

PRÁCTICA 4

Programación dinámica

Álvaro Maximino Linares Herrera



Notas:

En esta práctica el código voraz es una implementación del pseudocódigo de las transparencias del tema 3, archivo **monedasav.cpp**.

El código dinámico es una implementación del pseudocódigo del libro Técnicas de Diseños de Algoritmos, sus autores son Rosa Guerequeta García y Antonio Vallecillo Moreno. Archivo **monedaspd.cpp**.

PROBLEMA DEL CAMBIO DE MONEDAS:

El problema que nos planteamos en esta 4º práctica es el cambio de monedas y como es más eficiente en programación dinámica que en algoritmos voraces.

Lo haré de una forma sencilla, la cantidad a devolver en monedas será 32, y como en el enunciado de la práctica se nos dice que deben de ser 5 tipos de monedas distintos, que serán 1, 5, 10, 12, 25, el resultado óptimo de la devolución será 3 monedas, 2 de 10 y una de 12 ($10+10+12=32$).

El problema de algoritmos voraces es que cuando le metemos la moneda de 12 empieza a dar soluciones que no son la óptima, de hecho una ejecución suya nos daría el siguiente resultado:

2 de 1

1 de 5

1 de 25

que como vemos también nos da 32, ($1+1+5+25=32$) pero en lugar de usar 3 monedas como en programación dinámica usa 4 monedas.

A continuación presentaré el código usado, primero el algoritmo voraz y el segundo el de programación dinámica.

Código Voraz:

```
void CambioVoraz(int cantidad, int valores[], int *&solucion, int tam){
    int devuelta=0;
    int h=tam;
    while(devuelta!=cantidad){
        while(valores[h]>(cantidad-devuelta) && (h>0)){
            h--;
            solucion[h]=((cantidad-devuelta)/valores[h]);
            devuelta= devuelta + (valores[h]*solucion[h]);
        }
    }
}
```

Como vemos a la función CambioVoraz le pasamos una serie de parámetros:

- Un entero cantidad: Corresponde a la cantidad a devolver.
- Un vector de enteros valores: Corresponde a los distintos valores de las monedas.
- Un puntero de enteros solución: Lo pasamos por referencia ya que lo modificaremos a lo largo de la ejecución del problema y en la que almacenaremos la cantidad de monedas de cada tipo que usaremos.

- Un entero tam: Se corresponde con el tamaño del vector valores, los distintos tipos de monedas que hay (en nuestro caso porque lo pide el problema serán 5).

El funcionamiento es bien sencillo, declaramos dos enteros, el primero “devuelta” que indica la cantidad devuelta que llevamos, y el segundo h que lo igualamos al tamaño del vector de valores.

Ahora con dos bucles while vamos a recorrer el vector de valores al revés para así obtener la devolución con los valores de las monedas más altos, hasta que la cantidad devuelta sea igual a la cantidad a devolver.

En el segundo bucle while vamos a ir recorriendo el vector valores y cada vez que podemos usar una moneda para dar el cambio la meteremos en el vector solución tantas veces como sea posible.

Al final del programa tendremos en el vector solución la cantidad de monedas que usaremos.

Ejemplo de una ejecución con los parámetros descritos en la presentación de este problema



```
alvaro@alvaro-Inspiron-1545: ~/Escritorio/practica 4
alvaro@alvaro-Inspiron-1545:~/Escritorio/practica 4$ ./av
Cantidad: 32
2 de 1
1 de 5
1 de 25
alvaro@alvaro-Inspiron-1545:~/Escritorio/practica 4$
```

Como se puede apreciar muestra la cantidad a devolver, 32, y por orden de menor a mayor las monedas que se cogen, el trozo de código para mostrarlas será este:

```
for(int i=0;i<tam; i++){
    if(solucion[i]!=0)
        cout << solucion[i] << " de " << valores[i] << endl;
}
```

Como se puede apreciar solucion muestra la cantidad de monedas de un tipo cogidas y valores los valores de las monedas, uso el if(solucion[i]!=0) para que no muestre monedas que no se usan.

Código Dinámico:

En programación dinámica trabajaremos con matrices, por lo cual cuando tenga que crear una matriz nueva usaré la siguiente función:

```
int **Crear(int fils, int cols){
    int **matriz;
```

```

matriz=new int *[fils];
for(int i=0; i<fils; i++)
    matriz[i]=new int [cols];
return matriz;
}

```

Es una función muy sencilla extraída de un problema de la asignatura de Metodología de la Programación de 1º.

Por otro lado el código del CambioDinamico será el siguiente:

```

int CambioDinamico(int devolucion, int monedas[], int numero_monedas){

```

```

    int **cambio=Crear(numero_monedas+1, devolucion+1);
    int **aux=Crear(numero_monedas+1, devolucion+1);
    int i, j;
    for(i=0; i<numero_monedas+1; i++)
        for(j=0; j<devolucion+1; j++)
            aux[i][j]=0;

    for(i=0; i<numero_monedas; i++)
        cambio[i][0]=0;

    for(i=1; i<=devolucion; i++)
        cambio[0][i]=100000;

    for(i=1; i<=numero_monedas; i++)
        for(j=1; j<=devolucion; j++){

            if(monedas[i-1]>j)
                cambio[i][j]=cambio[i-1][j];
            else{
                int menor=0;
                if(cambio[i-1][j]<cambio[i][j-monedas[i-1]] +1)
                    menor=cambio[i-1][j];
                else
                    menor=cambio[i][j-monedas[i-1]]+1;
            }
        }
    }

```

```

        cambio[i][j]=menor;
        aux[i][j]=1;
    }
}

return cambio[numero_monedas][devolucion];
}

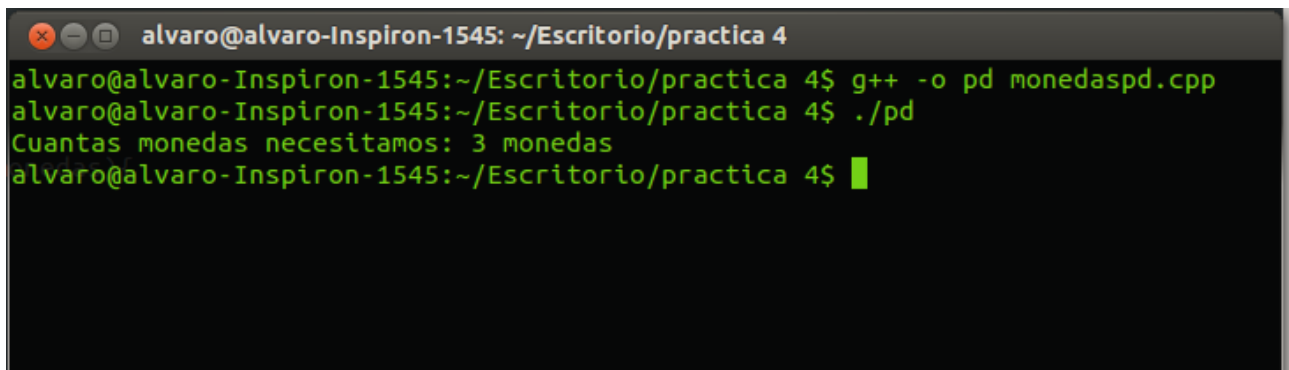
```

En la cabecera usamos:

- Un entero devolucion: Que representa la cantidad a devolver.
- Un vector de enteros monedas: Que representa los distintos tipos de monedas que tenemos, 1, 5, 10, 12, 25.
- Un entero numero_monedas: Que dice cuantas tipos distintos de monedas tenemos (5 en nuestro caso)

Básicamente lo que hacemos es rellenar la matriz cambio con números bajos (0) en la primera columna y numero altos en la primera fila y usamos la matriz aux como una matriz para saber si hemos cogido en la matriz cambio el numero de esa posición o no. Con una serie de if vamos controlando que es lo que metemos en las distintas posiciones y al final devolvemos la última posición de la matriz cambio que contendrá que cantidad de monedas tenemos (nos dará el número óptimo).

Ejemplo de ejecución con cantidad=32 y monedas de 1, 5, 10, 12, 25:



```

alvaro@alvaro-Inspiron-1545: ~/Escritorio/practica 4
alvaro@alvaro-Inspiron-1545:~/Escritorio/practica 4$ g++ -o pd monedaspd.cpp
alvaro@alvaro-Inspiron-1545:~/Escritorio/practica 4$ ./pd
Cuantas monedas necesitamos: 3 monedas
alvaro@alvaro-Inspiron-1545:~/Escritorio/practica 4$

```

Para que muestre el numero de monedas uso un simple cout:

```
cout << "Cuantas monedas necesitamos: " << CambioDinamico(cantidad,
monedas,numero_monedas) << " monedas" << endl;
```

Como ve usted, esta ejecución si nos devuelve el número optimo de monedas que necesitamos y que yo comenté arriba, 3 monedas.

El orden de eficiencia seria $O(n \cdot devolucion)$. Fuente: Libro de Técnicas de Diseño de Algoritmos.