


Programación dinámica

Profesor: Juan R. Rico 

Desafíos de programación

Dpto. Lenguajes y Sistemas Informáticos. Universidad de Alicante

Versión 1.0



(Algunas partes de este documento están basadas en “Programación Dinámica” de Rafael C. Carrasco)

Índice

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

- 1 Introducción
- 2 Programación dinámica recursiva
- 3 Programación dinámica iterativa
- 4 La mayor subsecuencia común
- 5 El problema de la mochila
- 6 Respuestas a los ejercicios

Índice

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

- 1 **Introducción**
- 2 Programación dinámica recursiva
- 3 Programación dinámica iterativa
- 4 La mayor subsecuencia común
- 5 El problema de la mochila
- 6 Respuestas a los ejercicios

Hoy en día durante las entrevistas de trabajo grandes compañías como Google, Amazon o Microsoft donde se les propone a los candidatos problemas de optimización. Cada vez hay más compañías que realizan este tipo de prueba para contratar a personas con visión para resolver problemas, más que a programadores de software.

Un par de sitios web con ejemplos son:

- [Coder Career](#)
- [Hacker Rank](#)

Origen del nombre

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

La **programación dinámica** debe su nombre a los problemas de *dinámica de sistemas* a los que se aplicó originalmente. Su característica más destacable es que aumenta considerablemente la *eficiencia* de numerosos procedimientos recursivos.

Introducción

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Antes de entrar en detalle en la **programación dinámica** vamos a plantear un ejemplo sencillo basado en el juego del Nim para transmitir la idea básica de esta técnica.

El Nim

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

- El *Nim* es un juego de estrategia que comienza con N fichas en un tablero y cada jugador debe retirar alternativamente entre una y M fichas, mientras queden. Pierde aquel jugador que no es capaz de coger más fichas.
- En todos los juegos que no pueden acabar en empate, uno de los dos jugadores dispone de una estrategia ganadora.
- Una estrategia ganadora es aquella que conduce a la derrota del contrario (independientemente de las jugadas que este último elija).

¿Qué vamos a hacer?

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

- Realizar un diseño sencillo recursivo: primero un planteamiento gráfico, y luego, la función recursiva que relacione la solución de un problema con la de sus descendientes.
- Plantear tres tipos de algoritmos para resolver el problema en orden de dificultad:
 - Recursivo (puro).
 - Recursivo con almacén.
 - Iterativo.

A continuación hay un ejemplo gráfico que podemos usar como base para el planteamiento de los algoritmos.

Ejemplo gráfico Nim. Planteamiento recursivo

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Fichas para jugar, $N = 4$, y máximo número de fichas a retirar por jugada, $M = 2$ (rojo=pierde, verde=gana).

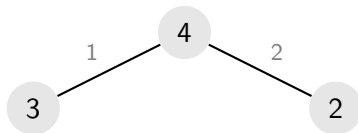


Figura: Árbol de llamadas

Ejemplo gráfico Nim. Planteamiento recursivo

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Fichas para jugar, $N = 4$, y máximo número de fichas a retirar por jugada, $M = 2$ (rojo=pierde, verde=gana).

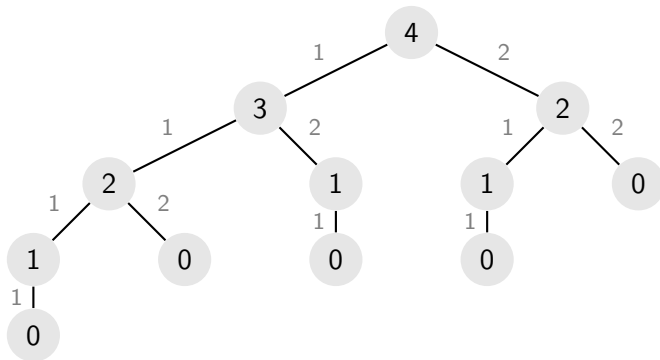


Figura: Árbol de llamadas

Ejemplo gráfico Nim. Planteamiento recursivo

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Fichas para jugar, $N = 4$, y máximo número de fichas a retirar por jugada, $M = 2$ (rojo=pierde, verde=gana).

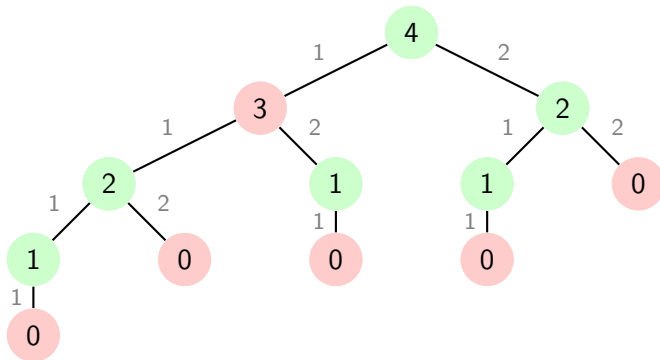


Figura: Árbol de llamadas

Revisar soluciones del Nim

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

- Planteamiento:
 - Recursivo (puro).
 - Recursivo con almacén.
 - Iterativo.
- Revisar el código del P004.java (Campus Virtual).

Revisión de funciones de acotación asintóticas

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Ejercicio 1

¿Cuántos pasos realizan los algoritmos anteriores hasta encontrar una respuesta?

- *Recursivo (puro).*
- *Recursivo con almacén.*
- *Iterativo.*

Diferencias entre los algoritmos

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Algoritmo recursivo (puro), recursivo con almacén e iterativo

¿Por qué existe esta diferencia de costes para resolver el mismo problema?

Diferencias entre los algoritmos

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Algoritmo recursivo (puro), recursivo con almacén e iterativo

¿Por qué existe esta diferencia de costes para resolver el mismo problema?

La respuesta a esta pregunta es la base de este tema, y la esencia de la **programación dinámica**.

Índice

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

- 1 Introducción
- 2 Programación dinámica recursiva
- 3 Programación dinámica iterativa
- 4 La mayor subsecuencia común
- 5 El problema de la mochila
- 6 Respuestas a los ejercicios

PD recursiva

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

La **programación dinámica recursiva** es un diseño fácil de plantear para resolver un problema. Se hace uso de la recursividad del sistema, se suelen repetir llamadas a los mismos subproblemas (se repiten operaciones) y generalmente desemboca en costes temporales exponenciales.

El Nim (sin repetir jugada)

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Vamos a considerar una variante de este juego en la que no se permite retirar el mismo número de fichas que retiró el contrario en su jugada anterior. Por tanto, se pierde tanto si no quedan fichas en la mesa como si se encuentra una pero el jugador contrario retiró una en su último turno.

Ejercicio 2

Diseña una función recursiva booleana $best(n, m)$ para el Nim de dos jugadores y $M > 2$ que sea cierta si existe una estrategia ganadora para el jugador que encuentra n fichas sobre la mesa si el jugador anterior retiró m fichas (y falsa en caso contrario).

Ejercicio 3

Compara el tiempo de cálculo de la función anterior para $N = 50$ y para $N = 60$ con $M = 3$. ¿Para qué valores de N es útil el algoritmo?

Ejercicio 4

*¿Cuál es la **causa del crecimiento exponencial** en el **coste temporal** y en función del tamaño de la entrada N ?*

Ayuda: dibuja el árbol de llamadas recursivas que se genera para $\text{best}(6, 0)$ y $M = 3$.

Ejercicio 5

Modifica el algoritmo para evitar, en la medida de lo posible, el aumento de llamadas a la función best.

PD recursiva

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

La esencia de la programación dinámica es utilizar un **almacén** para que el programa recursivo guarde resultados intermedios que pueden ser reutilizados para acelerar el cómputo. A esta técnica la llamaremos **programación dinámica recursiva con almacén** (en inglés, *memoization*).

Ejercicio 6

¿Cuál es el valor límite de N para el que se puede calcular el resultado con la nueva versión del algoritmo?

Ejercicio 7

¿Cuál es el coste temporal del nuevo programa recursivo con almacén en términos de $O(f(n))$ en función de N y M ?

Ejercicio 8

Implementa en Java un programa que juegue al Nim. Para ello, modifica la función para que el valor de retorno sea el número de fichas que debe retirarse para ganar o un valor negativo si no existe estrategia ganadora.

Índice

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

- 1 Introducción
- 2 Programación dinámica recursiva
- 3 Programación dinámica iterativa
- 4 La mayor subsecuencia común
- 5 El problema de la mochila
- 6 Respuestas a los ejercicios

PD iterativa

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

La **programación dinámica iterativa** suele resultar más rápida y además evita el coste de memoria asociado a las llamadas recursivas. Por contra, a veces calcula más posiciones del almacén de las necesarias, por lo que no es siempre más rápida que la programación dinámica recursiva con almacén.

Ejercicio 9

Implementa una versión iterativa que evite el desbordamiento de la pila de llamadas recursivas en el algoritmo para el Nim y compara la velocidad y el número de elementos del almacén calculados. Observación: la jugada inicial corresponde a la llamada con parámetro $m = 0$.

Ejercicio 10

Transforma el algoritmo iterativo anterior en otro que utilice un almacén de tamaño independiente de N .

Ejercicio 11

¿Cómo podemos en general evitar que un algoritmo de programación dinámica iterativa calcule más elementos de los que necesita la versión recursiva?

Índice

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

- 1 Introducción
- 2 Programación dinámica recursiva
- 3 Programación dinámica iterativa

- 4 La mayor subsecuencia común
- 5 El problema de la mochila
- 6 Respuestas a los ejercicios

Mayor subsecuencia común

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

La orden `diff` de Unix compara las líneas de dos ficheros A y B e informa sobre las diferencias encontradas entre ambos. Para ello aplica una función de dispersión (“hashing”) que transforma cada línea en un único entero. Posteriormente busca la mayor subsecuencia común a las secuencias obtenidas, esto es, qué líneas deben borrarse de cada uno para obtener dos ficheros idénticos.

Ejercicio 12

¿Qué operaciones deben realizarse para transformar la secuencia “nuclear” en “unclear”? ¿Cuál es la longitud de la mayor subsecuencia común?

Ejercicio 13

¿Cuál es la m.s.c. de las cadenas “algoritmia” y “avanzada”? Une mediante líneas cada par de letras iguales que forma parte de la m.s.c.

Ejercicio 14

Analiza en qué casos es trivial comparar dos ficheros por líneas. Escribe una fórmula recursiva para los casos más complejos pensando en qué posibilidades hay

- 1 *cuando las dos últimas líneas son distintas y*
- 2 *cuando son iguales.*

Ejercicio 15

*Calcula la mayor subsecuencia común de las dos cadenas siguientes:
01001100011100001111 y 11110011011000100011.*

Ejercicio 16

¿Cuántas casillas de la tabla se calculan cuando se aplica programación dinámica recursiva al ejemplo anterior? ¿Y en el caso iterativo?

Ejercicio 17

¿Cuál es el coste temporal (en el peor caso) de calcular la mayor subsecuencia común de dos cadenas A y B usando programación dinámica? ¿Y el de obtener la de tres cadenas A , B y C ?

Índice

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

- 1 Introducción
- 2 Programación dinámica recursiva
- 3 Programación dinámica iterativa
- 4 La mayor subsecuencia común
- 5 El problema de la mochila
- 6 Respuestas a los ejercicios

Problema de la mochila

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

El problema de la mochila discreto se plantea de la siguiente forma: dados N objetos de pesos w_1, \dots, w_N y valores v_1, \dots, v_N , calcúlese el valor máximo de los objetos que se pueden transportar en una mochila de capacidad M .

Ejercicio 18

Escribe una fórmula recursiva para el valor máximo $f(n, m)$ que se puede transportar en la mochila sin sobrepasar el peso m tomando sólo objetos entre los n primeros.

Ejercicio 19

El problema de la mochila discreto es un ejemplo de problema de complejidad asintótica exponencial al que se puede aplicar programación dinámica. Si los pesos son enteros, el coste temporal está en $O(NM)$ siendo N el número de objetos y M la capacidad de la mochila. ¿Significa lo anterior que hemos encontrado un algoritmo polinómico para un problema exponencial?

Ejercicio 20

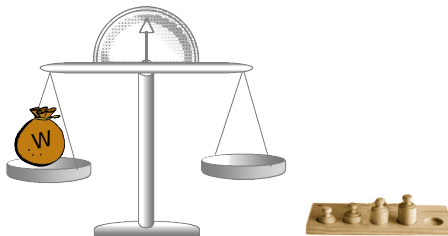
Calcula una solución óptima y dibuja las posiciones de la tabla calculadas en el caso con $N = 10$, $M = 40$ en el que pesos y valores coinciden y son los siguientes: 23, 6, 35, 33, 15, 26, 12, 9, 21 y 3.

Ejercicio 21

Explica cómo evitar el coste proporcional a NM cuando $2^N \ll M$.

Problema

Escribe un programa para determinar el número mínimo de pesas que permitirá determinar con una balanza si un objeto pesa exactamente W gramos si se dispone de un conjunto de N pesas cuyos pesos son w_1, \dots, w_N (no necesariamente distintos). Todas las pesas pueden utilizarse como contrapeso, y como tara, es decir, pueden colocarse en cualquiera de los dos platillos.



Índice

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

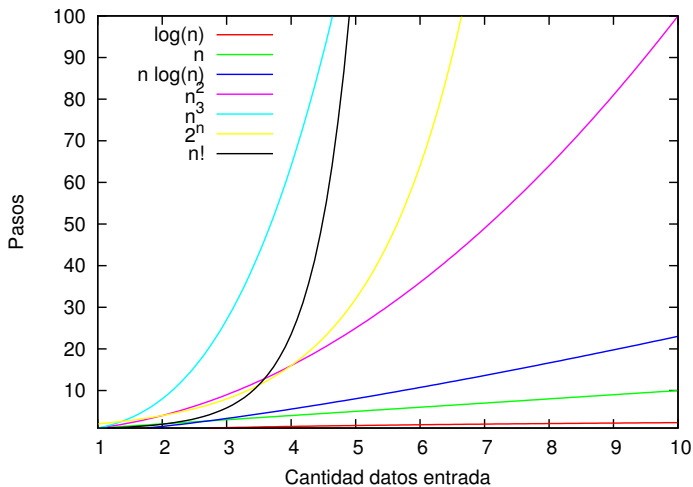
- 1 Introducción
- 2 Programación dinámica recursiva
- 3 Programación dinámica iterativa
- 4 La mayor subsecuencia común
- 5 El problema de la mochila
- 6 Respuestas a los ejercicios

Ejercicio 1 Las funciones asintóticas representan el orden de la función $f(n)$. Consideremos 3 de los principales tipos de acotación asintótica para una función que represente el número de pasos que efectúa el algoritmo hasta encontrar una respuesta:

- $O(f(n))$: Big O, Omicron. Peor de los casos.
- $\Omega(f(n))$: Omega. Mejor de los casos.
- $\Theta(f(n))$: Theta. Cuando coinciden los anteriores $O(f(n)) = \Omega(f(n))$

Función asintótica	Orden
$O(1)$	Constante
$O(\log(n))$	Logarítmico
$O(n)$	Lineal
$O(n \log(n))$	Cuasi-lineal
$O(n^2)$	Cuadrático
$O(n^3)$	Cúbico
$O(n^a)$	En general, polinómico
$O(a^n)$	Exponencial ($2^n, 3^n, \dots$)
$O(n!)$	Factorial

Funciones asintóticas. Representación gráfica.



El problema del Nim lo podemos acotar con:

- Recursivo (puro) : $O(M^N)$
- Recursivo con almacén: $O(MN)$
- Iterativo : $O(MN)$

Respuesta

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Ejercicio 2 Obviamente, $\text{best}(n, m) = \text{false}$ si $n = 0$ o si $n = 1$ y $m = 1$. En cambio, si $n = 1$ y $m \neq 1$ las reglas permiten al jugador retirar 1 ficha y ganar. En general, si se retiran k fichas quedarán $n - k$ y el contrario ganará si $\text{best}(n - k, k)$ es positivo y juega correctamente. Si para alguno de los valores de k permitidos $\text{best}(n - k, k)$ es negativo tenemos una estrategia ganadora, retirar k fichas, contra la que el rival no tiene respuesta ganadora. Por tanto, la función $\text{best}(n, m)$ se puede expresar como:

$best(n, m) =$

$$\begin{cases} falso & \text{si } (n = 0) \vee (m = n = 1) \\ \bigvee_{1 \leq k \leq \min(n, M): k \neq m} \neg best(n - k, k) & \text{en otro caso} \end{cases}$$

La función anterior la podríamos programar como:

```
public boolean best (int n, int m) {  
    boolean res=false;  
    if ( n==0 || (n==1 && m==1)){  
        res = false;  
    } else {  
        for ( int k = 1; k <= Math.min(n, M); ++k )  
            if ( k != m && !best(n - k, k) )  
                res = true;  
    }  
    return res;  
}
```

Respuesta

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

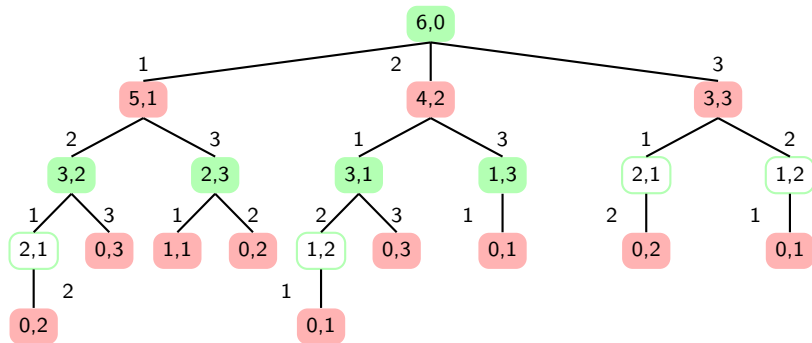
El problema
de la mochila

Respuestas a
los ejercicios

Ejercicio 3 El algoritmo es 50 veces más lento en el segundo caso y, por tanto, inútil para valores de n que se acerquen a 100.



Ejercicio 4



En el árbol se observa que se generan dos llamadas idénticas a la función con valores $best(1,2)$ y $best(2,1)$. Si n es mayor, el número de repeticiones crece enormemente.

Respuesta

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Ejercicio 5 Una solución sencilla consiste en utilizar un almacén A para guardar los resultados obtenidos. De esta forma, las llamadas a la función f con parámetros idénticos se limitan a consultar el contenido almacenado en la posición (n, m) , que se calcula sólo la primera vez. La implementación puede consultarse en la página siguiente.

```
Boolean[] [] A = new Boolean[N+1][M+1];
public boolean best (int n, int m) {
    boolean res = false;
    if ( A[n][m] == null ) {
        if ( n==0 || (n==1 && m==1)){
            res = false;
        } else {
            for ( int k = 1; k <= Math.min(n, M); ++k )
                if ( k != m && !best(n - k, k) )
                    res = true;
        }
        A[n][m] = res;
    } else {
        res = A[n][m];
    }
    return res;
}
```

Respuesta

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Ejercicio 6 El valor dependerá del ordenador que se esté usando para realizar el cálculo, pero típicamente será del orden de los centenares de miles. La causa de este límite es el desbordamiento de la pila de llamadas recursivas.



Ejercicio 7 La función principal llamará una vez a la función `best`.

Dado que todo elemento del almacén A es distinto de cero después de la llamada a `best(n, m)` y que siempre se llama a esta función con n en el rango $[0, N]$ y m en $[0, M]$, las instrucciones del bloque condicional no pueden ser ejecutado más de $(N + 1)(M + 1)$ veces (una por cada vez que es cierta la condición). Además, como este bloque contiene M llamadas recursivas, no puede haber en total más de $(N + 1)(M + 1)M$ llamadas recursivas a la función `best`.

Es decir, el coste temporal es de $O(NM^2)$.

El coste espacial o memoria utilizada por el algoritmo es proporcional a N pues utiliza una matriz de tamaño $(N + 1)M$ y pero la pila de llamadas recursivas no necesita almacenar más de M llamadas.

Respuesta

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Ejercicio 8 Se resolverá en el laboratorio.



Ejercicio 9 Con un algoritmo como el siguiente, la velocidad es significativamente mayor (del orden de 4 veces para $N = 3$). Además permite calcular el resultado con un coste de tiempo y memoria viables hasta valores de m mucho mayores. La versión iterativa calcula todas las posiciones del almacén anteriores a la requerida, mientras que la versión recursiva se ahorra algunas de ellas (muy pocas) que no necesita.

```
public boolean best (int n0, int m0) {  
    boolean[] [] A = new boolean[N+1][M+1];  
    for ( int n = 0; n <= N; ++n )  
        for ( int m = 0; m <= M; ++m ) {  
            A[n][m] = false;  
            for ( int k = 1; k <= Math.min(n, M); ++k ){  
                if ( k != m && !A[n-k][k] ) {  
                    A[n][m] = true;  
                }  
            }  
        }  
    }  
    return A[n0][m0];  
}
```

Ejercicio 10 Dado que para el cálculo de la fila n se requieren, como mucho, las filas $n - 1, n - 2, \dots, n - M$, podemos usar una matriz de tamaño $(M + 1) \times (M + 1)$ si sustituimos en el programa todos los accesos a la posición (i, j) de la matriz por accesos a la posición $(i \bmod (M + 1), j)$.

Ejercicio 11 Una forma simple es modificar el programa para que haga dos pasadas por el almacén (a costa de duplicar aproximadamente el tiempo de cálculo). En la primera se determina, en orden inverso al de llenado, qué posiciones del almacén se van a necesitar. La segunda calcula las posiciones marcadas en el orden que garantiza que nunca se consultan posiciones no calculadas.



1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Ejercicio 12 Tenemos dos posibilidades: borrar la u de ambas cadenas o borrar la n de ambas. En ambos casos, la subsecuencia común (“nclear” o “uclear”) es de longitud 6.



Respuesta

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Ejercicio 13 La mayor subsecuencia común es “aa”. Hay tres formas de conseguir esta subsecuencia. Una de ellas, por ejemplo, es:

```
a l g o r i t m i a
|                               /
a v a n z a d a
```



Ejercicio 14 Sean dos ficheros $A = A_1A_2 \dots A_N$ y $B = B_1B_2 \dots B_M$.

La solución es trivial si alguno de los ficheros está vacío, en cuyo caso la m.s.c. tiene longitud 0. Por tanto, si representamos mediante $f(i, j)$ el tamaño de la mayor subsecuencia común a las primeras i líneas de A y las primeras j de B , podemos escribir $f(i, 0) = f(0, j) = 0$.

Si la línea A_i es distinta de la B_j , entonces una de las dos líneas no puede ser parte de la mayor subsecuencia común y podemos escribir $f(i, j) = \max(f(i-1, j), f(i, j-1))$. En caso de que $A_i = B_j$, la pareja (i, j) puede ser parte de la mayor subsecuencia común. Un razonamiento simple nos permite deducir que no puede haber subsecuencias comunes mayores (aunque sí iguales) que la que incluye esta pareja, así que $f(i, j) = 1 + f(i-1, j-1)$.

En resumen:

$$f(i, j) = \begin{cases} 0 & \text{si } i = 0 \vee j = 0 \\ 1 + f(i - 1, j - 1) & \text{si } A_i = B_j \\ \max(f(i - 1, j), f(i, j - 1)) & \text{en caso contrario} \end{cases}$$

Respuesta

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Ejercicio 15 Una de las secuencias más largas es 01011000100011

A: 01 001100011100001111

B: 111100110 110001 00011

m.s.c: 01 0 110001 00011



Respuesta

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Ejercicio 16 El programa recursivo calcula 252 casillas frente a 400 del iterativo. Sin programación dinámica, se realizan más de 200.000 llamadas.



Respuesta

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Ejercicio 17 En el peor caso, los costes están en $O(|A||B|)$ y $O(|A||B||C|)$ respectivamente.



Respuesta

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Ejercicio 18 El problema tiene una solución trivial si $n = 0$ o si $m = 0$. En ese caso (suponiendo valores y pesos positivos), $f(n, m) = 0$. En general, el valor de $f(n, m)$ depende de qué decisión se tome sobre el objeto n . Si éste se incluye en la mochila (lo que es posible sólo si $w_n \leq m$), entonces $f(n, m) = v_n + f(n - 1, m - w_n)$. En cambio, si no se toma el objeto n , entonces $f(n, m) = f(n - 1, m)$.

Como a priori no es posible saber cuál de estas opciones es la mejor, debemos comparar ambas y la fórmula recursiva queda:

$$f(n, m) = \begin{cases} 0 & \text{si } n = 0 \vee m = 0 \\ \max(f(n-1, m), v_n + f(n-1, m - w_n)) & \text{si } w_n \leq m \\ f(n-1, m) & \text{en el resto de los casos} \end{cases}$$

Ejercicio 19 La ambigüedad surge de haber expresado la complejidad en términos de dos parámetros (N y M) y no en función del tamaño L de la entrada. Si bien el espacio requerido para codificar los tamaños y valores de N objetos es proporcional a N , el espacio requerido para codificar el valor de M es proporcional a $\log(M)$. Por ello, la complejidad en función del tamaño de la entrada L es proporcional a $NM \simeq L \exp(L)$.

Respuesta

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Ejercicio 20

En este caso, el valor óptimo es 39 (por ejemplo, $6 + 12 + 21$ o $6 + 33$).

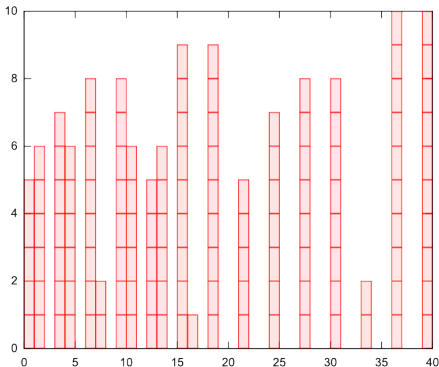


Figura: Parte calculada de la tabla mediante PDR con almacén.

Índice de ocupación del almacén es un 28,28 % ($130/(41 \times 11) = 0,2882$).

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

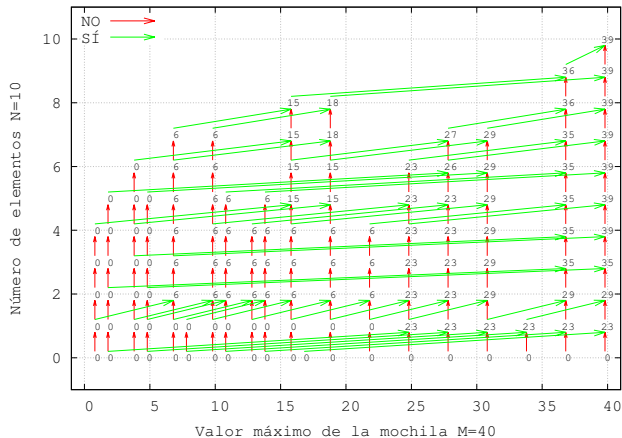


Figura: Soluciones y relación entre problemas (PDR con almacén).

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

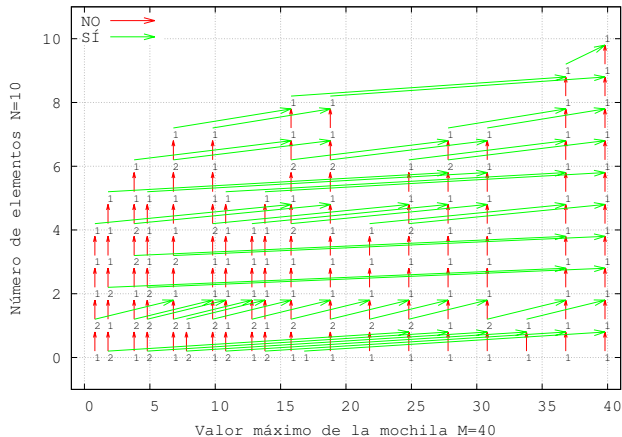


Figura: Repeticiones y relación entre problemas (PDR con almacén).

Índice de ocupación del almacén es un 100 %.

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

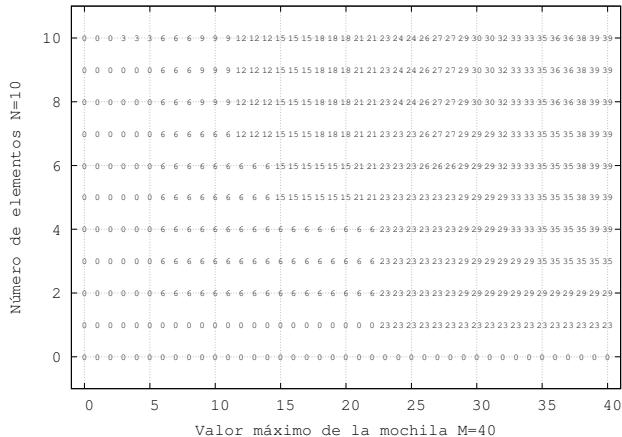


Figura: Soluciones PDI.

Respuesta

1-PD

Juan R. Rico

Introducción

Programación
dinámica
recursiva

Programación
dinámica
iterativa

La mayor
subsecuencia
común

El problema
de la mochila

Respuestas a
los ejercicios

Ejercicio 21 En el caso $2^N \ll M$, el mayor coste del programa recursivo se produce en la creación del almacén, ya que el número de posiciones calculadas no puede ser mayor que 2^N . Para evitar este sobrecoste se debe utilizar un contenedor que permita consultar si el elemento está almacenado o no sin necesidad de crearlo.

←P