



Resolución de Problemas y Algoritmos

Clase 19: Resolución de problemas utilizando recursión



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



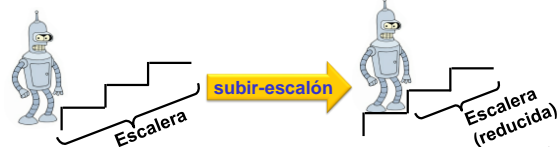
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca - Argentina

Soluciones recursivas

Considere un escenario donde debemos programar un robot para subir una escalera, y se dispone de estas primitivas:

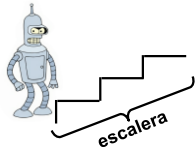
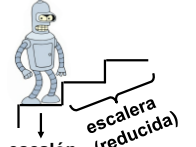

- **hay-un-solo-escalón:** retorna TRUE o FALSE
- **hay-más-de-un-escalón:** retorna TRUE o FALSE
- **subir-escalón:** hace al robot subir un escalón.


Observe el dibujo de la izquierda, donde el robot tiene una escalera por delante, si ejecuto **subir-escalón** el robot tendrá una escalera por delante (reducida en un escalón):



Resolución de Problemas y Algoritmos Dr. Alejandro J. García 2

Soluciones recursivas

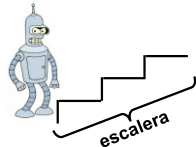
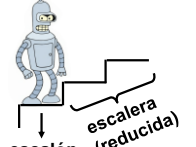
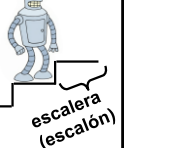



Una **escalera** puede verse entonces como: **un único escalón, o un escalón seguido de una escalera.**

Observación: en las tres situaciones de arriba, el robot siempre tiene una escalera por delante.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 3

Soluciones recursivas



Algoritmo: **subir-una-escalera**
Si **hay-un-solo-escalón** entonces: **subir-escalón**
Si **hay-más-de-un-escalón** entonces: **- subir-escalón**
- subir-una-escalera

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 4

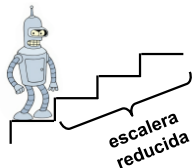
Conceptos: algoritmos recursivos

- **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 sí 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.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 5

Algoritmos recursivos

- Un algoritmo recursivo **será válido**, si:
 - (a) **existe un caso base** que **no** se define en términos de sí 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 **hay-un-solo-escalón** entonces: **subir-escalón**
Si **hay-más-de-un-escalón** entonces: **- subir-escalón**
- subir-una-escalera

OK

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 6

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) 27/05/2016

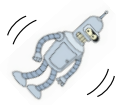
Algoritmos recursivos

- Un algoritmo recursivo **será válido**, si:
 - (a) existe un caso base que **no** se define en términos de sí 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 ?

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 7

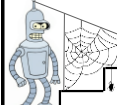
Algoritmos recursivos

- Un algoritmo recursivo **será válido**, si:
 - (a) existe un caso base que **no** se define en términos de sí 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 hay-más-de-un-escalón
 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 ?

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 8

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 sí 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 dividido 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 usemos un planteo recursivo.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 9

Concepto: planteo recursivo

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

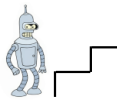
(a) un caso base que **no** se define en términos de sí mismo, y
 (b) un caso general donde se hace referencia a sí mismo, pero con una instancia reducida del problema.

Observación: Una escalera puede verse como “un único 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.



Resolución de Problemas y Algoritmos Dr. Alejandro J. García 10

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 sí 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 controlada.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 11

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 sí mismo, y además,
 (b) se indica 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.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 12

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) 27/05/2016

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 sí 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: 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 único trabajo en Q, enviar a la impresora

Caso general: si hay más de un trabajo en Q, sacar el primero de Q y enviar ia la impresora, y luego [imprimir trabajos de la cola de impresión Q](#).

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 13

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 sí 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 (¡para el 29!): escribir un planteo para comer un plato de ñoquis (que no está vacío).

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](#).



Resolución de Problemas y Algoritmos Dr. Alejandro J. García 14

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 sí 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: (para acompañar los ñoquis) escribir un planteo para tomar un vaso de una bebida de a sorbos (que no está vacío).

Planteo recursivo: [tomar un vaso de bebida](#)

Caso base:

Caso general:

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 15

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.

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 16

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 sí 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.


Resolución de Problemas y Algoritmos Dr. Alejandro J. García 17

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:

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


Observe que ya está dividido en 2 casos.

Planteo recursivo para 2^N

- Caso base: si N=0 entonces 2^N es 1
- Caso general: Si N>0 entonces 2^N es 2 * 2^{N-1}

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 18

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) 27/05/2016

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 una forma de implementar en Pascal la función recursiva que respeta el planteo recursivo (no es la única).

Planteo recursivo para 2^N

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

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 19

```

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 ?

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 20

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 forma de implementar en Pascal la función recursiva que respeta el planteo recursivo (no es la única).

Planteo recursivo para 2^N

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

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 21

```

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 ?

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 22

Traza ingresando 0

Prueba	exp	0	resu	?
--------	-----	---	------	---

→

dosAlaN	N	0	aux	?	nuevoN	?
---------	---	---	-----	---	--------	---

← retorna 1

Prueba	exp	0	resu	1
--------	-----	---	------	---

↓ EL TIEMPO AVANZA EN ESTA DIRECCIÓN

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

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 23

Traza ingresando 3 (primera parte)

Prueba	exp	3	resu	?
--------	-----	---	------	---

→

dosAlaN	N	3	aux	?	nuevoN	2
---------	---	---	-----	---	--------	---

→

dosAlaN	N	2	aux	?	nuevoN	1
---------	---	---	-----	---	--------	---

→

dosAlaN	N	1	aux	?	nuevoN	0
---------	---	---	-----	---	--------	---

→

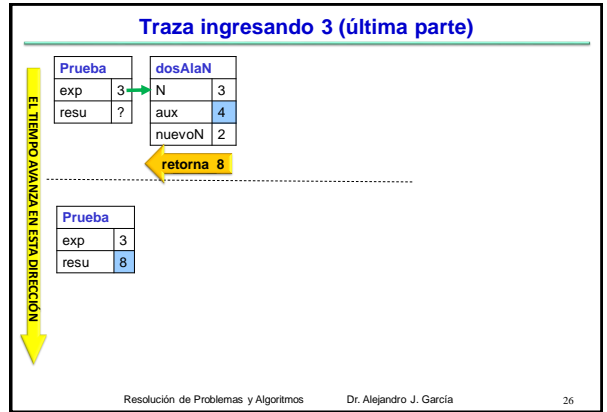
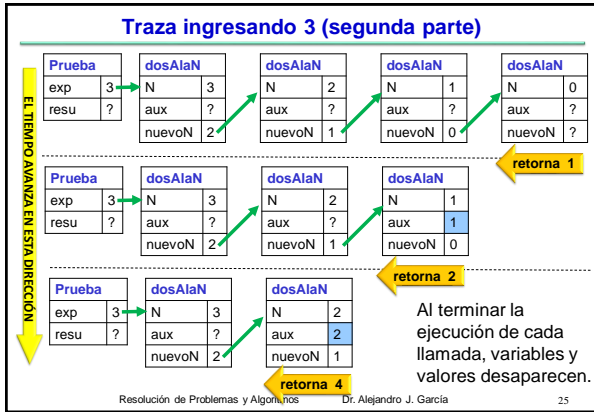
Prueba	exp	3	resu	?
--------	-----	---	------	---

↓ EL TIEMPO AVANZA EN ESTA DIRECCIÓN

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).

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 24

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) 27/05/2016



Continuará ...

Resolución de Problemas y Algoritmos Dr. Alejandro J. García 27

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) 27/05/2016