Soluciones - práctico 9

Programación Dinámica

EJERCICIO 5 (R)

Una partición de un entero positivo *n*, es una secuencia de enteros positivos $m_1, ..., m_k$ tal que:

$$m_1 + \quad + m_k = n$$

Suponga que se tiene una tabla de pesos asociada a los enteros positivos del 1 al *n*, siendo cada peso un número real. El peso total en una partición $m_1, m_2, ..., m_k$ es la suma de los pesos asociados a cada entero m_j por el valor de m_j o sea

$$\sum_{i=1}^{k} m_i \quad \text{tiene como peso asociado} \quad \sum_{i=1}^{k} (p_{m_i} \cdot m_i)$$

Se define como la mejor partición de *n* aquella cuyo peso total es mínimo.

Se pide:
a) escribir un algoritmo utilizando *Programación Dinámica* para calcular el peso total de la mejor partición.
b) estudiar el orden de su algoritmo.

Solución:

a) La mejor partición es aquella que minimiza la suma de los pesos.

A continuación se muestra un ejemplo para n = 5:

Dado el vector de pesos **p[1, 3, 2, 5, 100]**, para el rango de números: **1..5** se tienen las siguientes particiones del número **5**:

Partición	Costo
5	5*100 = **500**
4 1	4*5 + 1*1 = **21**
3 2	3*2 + 2*3 = **12**
3 1 1	3*2 + 1*1 + 1*1 = **8**
2 2 1	2*3 + 2*3 + 1*1 = **13**
2 1 1 1	2*3 + 1*1 + 1*1 + 1*1 = **9**
1 1 1 1 1	1*1 + 1*1 + 1*1 + 1*1 + 1*1 = **5**

Obteniendo que la mejor partición es la que tiene costo **5**.

De lo anterior se puede llegar a:

Partición	Costo
5	**5** * p (**5**)
Partición (4), 1	MejorPartición (**4**) + **1** * p [**1**]
Partición (3), 2	MejorPartición (**3**) + **2** * p [**2**]
Partición (2), 3	MejorPartición (**2**) + **3** * p [**3**]
Partición (1), 4	MejorPartición (**1**) + **4** * p [**4**]

Se puede observar que*: MejorPartición*(4) tiene un costo de 4 unidades y al agregarle el entero 1 que suma un costo de una unidad, se llega a la conclusión de que *MejorPartición*(5) tiene un costo de 5 unidades.

Lo anterior puede resumirse en la siguiente fórmula:

MejorParticion(n) = min{1.*p*[1] + *MejorParticion*(n-1), 2.*p*[2] + *MejorParticion*(n-2), …, n*p*[n] + *MejorParticion*(0)}

Luego de ver este ejemplo se presenta la solución al problema

La recursión puede escribirse de la siguiente manera:

Caso base*:*

MejorParticion(0) = 0

Paso recursivo:

MejorParticion(i) = min{k.p[k] + MejorParticion(i – k)} con $1 \leq k \leq i$ siendo *i* el entero que se desea particionar y *k* todos los enteros entre 1 e *i* particionados hasta el momento.

La solución se obtiene con ***MejorParticion*(n)**.

Esta recurrencia retorna el valor óptimo ya que surge de probar todas las posibles soluciones y elegir la mejor. En la misma, se parte con la hipótesis de que las soluciones antes calculadas son óptimas. Resulta entonces, que la mejor solución debe estar entre las soluciones antes calculadas luego de sumarle el entero que falta (multiplicado por su costo) para alcanzar el valor a particionar.

Si se considera que la solución óptima para *n* utiliza el entero *x*, entonces, sabiendo que la partición óptima utiliza *x* se obtiene que el primer sumando tiene que aparecer en *MejorParticion*(*n*):

MejorParticion(n) = x.p[x] + MejorParticion(n – x) con $1 \leq x \leq n$

Además, por hipótesis inductiva, se parte de que *MejorParticion*(*n-x*) es óptimo , entonces se llega a que la solución planteada también debe serlo.

Programación Dinámica sugiere el uso de memoria para realizar este cálculo, por lo tanto se puede escribir el algoritmo de la siguiente manera:

```
void mejorParticion (int n, int* p, int* mp){
  mp[0] = 0;
  for (int i = 1; i <= n; i++){
        mp[i] = p[i] * i;
        for (int k = 1; k < i; k++){
              mp[i] = min (mp[i], (k * p[k]) + mp[i-k]);
        }
  }
}
```

Una forma de probar este algoritmo es la siguiente:

```
#include <limits.h>
#include <stdio.h>

int main (){
       const int n = ...;
       int* mp = new int[n + 1];
```

```
        int p[n+1] = .........;

        mejorParticion (n, p, mp);

        printf ("mejor particion de %d = %d\n", n, mp[n]);
        delete [] mp;
        return 0;
}
```

b) El cálculo del orden de ejecución del algoritmo se puede obtener de la siguiente manera:

$$MejorParticion(n) = \sum_{i=1}^{n} \sum_{k=1}^{i-1} 1 = \sum_{i=1}^{n} i - 1 = -n + \sum_{i=1}^{n} i = -n + \frac{n(n+1)}{2} = \frac{n(n-1)}{2}$$

lo que da como resultado $O(n^2)$