Resolución de Problemas y Algoritmos

Clase 17:
Resolución de problemas utilizando recursión

Dr. Alejandro J. García
http://cs.uns.edu.ar/~ajg

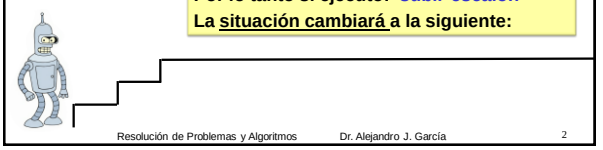
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca - Argentina

Soluciones recursivas

Considere un escenario donde debemos programar un robot para subir una escalera, y se dispone de dos primitivas (una de *percepción* y otra de *efecto*):

- **quedan-escalones: función que retorna TRUE si hay escalones al frente, o FALSE en caso contrario.**
- **subir-escalón: hace al robot subir un escalón.**

En la situación representada por el dibujo :
quedan-escalones retorna TRUE
Por lo tanto si ejecuto: subir-escalón
La situación cambiará a la siguiente:

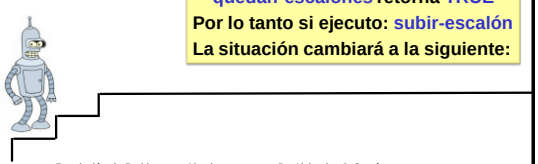


Soluciones recursivas

Considere un escenario donde debemos programar un robot para subir una escalera, y se dispone de las primitivas (una de *percepción* y otra de *efecto*):

- **quedan-escalones: función que retorna TRUE si hay escalones al frente, o FALSE en caso contrario.**
- **subir-escalón: hace al robot subir un escalón.**

En esta nueva situación:
quedan-escalones retorna TRUE
Por lo tanto si ejecuto: subir-escalón
La situación cambiará a la siguiente:

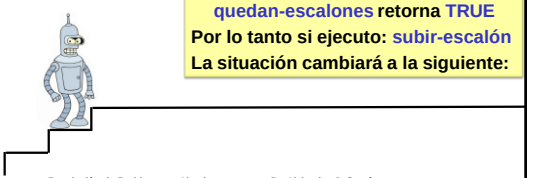


Soluciones recursivas

Considere un escenario donde debemos programar un robot para subir una escalera, y se dispone de las primitivas (una de *percepción* y otra de *efecto*):

- **quedan-escalones: función que retorna TRUE si hay escalones al frente, o FALSE en caso contrario.**
- **subir-escalón: hace al robot subir un escalón.**

En esta nueva situación:
quedan-escalones retorna TRUE
Por lo tanto si ejecuto: subir-escalón
La situación cambiará a la siguiente:

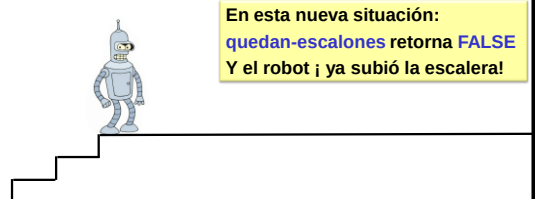


Soluciones recursivas

Considere un escenario donde debemos programar un robot para subir una escalera, y se dispone de las primitivas (una de *percepción* y otra de *efecto*):

- **quedan-escalones: función que retorna TRUE si hay escalones al frente, o FALSE en caso contrario.**
- **subir-escalón: hace al robot subir un escalón.**

En esta nueva situación:
quedan-escalones retorna FALSE
Y el robot ¡ ya subió la escalera!



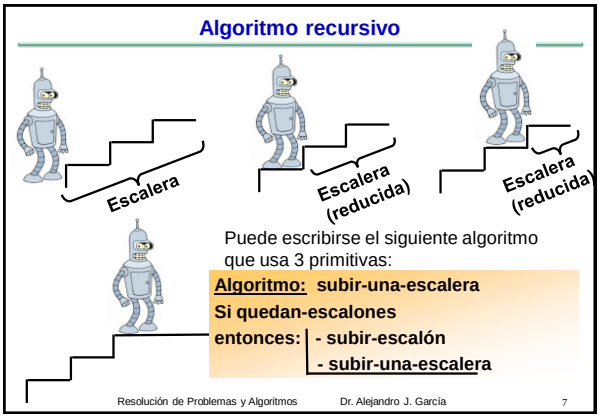
Soluciones recursivas

Observación: a excepción de la última situación, el robot siempre tiene una "escalera" por delante.

Una escalera puede verse entonces como "un simple escalón, o un escalón seguido de una escalera".

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
***"Resolución de Problemas y Algoritmos. Notas de Clase".* Alejandro J. García. Universidad Nacional del Sur. (c) 2015**

Algoritmo recursivo



Puede escribirse el siguiente algoritmo que usa 3 primitivas:

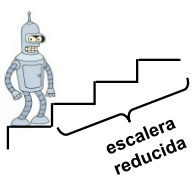
Algoritmo: subir-una-escalera
Si quedan-escalones
entonces: - subir-escalón
- subir-una-escalera

Conceptos: algoritmos recursivos

- **Recursión es la forma en la cual se especifica un proceso basado en su propia definición.**
- **Un algoritmo es recursivo si se define en términos de sí mismo.**
- **Un algoritmo no debe entrar en una ejecución infinita, por lo tanto:**
- **Un algoritmo recursivo será válido, si:**
  (a) existe un caso base que no se define en términos de si mismo, y
  (b) existe un caso general donde la referencia a sí mismo es sobre una instancia más sencilla (o reducida) que el caso considerado.

Algoritmos recursivos

- **Un algoritmo recursivo será válido, si:**
  (a) existe un caso base que no se define en términos de si mismo, y
  (b) existe un caso general donde la referencia a sí mismo es sobre una instancia más sencilla (o reducida) que el caso considerado.



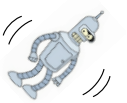
Algoritmo: subir-una-escalera
Si quedan-escalones
entonces: - subir-escalón
- subir-una-escalera

OK

Algoritmos recursivos

- **Un algoritmo recursivo será válido, si:**
  (a) existe un caso base que no se define en términos de si mismo, y
  (b) existe un caso general donde la referencia a sí mismo es sobre una instancia más sencilla (o reducida) que el caso considerado.

¿Por qué es incorrecto?



Algoritmo 2 subir-escalera
-subir-escalón
-subir-escalera

MAL

Falla porque no hay indicado un caso base (a).
¿ Termina de ejecutarse ?

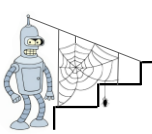
Algoritmos recursivos

- **Un algoritmo recursivo será válido, si:**
  (a) existe un caso base que no se define en términos de si mismo, y
  (b) existe un caso general donde la referencia a sí mismo es relativamente más sencilla o reducida que el caso considerado.

¿Por qué es incorrecto?

Algoritmo 3 subir-la-escalera
Si quedan-escalones
entonces: - subir-la-escalera
- subir-escalón

MAL



Falla (b): la referencia a sí mismo NO es relativamente más sencilla o reducida (es igual).
¿Termina de ejecutarse? ¿sube un escalón?

Concepto: planteo recursivo

La forma de resolver un problema puede plantearse de manera recursiva si se indica:
(a) un caso base que no se define en términos de si mismo, y
(b) un caso general donde se hace referencia a sí mismo, pero con una instancia reducida del problema.

- **De esta forma, al utilizar este tipo de planteo, el problema queda divido en dos sub-problemas:**
  (1) caso base (también llamado caso trivial) y
  (2) caso general (también llamado caso recursivo).
- **Para indicar como resolver un problema de manera recursiva en RPA usaremos un planteo recursivo.**

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
***"Resolución de Problemas y Algoritmos. Notas de Clase".* Alejandro J. García. Universidad Nacional del Sur. (c) 2015**

Concepto: planteo recursivo

La forma de resolver un problema puede plantearse de manera recursiva si se indica:

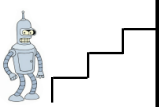
(a) un caso base que no se define en términos de si mismo, y

(b) un caso general donde se hace referencia a sí mismo, pero con una instancia reducida del problema.

Observación: Una escalera puede verse como "un simple escalón, o un escalón seguido de una escalera"

Planteo recursivo: subir una escalera

Caso base:
si hay un solo escalón, subo el escalón.



Caso general: si hay más de un escalón,
primero subo un escalón,
y luego subir una escalera que tiene un escalón menos.

Concepto: planteo recursivo

Un planteo recursivo es una solución a un problema donde:
(a) se indica un caso base que no se define en términos de si mismo, y además,

(b) se indica un caso general donde se hace referencia a sí mismo, pero con una instancia reducida del problema.

Problema propuesto: controlar un conjunto de 1 o más solicitudes de beca para asistir a un evento.

Planteo recursivo: Controlar un conjunto de solicitudes

Caso base:
si hay una sola solicitud, controlar dicha solicitud

Caso general: si hay más de una solicitud
Controlar una solicitud y luego controlar un conjunto de solicitudes sin considerar la ya contralada.

Concepto: planteo recursivo

La forma de resolver un problema puede plantearse de manera recursiva si se indica:

(a) un caso base que no se define en términos de si mismo, y

(b) un caso general donde se hace referencia a sí mismo, pero con una instancia reducida del problema.

- **Generalmente los trabajos de impresión (jobs) son enviados a una cola de impresión (queue) antes de ser impresos.**
  Se puede escribir una solución recursiva para el siguiente problema.

Problema propuesto: imprimir todos los documentos de una cola de impresión que puede tener 1 o más documentos.

Concepto: planteo recursivo

La forma de resolver un problema puede plantearse de manera recursiva si se indica:

(a) un caso base que no se define en términos de si mismo, y

(b) un caso general donde se hace referencia a sí mismo, pero con una instancia reducida del problema.

Problema propuesto: imprimir todos los documentos de una cola de impresión que puede tener 1 o más documentos.

Planteo recursivo: Imprimir trabajos de la cola de impresión Q

Caso base:
si hay un solo documento en Q, enviar a la impresora

Caso general: si hay más de un documento en Q,
enviar el primero de Q a la impresora, y luego imprimir trabajos de la cola de impresión Q sin incluir al primero.

Concepto: planteo recursivo

La forma de resolver un problema puede plantearse de manera recursiva si se indica:

(a) un caso base que no se define en términos de si mismo, y

(b) un caso general donde se hace referencia a sí mismo, pero con una instancia reducida del problema.

Problema propuesto: escribir un planteo para comer un plato de ñoquis (que no está vacio).

Planteo recursivo: Comer un plato de ñoquis

Caso base:
si hay un solo ñoqui en el plato, comer un ñoqui.

Caso general: si hay más de un ñoqui en el plato,
comer un ñoqui y luego comer un plato de ñoquis.

Concepto: planteo recursivo

La forma de resolver un problema puede plantearse de manera recursiva si se indica:

(a) un caso base que no se define en términos de si mismo, y

(b) un caso general donde se hace referencia a sí mismo, pero con una instancia reducida del problema.

Problema propuesto: escribir un planteo para tomar un vaso de una bebida de a sorbos (que no está vacio).

Planteo recursivo: tomar un vaso de bebida

Caso base:

Caso general:

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
***"Resolución de Problemas y Algoritmos. Notas de Clase".* Alejandro J. García. Universidad Nacional del Sur. (c) 2015**

Soluciones recursivas en Pascal

- Los procedimientos y funciones de Pascal nos permitirán implementar soluciones recursivas.
- A continuación, para mostrar como hacer funciones y procedimientos recursivos, vamos a usar ejemplos un poco más simples que los problemas asociados a programar el comportamiento de robots.
- Como verá en otras materias y en su vida profesional, "recursión" es una herramienta mucho más general y poderosa que la "iteración". Verá por ejemplo que hay lenguajes de programación que solo tienen recursión.

Metodología propuesta

1. Identificar ejemplos significativos que ayuden a entender el problema y su solución.
2. Realizar un planteo recursivo en el cual se distinga el "caso base", y el "caso general" (donde se define en términos de si mismo pero para una instancia más simple/reducida/menor).
3. Verificar que el planteo sea correcto (con alguno de los ejemplos significativos).
4. Determinar si se realizará una función o un procedimiento recursivo, e implementarlo en Pascal.
5. Realizar la traza de la primitiva en Pascal.

Problema propuesto: 2^N

Escribir una función recursiva en Pascal para computar la función 2^N (N no negativo).

Para N no negativo, la función 2^N puede definirse recursivamente de la siguiente manera:

$$2^N \begin{cases} 2^0 = 1 & \text{(para N=0)} \\ 2^N = 2 * 2^{N-1} & \text{para N>0} \end{cases}$$

Observe que ya está dividido en 2 casos.

Planteo recursivo para 2^N

- Caso base o trivial: si N=0 entonces 2^N es 1
- Caso general o caso recursivo: Si N>0 entonces 2^N es $2 * 2^{N-1}$

Problema propuesto: 2^N

```
function dosAlaN (N:integer):integer;
var aux,nuevoN:integer;
 begin {función recursiva para 2 a la N}
  if (N = 0) then dosAlaN :=1 {caso base}
  else begin {caso general}
       nuevoN:=N-1;
       aux:= dosAlaN(nuevoN);
       dosAlaN:=2 * aux;
      end;
end;
```

Esta es una posible función en Pascal que respeta el planteo (pero no es la única).

Planteo recursivo para 2^N

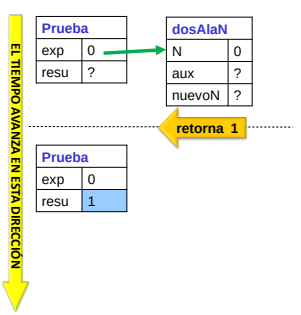
- Caso base o trivial: si N=0 entonces 2^N es 1
- Caso general o caso recursivo: Si N>0 entonces 2^N es $2 * 2^{N-1}$


```
program Prueba; {Programa de prueba para 2 a la N}
var  exp, resu: integer;
    function dosAlaN (N:integer):integer;
    var aux,nuevoN:integer;
      begin {función recursiva para 2 a la N}
       if (N = 0) then dosAlaN :=1 {caso base}
       else begin {caso general}
            nuevoN:=N-1;
            aux:= dosAlaN(nuevoN);
            dosAlaN:=2 * aux;
           end;
    end;
{El programa valida la entrada y llama a la función recursiva}
begin
writeln(' Ingrese un exponente >= 0');
repeat readln(exp) until exp>=0;
resu:=dosAlaN(exp); writeln(' 2 a la ',exp,' es ' , resu );
end.
```

Realice trazas para exp=0 y exp=3.

¿Cuántas veces se llama a la función dosAlaN con respecto al valor de exp?

Traza ingresando 0

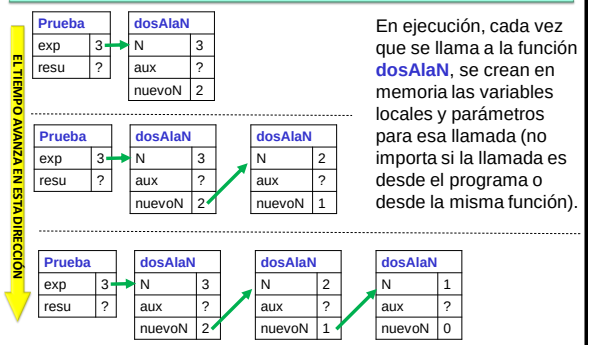


En ejecución, cada vez que se llama a un bloque (programa, función o procedimiento), se crean en memoria las variables locales y parámetros para esa llamada.

Al terminar la ejecución de una llamada, se eliminan de memoria esas variables locales y sus valores desaparecen

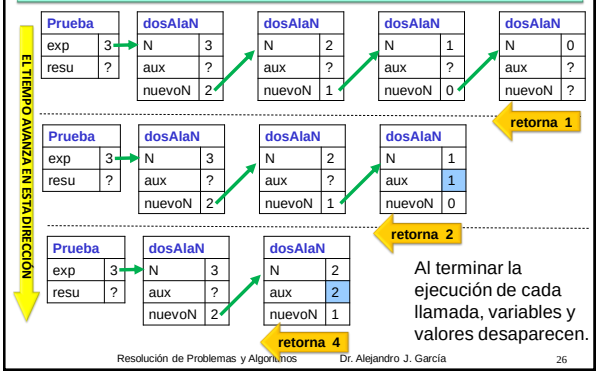
El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
"Resolución de Problemas y Algoritmos. Notas de Clase". **Alejandro J. García. Universidad Nacional del Sur. (c) 2015**

Traza ingresando 3 (primera parte)



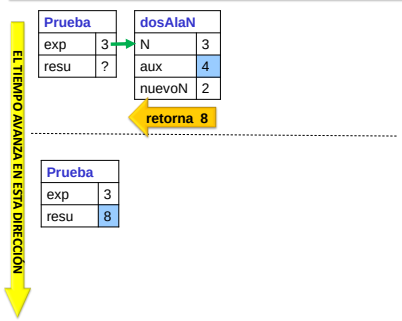
En ejecución, cada vez que se llama a la función **dosAlaN**, se crean en memoria las variables locales y parámetros para esa llamada (no importa si la llamada es desde el programa o desde la misma función).

Traza ingresando 3 (segunda parte)



Al terminar la ejecución de cada llamada, variables y valores desaparecen.

Traza ingresando 3 (última parte)



Problema propuesto: 2^N

```
function dosAlaN (N:integer):integer;
{Otra versión de una función recursiva para
2 a la N (que no usa variables auxiliares)}
begin
  if (N = 0)   then dosAlaN :=1 {caso base}
  else dosAlaN := 2 * dosAlaN(N-1); {c. general}
end;
```

Esta es otra forma de implementar en Pascal la función recursiva que respeta el mismo planteo recursivo.

Planteo recursivo para 2^N

- **Caso base o trivial: si N=0 entonces 2^N es 1**
- **Caso general o caso recursivo: Si N>0 entonces 2^N es $2 * 2^{N-1}$**


```
program Prueba2; {Prueba otra versión de 2 a la N}
var  exp, resu: integer;

function  dosAlaN (N:integer):integer;
{Otra versión de una función recursiva para
2 a la N (que no usa variables auxiliares)}
begin
  if (N = 0) then dosAlaN :=1 {caso base}
  else dosAlaN := 2 * dosAlaN(N-1); {general}
end;

{El programa valida la entrada y llama a la función recursiva}
begin
writeln(' Ingrese un exponente >= 0');
repeat readln(exp) until exp>=0;
resu:=dosAlaN(exp); writeln(' 2 a la ',exp,' es ' , resu );
end.
```

Realice trazas para exp=0 y exp=3.

¿Cuántas veces se llama a la función dosAlaN con respecto al valor de exp?

Problema propuesto

Escriba un programa que permita ingresar por teclado una secuencia de caracteres terminada en un punto (por ejemplo: "hola que tal.") y que la muestre por pantalla en orden inverso ("lat euq aloh"). Ejemplo:

Ingrese una cadena terminada en punto: un animal.
Invertida queda así: lamina nu

Planteo Recursivo para MostrarInvertida una secuencia S

caso base: si la secuencia S es solamente un "." mostrar el cartel "Invertida queda así:"

caso general: MostrarInvertida la secuencia S sin su primer elemento, y luego mostrar el primer elemento de S.

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
"*Resolución de Problemas y Algoritmos. Notas de Clase*". Alejandro J. García. Universidad Nacional del Sur. (c) 2015

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
"*Resolución de Problemas y Algoritmos. Notas de Clase*". Alejandro J. García. Universidad Nacional del Sur. (c) 2015