Algoritmos
TAREA 4

Israel Sandoval Grajeda y Fausto Salazar Mora

5 de diciembre de 2013

Problemas Programación dinámica

Representación de un número n como suma de m enteros no negativos

Como en los ejercicios anteriores de programación dinámica, se procede a realizar la versión recursiva.

Algorithm 1 Calculo de numero de representaciones version recursiva

Require: Numero n a representar; cantidad m de números con los que se quiere representar a n con sumas
Ensure: Cantidad r de posibles sumas de los m números para representar a n

```
1: sumas(n,m)
2: if n==1 then
3:     return m
4: end if
5: if m==1 then
6:     return 1
7: end if
8: r=sumas(n-1,m)+sumas(n,m-1)
9: return r
```

El análisis fue el siguiente: Cuando se tiene al 1 (n=1) este tiene m formas de ser representado, como en el ejemplo propuesto en el enunciado de la presente tarea.

Cuando se tiene cualquier número y se requiere su representación usando un solo número, entonces él mismo es su representación y es única.

La observación mas importante se encuentra en la línea 8 y es la siguiente: Para contar las representaciones se toman las que no incluyan a n con m numeros, y las que si lo incluyan pero con un elemento menos (porque ya se incluye n).

La ecuación funcional asociada es:

$$Sumas(n,m)\begin{cases} 1 \text{ Si } m=1 \\ m \text{ Si } n=1 \\ sumas(n-1,m)+sumas(n,m-1) \text{en otro caso} \end{cases}$$

Se procede al algoritmo secuencial:

Algorithm 2 Calculo de numero de representaciones version Programación dinámica

Require: Numero n a representar; cantidad m de números con los que se quiere representar a n con sumas
Ensure: Cantidad de posibles sumas de los m números para representar a n y una matriz de $n \times m$ como memoria

```
1: sumas(n,m)
2: for r=1 hasta n-1 do
3:     mem[r][1]=1
4: end for
5: for s=1 hasta m-1 do
6:     mem[1][s]=s
7: end for
8: for j=1 hasta n-1 do
9:     for i=1 hasta m-1 do
10:        mem[i][j]=mem[i-1][j]+mem[i][j-1]
11:    end for
12: end for
13: return mem[m]][n]
```

Análisis de complejidad

El algoritmo contiene 3 ciclos For.

1. Lineas 2-4: tiene complejidad O(n)
2. Lineas 5-7: tiene complejidad O(m)
3. Lineas 8-12: Ya que dentro de un ciclo de 0 hasta n existe otro ciclo de 1 hasta m, el trozo de código completo tiene complejidad O(mn).

Por lo tanto, la complejidad del algoritmo completo es O(mn)

Notesé que todas las operaciones son constantes O(1) pero hay ciclos anidados, el primero va de 1 a m el segundo de n a 0 y el tercero de 0 a lo que sea que lleve el segundo ciclo en términos generales esto quiere decir que tengo O(m *)

Análisis de correctez

Las lineas 2 a 4 cumplen con el primer criterio de la ecuación funcional; las lineas 5 a 7 con el segundo criterio y de la 8 a la 11 con el tercero.

Se identificó que las soluciones para un elemento de la matriz provenían únicamente de los lugares adyacentes izquierdo y superior, por lo que se realizó un recorrido por renglones para asegurar los cálculos.

El algoritmo es correcto pues en todo momento el valor de la celda que se busca calcular tiene sus orígenes en una posición de la matriz que ya está calculada en la columna anterior.