Curso de nivelación de algoritmos

Recursión Algorítmica

Recursión

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 en Álgebra

Definición de naturales (con cero):

- $0 \in \mathbb{N}$
- Si $n \in \mathbb{N}$ entonces $\text{suc}(n) \in \mathbb{N}$

Recursión en Álgebra

Definición de naturales (con cero):

- $0 \in \mathbb{N}$
- Si $n \in \mathbb{N}$ entonces $\text{suc}(n) \in \mathbb{N}$

Para $n, m \in \mathbb{N}$:

- $n + 0 = n$
- $n + \text{suc}(m) = \text{suc}(n + m)$
- $n * 0 = 0$
- $n * \text{suc}(m) = n * m + n$
- $0! = suc(0)$
- $\text{suc}(n)! = \text{suc}(n) * n!$

Recursión algorítmica: ejemplo (producto)

Dados dos números n y m mayores o iguales que cero, calcular $n \times m$.

Recursión algorítmica: ejemplo (producto)

Dados dos números n y m mayores o iguales que cero, calcular $n \times m$.

```
// Algoritmo iterativo
Producto(n, m) {
    RV ← 0
    while (m > 0) {
        RV ← RV + n
        m ← m − 1
    }
}
```

Recursión algorítmica: ejemplo (producto)

Dados dos números n y m mayores o iguales que cero, calcular $n \times m$.

```
// Algoritmo iterativo
Producto(n, m) {
    RV ← 0
    while (m > 0) {
        RV ← RV + n
        m ← m − 1
    }
}
```

```
// Algoritmo recursivo
Producto(n, m) {
    if (m = 0) {
        RV ← 0
    } else {
        RV ← Prod(n, m − 1) + n
    }
}
```

Recursión algorítmica: otro ejemplo (factorial)

Dado un número natural n, calcular $n!$.

Recursión algorítmica: otro ejemplo (factorial)

Dado un número natural n, calcular $n!$.

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

Recursión algorítmica: otro ejemplo (factorial)

Dado un número natural n, calcular $n!$.

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

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

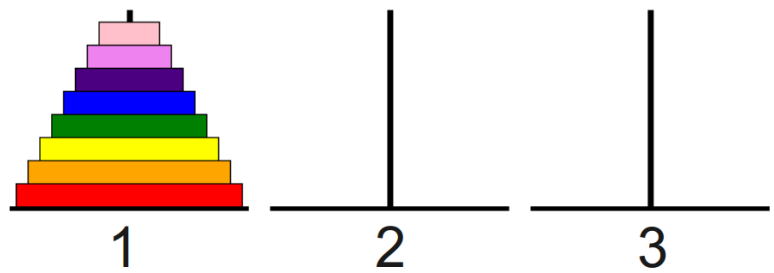
Herramienta poderosa para encontrar algoritmos para problemas no triviales, mediante técnicas como Divide and conquer o Backtracking.

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

- 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

- 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

- 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

- 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.

Ejemplo de D&C: Mergesort

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

1. **Divide:**

Ejemplo 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 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 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 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 $O(n \log n)$.

Ejemplo 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 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 de D&C: Búsqueda Binaria Recursiva

$Buscar\ (x, A) \rightarrow (está, pos)$

Ejemplo de D&C: Búsqueda Binaria Recursiva

$Buscar\ (x, A) \to (está, pos)$

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

Ejemplo de D&C: Búsqueda Binaria Recursiva

Buscar $(x, A) \to (\textit{está}, \textit{pos})$

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

BuscarDesdeHasta $(x, A, \textit{desde}, \textit{hasta}) \to (\textit{está} \times \textit{pos})$

if $(\textit{desde} = \textit{hasta})$ {

$(\textit{está}, \textit{pos}) \leftarrow (x = A[\textit{desde}],\ \textit{desde})$

} else {

Ejemplo de D&C: Búsqueda Binaria Recursiva

$Buscar\ (x, A) \to (está, pos)$

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

$BuscarDesdeHasta\ (x, A, desde, hasta) \to (está \times pos)$

```
if (desde = hasta) {
    (está, pos) ← (x = A[desde], desde)
} else {
    med ← (desde + hasta) div 2
    if (A[med] < x) {
        (está, pos) ←
                    BuscarDesdeHasta(x, A, med + 1, hasta)
    } else {
        (está, pos) ← BuscarDesdeHasta(x, A, desde, med)
    }
```

Ejemplo de D&C: Búsqueda Binaria Recursiva

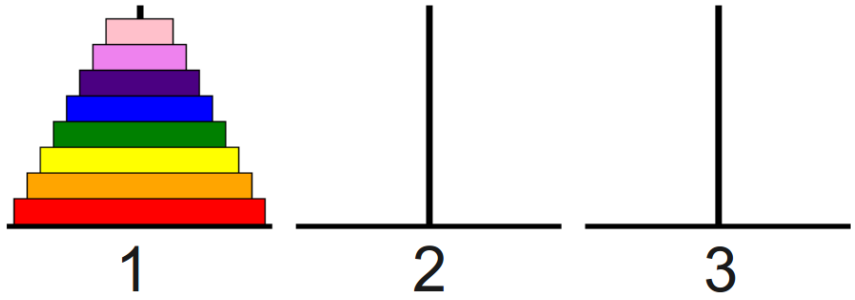
$Buscar\ (x, A) \to (está, pos)$

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

$BuscarDesdeHasta\ (x, A, desde, hasta) \to (está \times pos)$

```
if (desde = hasta) {
    (está, pos) ← (x = A[desde], desde)
} else {
    med ← (desde + hasta) div 2
    if (A[med] < x) {
        (está, pos) ←
                    BuscarDesdeHasta(x, A, med + 1, hasta)
    } else {
        (está, pos) ← BuscarDesdeHasta(x, A, desde, med)
    }
}
```

Ejemplo 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.webgamesonline.com/towers-of-hanoi/

Ejemplo de D&C: Torre de Hanoi

Hanoi(n):
    Hanoi_aux(n,1,3,2)

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

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

Repaso de la clase de hoy

- Recursión algorítmica.
- Producto, factorial.
- Divide & Conquer.
- Mergesort, Hanoi, Búsqueda binaria.

Próximos temas

- Backtracking.
- Tipos abstractos de datos.