Algoritmos
TAREA 4

Israel Sandoval Grajeda y Fausto Salazar Mora

5 de diciembre de 2013

Problemas Programación dinámica

Formas de dar cambio

Caso recursivo

Algorithm 1 Formas de dar cambio recursivo

```
1: S = {1, 5, 10, 25, 50} // los centavos
2: Cambio(x,S)
3: n = longitud(S)
4: Sm = S sin S[n-1]
5: if x=0 then //Cambio exacto
6:     return 1
7: else if x<0 then // Me pasé de cambio
8:     return 0
9: else if n=0 then
10:    return 0
11: end if
12: return Cambio(x-S[n-1],S) + Cambio(x,Sm)
```

La recursión consiste en tomar las combinaciones posibles del conjunto que toma al menos una moneda de mayor denominación y el otro conjunto donde se descarta que la moneda de mayor denominación. Para la memoización considerar una matriz bidimensional donde los renglones representan el valor de x y las columnas las diferentes denominaciones de monedas restantes en mi conjunto S entonces pues Las ecuaciones funcionales obtenidas en base al ejemplo recursivo son:

$$Opt(x,n)\begin{cases} 1 \text{ Si } x=0 \\ 0 \text{ Si } n=0 \text{ o } x<0 \\ Opt(x-S(n),n)+Opt(x,n-1) \text{ en otro caso} \end{cases}$$

El valor del elemento M[x,n] de la matriz depende de alguno de los elementos que se encuentran hacia arriba de x (parte superior de la columna) y el elemento a la izquierda de n (mismo renglón en la columna anterior).

El algoritmo de programación dinámica queda entonces de la siguiente manera:

Algorithm 2 Formas de dar cambio con Prog. Danámica

```
1: S = {1, 5, 10, 25, 50} // los centavos
2: Cambio(x,S)
3: n = longitud(S)
4: Matriz [x+1][n+1] //Creo la matriz del tamaño x+1*n+1"
5: for i = 0 i, <= x, i=i+1 do
6:     Matriz [i,0] = 1
7: end for
8: for i = 0, i <=n, i=i+1 do
9:     Matrix [0,i] = 0
10: end for
11: for i=1, i <=x, i=i+1 do
12:     for j=1, j<=n, j=j+1 do
13:         if i-S[j] >= 0 then
14:             Matriz[i,j] = Matriz[i-S[j]] + Matriz[i,j-1]
15:         else
16:             Matriz[i,j] = Matriz[i,j-1]
17:         end if
18:     end for
19: end for
20: return Matrix[x,n]
```

Análisis de complejidad

Notesé que todas las operaciones son constantes O(1) pero hay un ciclo anidado dentro de otro así que para cada valor entre 1 y x se cicla entre 1 y n elementos que contenga S que es la cantidad de monedas de diferente denominación a utilizar así pues la complejidad es de O(xn).

Análisis de correctez

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.