AYED II
Lautaro Gastón Peralta

Práctico 3.4

1. Dar una definición de la función cambio utilizando la técnica de programación dinámica a partir de la siguiente definición recursiva (backtracking):

$$cambio(i,j) = \begin{cases} 0 & j = 0 \\ \infty & j > 0 \wedge i = 0 \\ \min_{q \in \{0,1,\ldots,j \div d_i\}}(q + cambio(i-1, j - q * d_i)) & j > 0 \wedge i > 0 \end{cases}$$

```
fun cambio (d : array [1..N] of nat, k : nat) ret r : nat
        var cam[0..n,0..k] of nat
        var temp : nat
        var q : nat
        for i:=0 to n do cam[i,0]:=0
        for j:=1 to k do camb[0,j]:=infinito
        for i:=1 to n do
                for j:=1 to k do
                        temp := cam[i,j]
                        for  q:=0 to k/d[i] do
                                temp := min(temp, temp+cam[i-1,j-q*d[i]])
                        od
                        cam[i,j]:=temp
                od
        od
end fun
```

2. Para el ejercicio anterior, ¿es posible completar la tabla de valores "de abajo hacia arriba"? ¿Y "de derecha a izquierda"? En caso afirmativo, reescribir el programa. En caso negativo, justificar.

De arriba hacia abajo.

3. Dar una definición de la función cambio utilizando la técnica de programación dinámica a partir de cada una de las siguientes definiciones recursivas (backtracking):

(a)

$$cambio(i,j) = \begin{cases} 0 & j = 0 \\ 1 + \min_{i' \in \{1,2,\ldots,i \mid d_{i'} \leq j\}}(cambio(i', j - d_{i'})) & j > 0 \end{cases}$$

(b)

$$cambio(i,j) = \begin{cases} 0 & j = 0 \\ \infty & j > 0 \wedge i = n \\ cambio(i+1, j) & d_i > j > 0 \wedge i < n \\ \min(cambio(i+1, j), 1 + cambio(i, j - d_i)) & j \geq d_i > 0 \wedge i < n \end{cases}$$

a)

```
fun cambio (d : array [1..N] of nat, k : nat) ret r : nat
        var cam[0..n,0..k] of nat
        var min : nat
        for i:=0 to n do cam[i,0]:=0 od
        for i:=0 to n do
                for j:=1 to k do
                        min:=infinito
```

```
                    for i` to n do
                        if d[i]<=j && min > (cam[i`,j-d[i]]) then
                            min:= (cam[i`,j-d[i]])
                        fi
                        cam[i,j]:= 1 + min
                    od
                od
        od
end fun

b)
fun cambio (d : array[1...N] of nat, k : nat) ret r : nat
        var cam : array [0..n,0..k] of nat
        for i:=0 to n do cam[i,0] := 0 od
        for j:=1 to k do cam[n,j] := infinito od
        for i:=1 to n do
                for j:=1 to k do
`                   if d[i] > j AND i < n then
                        cam[i,j]:= cam[i+1,j]
                    else
                        cam[i,j]:=cam[i+1,j] ‘min’ 1 + cam[i,j-d[i]]
                    fi
                od
        od
        r:=cam[n,k]
end fun
```