combine:**

Ejemplo 2 de D&C: Mergesort

Problema: Ordenar un arreglo A de enteros de tamaño n.

1. **Divide:** Dividir A en 2 subarreglos de tamaño $\sim \frac{n}{2}$.
2. **Conquer:** Ordenar cada subarreglo recursivamente.
   Si un subarreglo tiene tamaño 1 (caso base), no hacer nada.
3. **Combine:** Combinar los 2 subarreglos ordenados (función *merge*).

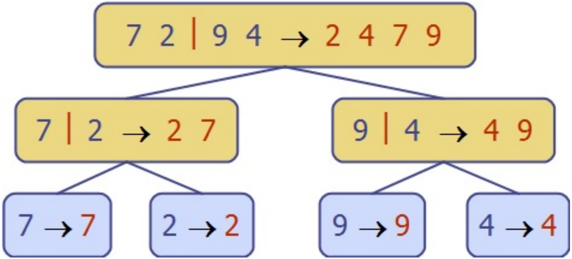
Ejemplo 2 de D&C: Mergesort

Problema: Ordenar un arreglo A de enteros de tamaño n.

1. **Divide:** Dividir A en 2 subarreglos de tamaño $\sim \frac{n}{2}$.
2. **Conquer:** Ordenar cada subarreglo recursivamente.
   Si un subarreglo tiene tamaño 1 (caso base), no hacer nada.
3. **Combine:** Combinar los 2 subarreglos ordenados (función *merge*).

Si *merge* tiene orden lineal, entonces *mergesort* tiene $O(n \log n)$. (Demostración: Clase que viene.)

Ejemplo 3 de D&C: Búsqueda Binaria



Ejemplo 3 de D&C: Búsqueda Binaria

4	7	23	41	44	59	97	134	165	187	210	212	249	280	314
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Buscamos el número **97**...

4	7	23	41	44	59	97	134
0	1	2	3	4	5	6	7

165	187	210	212	249	280	314
8	9	10	11	12	13	14

Ejemplo 3 de D&C: Búsqueda Binaria

4	7	23	41	44	59	97	134	165	187	210	212	249	280	314
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Buscamos el número **97**...

4	7	23	41	44	59	97	134
0	1	2	3	4	5	6	7

165	187	210	212	249	280	314
8	9	10	11	12	13	14

4	7	23	41
0	1	2	3

44	59	97	134
4	5	6	7

44	59
4	5

97	134
6	7

97
6

134
7

97 ✓
6

Ejemplo 3 de D&C: Búsqueda Binaria Recursiva

Buscar $: x \in \mathbb{Z} \times A \in \mathbb{Z}[\,] \rightarrow está \in \mathbb{B} \times pos \in \mathbb{Z}$

Ejemplo 3 de D&C: Búsqueda Binaria Recursiva

Buscar : $x \in \mathbb{Z} \times A \in \mathbb{Z}[\,] \rightarrow está \in \mathbb{B} \times pos \in \mathbb{Z}$

$$(está, pos) \leftarrow BuscarDesdeHasta(x, A, 0, |A| - 1)$$

BuscarDesdeHasta : $x \in \mathbb{Z} \times A \in \mathbb{Z}[\,] \times izq \in \mathbb{Z} \times der \in \mathbb{Z}$
$\rightarrow está \in \mathbb{B} \times pos \in \mathbb{Z}$

Ejemplo 3 de D&C: Búsqueda Binaria Recursiva

Buscar : $x \in \mathbb{Z} \times A \in \mathbb{Z}[\,] \to \textit{está} \in \mathbb{B} \times pos \in \mathbb{Z}$

$$(\textit{está}, pos) \leftarrow BuscarDesdeHasta(x, A, 0, |A| - 1)$$

BuscarDesdeHasta : $x \in \mathbb{Z} \times A \in \mathbb{Z}[\,] \times izq \in \mathbb{Z} \times der \in \mathbb{Z}$
$\to \textit{está} \in \mathbb{B} \times pos \in \mathbb{Z}$

Ejemplo 3 de D&C: Búsqueda Binaria Recursiva

Buscar : $x \in \mathbb{Z} \times A \in \mathbb{Z}[\,] \to \textit{está} \in \mathbb{B} \times pos \in \mathbb{Z}$

$$(\textit{está}, pos) \leftarrow BuscarDesdeHasta(x, A, 0, |A| - 1)$$

BuscarDesdeHasta : $x \in \mathbb{Z} \times A \in \mathbb{Z}[\,] \times izq \in \mathbb{Z} \times der \in \mathbb{Z}$
$\to \textit{está} \in \mathbb{B} \times pos \in \mathbb{Z}$

$med \leftarrow (izq + der)$ div 2
if $(A[med] < x)$ {

$(\textit{está}, pos) \leftarrow BuscarDesdeHasta(x, A, med + 1, der)$
} else {
$(\textit{está}, pos) \leftarrow BuscarDesdeHasta(x, A, izq, med)$
}

Ejemplo 3 de D&C: Búsqueda Binaria Recursiva

Buscar : $x \in \mathbb{Z} \times A \in \mathbb{Z}[\,] \to está \in \mathbb{B} \times pos \in \mathbb{Z}$

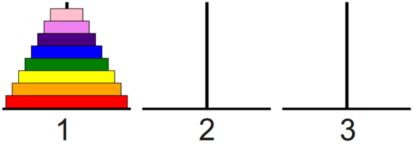
$$(está, pos) \leftarrow BuscarDesdeHasta(x, A, 0, |A| - 1)$$

BuscarDesdeHasta : $x \in \mathbb{Z} \times A \in \mathbb{Z}[\,] \times izq \in \mathbb{Z} \times der \in \mathbb{Z}$
$\to está \in \mathbb{B} \times pos \in \mathbb{Z}$

if $(izq = der)$ {
  $(está, pos) \leftarrow (x = A[izq],\ izq)$
} else {
  $med \leftarrow (izq + der)$ div 2
  if $(A[med] < x)$ {

$(está, pos) \leftarrow BuscarDesdeHasta(x, A, med + 1, der)$
  } else {
    $(está, pos) \leftarrow BuscarDesdeHasta(x, A, izq, med)$
  }
}

Ejemplo 4 de D&C: Torre de Hanoi



Objetivo: Mover N discos de la estaca 1 a la 3.

Restricciones:

- Mover de a un disco por vez.
- No se puede poner un disco sobre otro de menor tamaño.

Demo: http://www.uterra.com/juegos/torre_hanoi.php

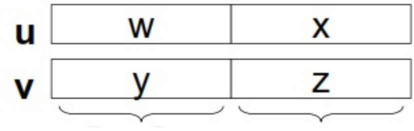
Ejemplo de D&C: Torre de Hanoi

Hanoi(n, $desde$, $hacia$, $otra$):
    if $(n > 1)$:
        Hanoi($n-1$, $desde$, $otra$, $hacia$)
        Mover el disco superior de $desde$ a $hacia$.
        Hanoi($n-1$, $otra$, $hacia$, $desde$)
    else:
        Mover el disco superior de $desde$ a $hacia$.

Por ejemplo, el llamado para resolver Hanoi de 8 discos es: Hanoi(8, 1, 3, 2).

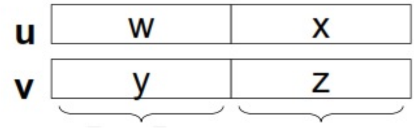
Multiplicación rápida de enteros largos. 1

Sean **u** y **v** dos números enteros de **n** dígitos.



Multiplicación rápida de enteros largos. 1

Sean **u** y **v** dos números enteros de **n** dígitos.



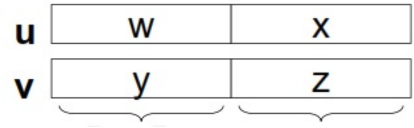
Tomo $\mathbf{S} = \mathbf{n}/2$

$$\mathbf{u} = \mathbf{w} * 10^{\mathbf{S}} + \mathbf{x}$$

$$\mathbf{v} = \mathbf{y} * 10^{\mathbf{S}} + \mathbf{z}$$

Multiplicación rápida de enteros largos. 1

Sean **u** y **v** dos números enteros de **n** dígitos.



Tomo $\mathbf{S} = \mathbf{n/2}$

$\mathbf{u} = \mathbf{w} * 10^{\mathbf{S}} + \mathbf{x}$

$\mathbf{v} = \mathbf{y} * 10^{\mathbf{S}} + \mathbf{z}$

Calculo la multiplicación con D&C usando:

$\mathbf{u*v} = \mathbf{w*y} * 10^{\mathbf{2S}} + (\mathbf{w*z} + \mathbf{x*y}) * 10^{\mathbf{S}} + \mathbf{x*z}$

Multiplicación rápida de enteros largos. 2

- El problema de tamaño **n** es descompuesto en 4 problemas de tamaño **n/2**.
- La suma se puede realizar en un tiempo lineal O(n).
- ¿Cuánto es el tiempo de ejecución?

Multiplicación rápida de enteros largos. 2

- El problema de tamaño **n** es descompuesto en 4 problemas de tamaño **n/2**.
- La suma se puede realizar en un tiempo lineal O(n).
- ¿Cuánto es el tiempo de ejecución? Próxima clase $O(n^2)$.

Multiplicación rápida de enteros largos. 2

- El problema de tamaño **n** es descompuesto en 4 problemas de tamaño **n/2**.
- La suma se puede realizar en un tiempo lineal O(n).
- ¿Cuánto es el tiempo de ejecución? Próxima clase $O(n^2)$.

¿Podemos mejorarlo?

Multiplicación rápida de enteros largos. 2

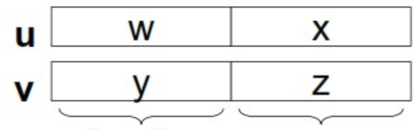
- El problema de tamaño **n** es descompuesto en 4 problemas de tamaño **n/2**.
- La suma se puede realizar en un tiempo lineal O(n).
- ¿Cuánto es el tiempo de ejecución? Próxima clase $O(n^2)$.

¿Podemos mejorarlo?

¿Y si en vez de 4 tuviéramos 3 subproblemas...?

Multiplicación rápida de enteros largos. 3

Método de Karatsuba y Ofman:



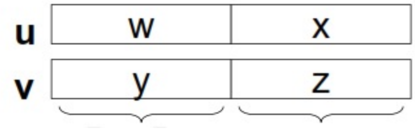
Tomo $\mathbf{S = n/2}$

$$\mathbf{u} = \mathbf{w} * 10^{\mathbf{S}} + \mathbf{x}$$

$$\mathbf{v} = \mathbf{y} * 10^{\mathbf{S}} + \mathbf{z}$$

Multiplicación rápida de enteros largos. 3

Método de Karatsuba y Ofman:



Tomo $\mathbf{S} = \mathbf{n/2}$

$$\mathbf{u} = \mathbf{w} * 10^{\mathbf{S}} + \mathbf{x}$$

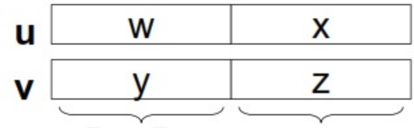
$$\mathbf{v} = \mathbf{y} * 10^{\mathbf{S}} + \mathbf{z}$$

Calculo la multiplicación con D&C usando:

$$\mathbf{u*v} = \mathbf{w*y} * 10^{\mathbf{2S}} + \mathbf{[(w\text{-}x)*(z\text{-}y) + w*y + x*z]} * 10^{\mathbf{S}} + \mathbf{x*z}$$

Multiplicación rápida de enteros largos. 3

Método de Karatsuba y Ofman:



Tomo $\mathbf{S = n/2}$
$\mathbf{u} = \mathbf{w} * 10^{\mathbf{S}} + \mathbf{x}$
$\mathbf{v} = \mathbf{y} * 10^{\mathbf{S}} + \mathbf{z}$

Calculo la multiplicación con D&C usando:
$\mathbf{u*v} = \mathbf{w*y} * 10^{\mathbf{2S}} + \mathbf{[(w\text{-}x)*(z\text{-}y) + w*y + x*z]} * 10^{\mathbf{S}} + \mathbf{x*z}$
Subproblemas:

P1$=$ **w*y**
P2$=$ **(w-x)*(z-y)**
P3$=$ **x*z**

Ejemplo de D&C: Par más cercano

Dados n puntos (x_i, y_i) en el plano, encontrar el par de puntos más cercanos entre sí (considerar la distancia euclideana).

1. Ordenar los puntos según su coordenada X.
2. Si el tamaño del conjunto es 2, devolver la distancia entre ellos. Si el conjunto tiene 0 o 1 elementos, devolver infinito.
3. Dividir el conjunto de puntos en dos partes iguales (del mismo número de puntos).
4. Solucionar el problema de forma recursiva en las partes izquierdas y derecha. Esto devolverá una solución para cada parte, llamadas dLmin y dRmin. Escoger el mínimo entre estas dos soluciones, llamado dLRmin.
5. Seleccionar los puntos de la parte derecha e izquierda que están a una distancia horizontal menor que dLRmin de la recta divisoria entre ambos. Aprovechar que los puntos están ordenados para elegir los últimos puntos de la parte izquierda y los primeros de la parte derecha.
6. Encontrar la distancia mínima dCmin entre todos los pares de puntos formados por un punto de cada parte del

Repaso de la clase de hoy

- Recursión algorítmica.
- Producto, factorial.
- Divide & Conquer.
- Fibonacci, Mergesort, Hanoi, Búsqueda binaria, Multiplicación de enteros grandes y par más cercano.

Próximos temas

- Complejidad algorítmica.
- Consumo de memoria de la recursión.