LECCION 9: ALGORITMOS VORACES (Greedy)

ALGORITMOS VORACES = De avance rápido = Greedy

- Se usan en problemas de optimizacion normalmente.

- El planteamiento básico es:
  - Tomar algunos elementos de entre un conjunto de candidatos con objeto de optimizar una funcion objetivo.

- El orden en que se toma los elementos puede ser importante o no.

PASOS GENERALES (eje: comprar 2Kg de patatas).

1) Inicialmente partimos de una solucion vacia.
   Ej: la bolsa no tiene ninguna patata

2) Funcion de seleccion: se escoge el siguiente elemento a añadir a la solucion entre los candidatos
   Ej: seleccionamos a ojo de buen cubero la patata mejor del montón

3) Una vez escogido el elemento no se podrá deshacer esa eleccion.
   Ej: Examinamos la patata y decidimos si la echamos a la bolsa o no.
   Si no se coge se aparta del montón
   Si se coge se mete en la bolsa y ya no se sacan.

4) El algoritmo acabará cuando el conjunto de elementos seleccionados constituya una solucion.
   Ej: Una vez que tenemos 2Kg paramos.

METODO GENERAL

- Elementos
  - C: conjunto con los candidatos a seleccionar
  - S: Candidatos seleccionados para la solucion

```
Voraz (C: Conjunto_Candidatos,
       S: Conjunto_solucion)

1) S ← ∅
2) mientras (C ≠ ∅) AND ! Solucion(S)
3)     x ← Seleccionar(C)
4)     C ← C - {x}
5)     si factible(S, x) entonces
6)         S ← S U {x}
7) end
8) Si ! Solucion(S)
9)     cerr << "No se puede encontrar solucion"

END.
```

FUNCIONES GENÉRICAS

Solucion(S): comprueba si un conjunto S es solucion (indep. de que sea optima o no)

Seleccion(C): devuelve el elemento más prometedor en C.

factible(S,x): si añadimos a S x, este nuevo conjunto permite obtener una solucion.

Funcion objetivo(S): Dada una solucion devuelve el coste asociado a la misma (útil en problemas de optimizacion)

LECCION 9.- ALGORITMOS VORACES.

Problema 1.- Cambio de Monedas.

Construir un algoritmo que dada una cantidad P devuelva esa cantidad usando el menor número posible de monedas.

Ej:- Supongamos que disponemos de monedas de 1, 2, 5, 10, 20, 50 céntimos de euro, 1 y 2 euros.

Sea P = 3.89 euros.

		resto
* 1 moneda de	2 euros	1.89
* 1 " "	1 euro	0.89
* 1 moneda de	50 centimos	0.39
* 1 moneda de	20 "	0.19
* 1 "	10 "	0.09
* 1 "	5 centimos	0.04
* 2 monedas de	2 centimos	0

1º PLANTEAR LOS ELEMENTOS del ALGORITMO VORAZ.

- Conjunto de candidatos, C: todos los tipos de monedas en una cantidad ilimitada.
- Solucion, S:- Conjunto de monedas que sumen P
- Función objetivo:- minimizar el numero de monedas
- Representacion de la solucion.

  $(x_1, x_2, \dots, x_8)$: x_i es el nº de monedas usadas de tipo i

  8: es el nº de monedas diferentes

- Cada moneda i vale c_i

- Formulacion:-

  minimizar $\sum_{i=1}^{8} x_i$ sujeto a $\sum_{i=1}^{8} x_i \cdot c_i = P$ $\quad x_i \geq 0$

LECCION 9.- ALGORITMOS Voraces

Funciones del Esquema

1) Inicialización $x_i = 0 \quad \forall\, i = 1 \ldots 8$

2) Solucion: El valor actual $(x_1, x_2 \ldots x_8)$ si cumple $\sum x_i c_i = P$

3) Seleccionar: Escoger la moneda de mayor valor posible, sin sobrepasar la cantidad que queda por devolver.

4) Factible - siempre es verdad

NOTA1.- En vez de coger moneda una a una, tomamos tantas de un tipo como la division entera de la cantidad que queda por devolver y el valor de la moneda.

NOTA2.- Vamos a usar una variable 'act para acumular la cantidad devuelta hasta este punto
Y supongamos que las monedas están ordenadas de mayor ci a menor.

```
bool DevolverCambio (float P, vector<float>&C, vector<int> & X) {
int n = X.size(); //nº de monedas diferentes.
float act=0.0;
int j=n-1;
for (int i=0; i<n; i++)
      X[i]=0;
while (act!= P) {
   while ( C[j]> P-act && j>=0)
         j=j-1;
   if (j<0){
         cout << "No existe solucion";
         return false;
   }
   X[j] = floor( (P-act)/C[j] );
   act = act + X[j] * C[j];
}
return true;
}
```

LECCION 9.- ALGORITMOS VORACES

EFICIENCIA del Problema del CAMBIO de MONEDAS

La eficiencia viene dada en el caso que tengamos que analizar coger o no coger todas los tipos de monedas que es n. Por lo tanto la eficiencia es O(n).

- CONCLUSIONES SOBRE OPTIMALIDAD

Para el sistema monetario que hemos escogido siempre vamos a obtener una solución aunque esta no sea la optima. Supón que eliminamos las monedas de 1 céntimo ¿siempre obtendríamos solución?

- Un ejemplo en el que obtenemos solución pero no la optima es para P= 180. Siguiendo la estrategia de coger la moneda de mayor valor, con monedas 100, 90 y 1, tomaría 1 de 100 y 80 de 1 euro. Pero la solución optima sería escoger 2 monedas de 90.

- EFICIENCIA en Terminos Generales de los ALG. VORACES.

$$T(n,m) \in O(n * f(m) + g(n) + h(m) + m * j(n,m))$$

- n: candidatos
- m: nº de elementos de la solución.
- $f(m)$: Eficiencia para ver si el valor actual es solución
- $g(n)$: Selección
- $h(m)$: Factible
- $j(n,m)$: Unión de un nuevo elemento a la solución.