Algoritmos y estructuras de datos

Recursión y eficiencia

Francisco Javier Zaragoza Martínez

Universidad Autónoma Metropolitana Unidad Azcapotzalco
Departamento de Sistemas

20 de mayo de 2015

Recursión

Recursión

La recursión es un método en el cual la solución a un problema se alcanza resolviendo primero instancias más pequeñas del mismo. La recursión es una de las ideas centrales de la computación.

Ejemplos clásicos

1. Cálculo de factoriales.
2. Cálculo de potencias.
3. Máximo común divisor.
4. Búsqueda binaria.
5. Números de Fibonacci.

Cálculo de factoriales

Definición de los factoriales

Usualmente se define $n!$ como el producto $1 \cdot 2 \cdots (n-1) \cdot n$.

Definición recursiva de los factoriales

Podemos definir los factoriales de esta manera:

$$f(n) = \begin{cases} 1 & \text{si } n = 0 \\ n \cdot f(n-1) & \text{si } n > 0 \end{cases}$$

Cálculo de factoriales

Ejemplo

$$
\begin{aligned}
f(3) &= 3 \cdot f(2) \\
&= 3 \cdot 2 \cdot f(1) \\
&= 3 \cdot 2 \cdot 1 \cdot f(0) \\
&= 3 \cdot 2 \cdot 1 \cdot 1 \\
&= 3 \cdot 2 \cdot 1 \\
&= 3 \cdot 2 \\
&= 6.
\end{aligned}
$$

Árbol de recursión

$$
\begin{array}{c}
f(3) = 3 \cdot f(2) \\
\downarrow \\
f(2) = 2 \cdot f(1) \\
\downarrow \\
f(1) = 1 \cdot f(0) \\
\downarrow \\
f(0) = 1
\end{array}
$$

Árbol de recursión

$$
\begin{array}{c}
3 \\
\downarrow \\
2 \\
\downarrow \\
1 \\
\downarrow \\
0
\end{array}
$$

Cálculo de factoriales

Definición de los factoriales

Podemos definir los factoriales de esta manera:

$$f(n) = \begin{cases} 1 & \text{si } n = 0 \\ n \cdot f(n-1) & \text{si } n > 0 \end{cases}$$

Implementación

```
int factorial(int n)
{
  if (n == 0)
    return 1;
  else
    return n*factorial(n-1);
}
```

Cálculo de factoriales

Ejercicios

1. ¿Cuántas llamadas recursivas hace `factorial(n)` en términos de `n`?
2. ¿Cuántas multiplicaciones hace `factorial(n)` en términos de `n`?
3. Escribe una función iterativa `int factorial(int n)`.
4. ¿Cuántas multiplicaciones hace tu función en términos de `n`?
5. Escribe funciones iterativa y recursiva `int suma(int n)` que calculen la suma $s(n) = 1 + 2 + \cdots + n$. Para la versión recursiva observe que si $n > 0$ entonces $s(n) = n + s(n-1)$.
6. Escribe funciones iterativa y recursiva `int suma(int n, int a[])` que calculen la suma $s(n) = a_0 + a_1 + \cdots + a_{n-1}$.

Cálculo de potencias

Potencias

Podemos definir la exponenciación de esta manera:

$$f(x,n) = \begin{cases} 1 & \text{si } n = 0 \\ x \cdot f(x, n-1) & \text{si } n > 0 \end{cases}$$

Cálculo de potencias

Ejemplo

$$
\begin{aligned}
f(2,3) &= 2 \cdot f(2,2) \\
&= 2 \cdot 2 \cdot f(2,1) \\
&= 2 \cdot 2 \cdot 2 \cdot f(2,0) \\
&= 2 \cdot 2 \cdot 2 \cdot 1 \\
&= 2 \cdot 2 \cdot 2 \\
&= 2 \cdot 4 \\
&= 8.
\end{aligned}
$$

Árbol de recursión

$$
\begin{aligned}
f(2,3) &= 2 \cdot f(2,2) \\
&\downarrow \\
f(2,2) &= 2 \cdot f(2,1) \\
&\downarrow \\
f(2,1) &= 2 \cdot f(2,0) \\
&\downarrow \\
f(2,0) &= 1
\end{aligned}
$$

Árbol de recursión

$$
\begin{aligned}
3 \\
&\downarrow \\
2 \\
&\downarrow \\
1 \\
&\downarrow \\
0
\end{aligned}
$$

Cálculo de potencias

Potencias

Podemos definir la exponenciación de esta manera:

$$f(x,n) = \begin{cases} 1 & \text{si } n = 0 \\ x \cdot f(x, n-1) & \text{si } n > 0 \end{cases}$$

Implementación

```
double potencia(double x, int n)
{
  if (n == 0)
    return 1.0;
  else
    return x*potencia(x, n-1);
}
```

Cálculo de potencias

Ejercicios

1. ¿Cuántas llamadas recursivas hace `potencia(x,n)` en términos de `n`?
2. ¿Cuántas multiplicaciones hace `potencia(x,n)` en términos de `n`?
3. Escribe una función iterativa `double potencia(double x, int n)`.
4. ¿Cuántas multiplicaciones hace tu función en términos de `n`?
5. Escribe una nueva función recursiva que calcule x^n basada en la siguiente idea: Si $n = 0$, entonces $x^n = 1$. Si $n = 1$, entonces $x^n = x$. Si $n \geq 2$ es par, entonces $x^n = (x^{n/2})^2$. Finalmente, si $n \geq 3$ es impar, entonces $x^n = x \cdot (x^{(n-1)/2})^2$. ¿Cuántas multiplicaciones y llamadas recursivas se hacen en términos de n?

Máximo común divisor

Algoritmo de Euclides

Podemos calcular el máximo común divisor de esta manera:

$$f(a,b) = \begin{cases} a & \text{si } b = 0 \\ f(b, a \mod b) & \text{si } b > 0 \end{cases}$$

Máximo común divisor

Ejemplo

$$
\begin{aligned}
f(54,42) &= f(42, 54 \mod 42) \\
&= f(42,12) \\
&= f(12, 42 \mod 12) \\
&= f(12,6) \\
&= f(6, 12 \mod 6) \\
&= f(6,0) \\
&= 6.
\end{aligned}
$$

Árbol de recursión

$$
\begin{array}{c}
f(54,42) = f(42,12) \\
\downarrow \\
f(42,12) = f(12,6) \\
\downarrow \\
f(12,6) = f(6,0) \\
\downarrow \\
f(6,0) = 6
\end{array}
$$

Árbol de recursión

$$
\begin{array}{c}
(54,42) \\
\downarrow \\
(42,12) \\
\downarrow \\
(12,6) \\
\downarrow \\
(6,0)
\end{array}
$$

Máximo común divisor

Algoritmo de Euclides

Podemos calcular el máximo común divisor de esta manera:

$$f(a,b) = \begin{cases} a & \text{si } b = 0 \\ f(b, a \mod b) & \text{si } b > 0 \end{cases}$$

Implementación

```
int euclides(int a, int b)
{
  if (b == 0)
    return a;
  else
    return euclides(b, a % b);
}
```

Ejercicios

Ejercicios

1. Los coeficientes combinatorios $C(n, m)$ se definen recursivamente de la siguiente manera: Si $m = 0$ o $m = n$, entonces $C(n, m) = 1$. Si $0 < m < n$, entonces $C(n, m) = C(n-1, m-1) + C(n-1, m)$. Escribe una función recursiva `int combina(int n, int m)` que calcule $C(n, m)$. ¿Cuántas sumas hace la llamada `combina(n,m)`?
2. También hay una fórmula para los coeficientes combinatorios:

$$C(n, m) = \frac{n!}{m!(n-m)!}.$$

Escribe una función iterativa que calcule $C(n, m)$. ¿Cuántos productos hace tu función?

Búsqueda binaria

Con arreglos

```
int binaria(int n, int a[], int x)
{
  int i = 0, j = n-1, m;

  while (i <= j) {
    m = (i+j)/2;
    if (x == a[m])
      return m;
    if (x < a[m])
      j = m-1;
    else
      i = m+1;
  }
  return -1;
}
```

Búsqueda binaria

Podemos hacer búsqueda binaria de x en el arreglo $(a_i, \ldots, a_j)$ así:

$$f(i,j) = \begin{cases} -1 & \text{si } i > j \\ m & \text{si } x = a_m \\ f(i, m-1) & \text{si } x < a_m \\ f(m+1, j) & \text{si } x > a_m \end{cases}$$

donde $m = \lfloor \frac{i+j}{2} \rfloor$.

Búsqueda binaria

Ejemplo: buscando 42

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	11	14	23	26	31	39	40	42	43	51	55	67	70	72	79
								42	43	51	55	67	70	72	79
								42	43	51					
								42							

$$
\begin{aligned}
f(0,15) &= f(\lfloor \tfrac{0+15}{2} \rfloor + 1, 15) \\
&= f(8,15) \\
&= f(8, \lfloor \tfrac{8+15}{2} \rfloor - 1) \\
&= f(8,10) \\
&= f(8, \lfloor \tfrac{8+10}{2} \rfloor - 1) \\
&= f(8,8) \\
&= 8.
\end{aligned}
$$

Búsqueda binaria

Ejemplo

$$
\begin{aligned}
f(0,15) &= f(\lfloor \tfrac{0+15}{2} \rfloor + 1, 15) \\
&= f(8,15) \\
&= f(8, \lfloor \tfrac{8+15}{2} \rfloor - 1) \\
&= f(8,10) \\
&= f(8, \lfloor \tfrac{8+10}{2} \rfloor - 1) \\
&= f(8,8) \\
&= 8.
\end{aligned}
$$

Árbol de recursión

$$
\begin{array}{c}
f(0,15) = f(8,15) \\
\downarrow \\
f(8,15) = f(8,10) \\
\downarrow \\
f(8,10) = f(8,8) \\
\downarrow \\
f(8,8) = 8
\end{array}
$$

Árbol de recursión

$$
\begin{array}{c}
(0,15) \\
\downarrow \\
(8,15) \\
\downarrow \\
(8,10) \\
\downarrow \\
(8,8)
\end{array}
$$

Búsqueda binaria

Con arreglos y recursiva

```
int binaria(int i, int j, int a[], int x)
{
  int m = (i+j)/2;

  if (i > j)
    return -1;
  else if (x == a[m])
    return m;
  else if (x < a[m])
    return binaria(i, m-1, a, x);
  else
    return binaria(m+1, j, a, x);
}
```

Búsqueda binaria

Ejercicios

1. Escribe una función recursiva `int lineal(int n, int a[], int x)` que lleve a cabo la búsqueda lineal en un arreglo desordenado.
2. Escribe una función recursiva `int lineal(int n, int a[], int x)` que lleve a cabo la búsqueda lineal en un arreglo ordenado.

Números de Fibonacci

Números de Fibonacci

Los números de Fibonacci se definen de esta manera:

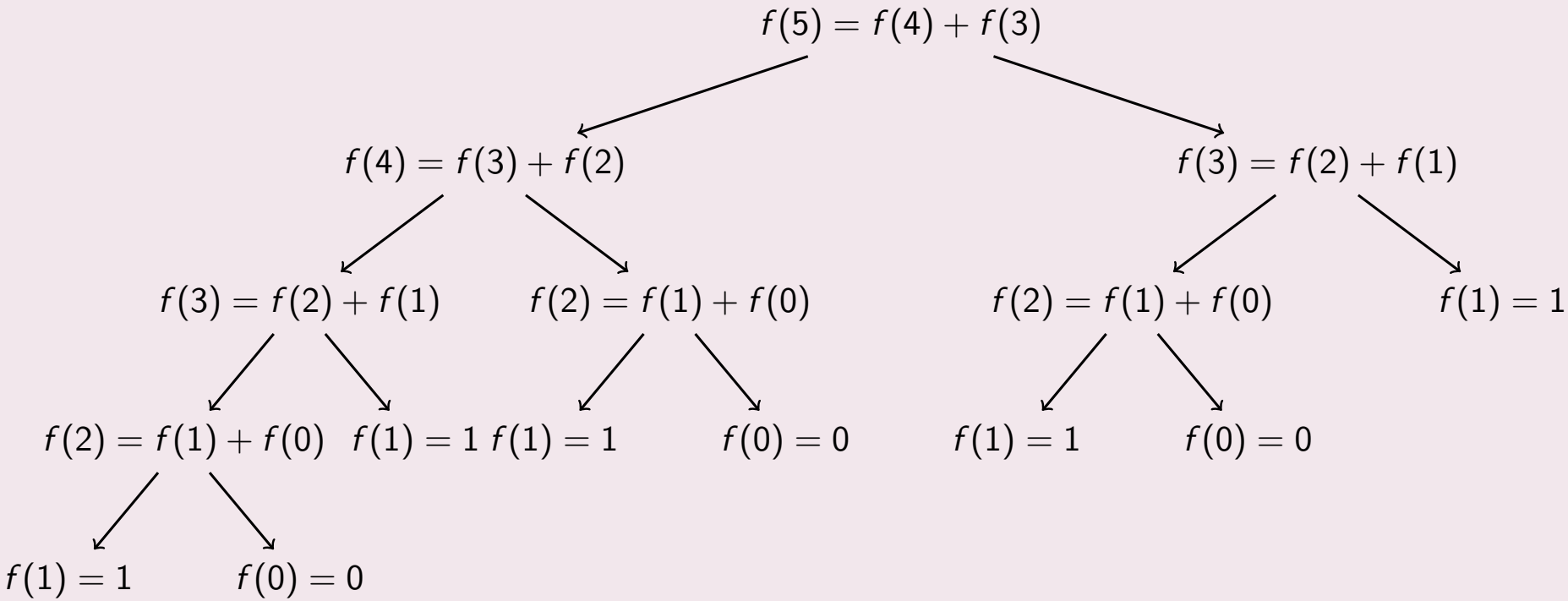
$$f(n) = \begin{cases} 0 & \text{si } n = 0 \\ 1 & \text{si } n = 1 \\ f(n-1) + f(n-2) & \text{si } n > 1 \end{cases}$$

Ejemplo

$$\begin{aligned} f(5) &= f(4) + f(3) \\ &= (f(3) + f(2)) + (f(2) + f(1)) \\ &= ((f(2) + f(1)) + (f(1) + f(0))) + ((f(1) + f(0)) + 1) \\ &= (((f(1) + f(0)) + 1) + (1 + 0)) + ((1 + 0) + 1) \\ &= (((1 + 0) + 1) + (1 + 0)) + ((1 + 0) + 1) \\ &= 5. \end{aligned}$$

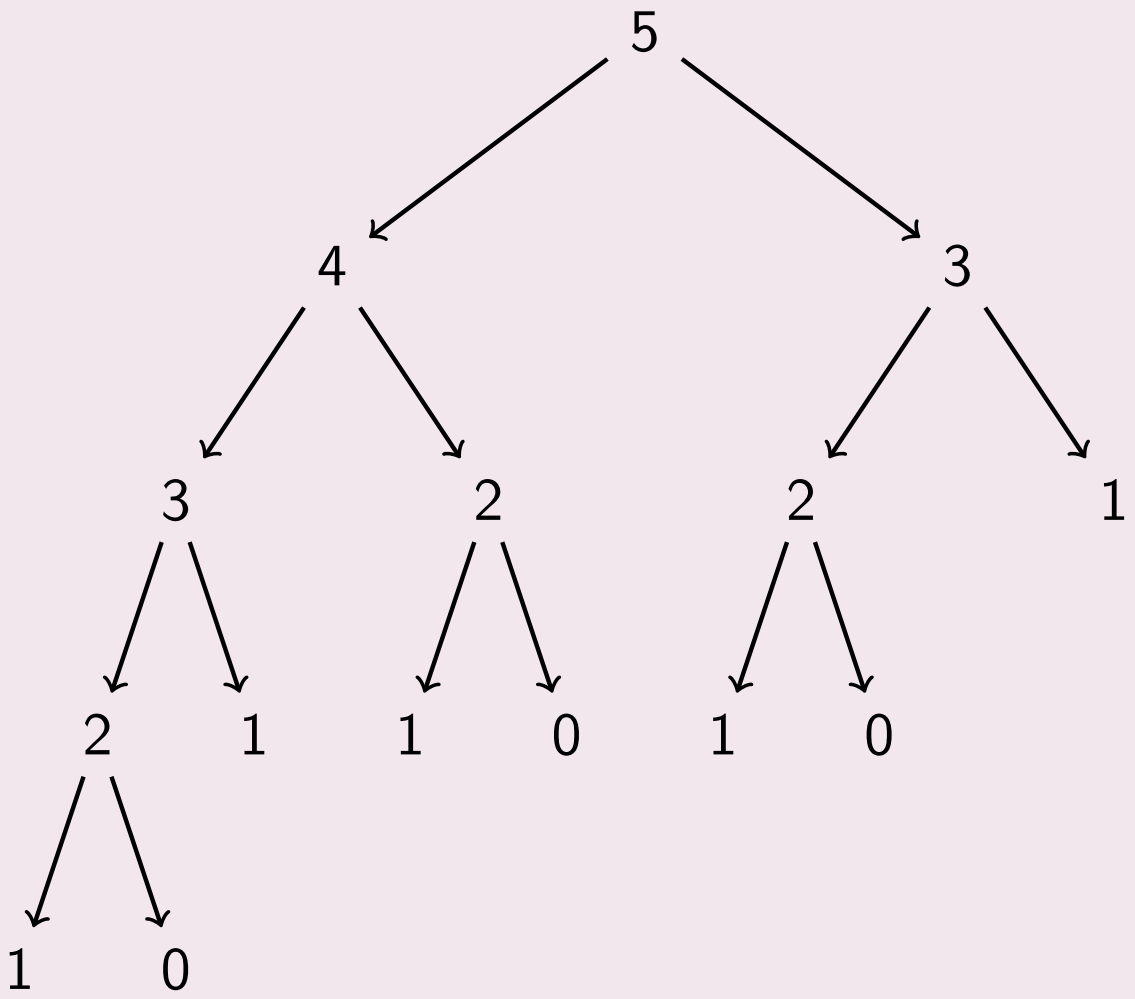
Números de Fibonacci

Árbol de recursión



Números de Fibonacci

Árbol de recursión



Números de Fibonacci

Números de Fibonacci

Los números de Fibonacci se definen de esta manera:

$$f(n) = \begin{cases} 0 & \text{si } n = 0 \\ 1 & \text{si } n = 1 \\ f(n-1) + f(n-2) & \text{si } n > 1 \end{cases}$$

Implementación

```
int fibonacci(int n)
{
  if (n == 0)
    return 0;
  else if (n == 1)
    return 1;
  else
    return fibonacci(n-1)+fibonacci(n-2);
}
```

Números de Fibonacci

Ejercicios

1. Escribe una función iterativa que encuentre $f(n)$ calculando de forma sucesiva $f(0), f(1), f(2), \ldots$ ¿Cuántas sumas hace en términos de n?
2. Escribe una nueva función recursiva que calcule $f(n)$ basada en la siguiente idea: Si $n = 0$, entonces $f(n) = 0$. Si $n = 1$, entonces $f(n) = 1$. Si $n \geq 2$ es par, entonces $f(n) = f(\frac{n}{2} + 1)^2 - f(\frac{n}{2} - 1)^2$. Finalmente, si $n \geq 3$ es impar, entonces $f(n) = f(\frac{n+1}{2})^2 + f(\frac{n-1}{2})^2$. ¿Cuántas llamadas recursivas se hacen en términos de n?
3. Los números de Lucas se definen como $L(0) = 2$, $L(1) = 1$ y $L(n) = L(n-1) + L(n-2)$ para $n \geq 2$. Escribe funciones iterativa y recursiva `int lucas(int n)` que calculen $L(n)$.
4. Los números de Pell se definen como $P(0) = 0$, $P(1) = 1$ y $P(n) = 2P(n-1) + P(n-2)$ para $n \geq 2$. Escribe funciones iterativa y recursiva `int pell(int n)` que calculen $P(n)$.