# Algoritmia Básica

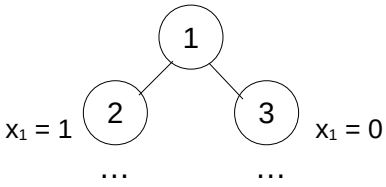
## Práctica 2: Problema de transporte

Andrés Gavín Murillo 716358
Andrew Mackay 737069

# Descripción del problema

Se trata de un problema, con similitudes al problema de la mochila (en este caso es un tren), donde la capacidad máxima es n y los pedidos (p pedidos) tienen un coste np (número de pasajeros). La principal diferencia consiste en que la restricción de la capacidad máxima se debe cumplir en todas las estaciones en las que para el tren {0..m}.

El problema se ha resuelto con la técnica de ramificación y poda, donde las soluciones son de la forma X = {$x_1$..$x_p$} con $x_i$ = 0 para cancelar el pedido i, y $x_j$ = 1 para aceptar el pedido j. Cada pedido está formado por una tupla $p_i$ = {$es_i$, $el_i$, $np_i$}, donde $es_i$ es la estación de salida, $el_i$ es la estación de llegada y $np_i$ es el número de pasajeros. Todas las tuplas son de tamaño fijo y las soluciones factibles.




Restricciones:

| | | |
|---|---|---|
| n ≥ 0 | $es_i$ = {0..m-1} | $np_i$ ≥ 0 |
| m = {1..7} | $el_i$ = {1..m} | |
| p = {1..22} | $el_i$ > $es_i$ | |

Un nodo es factible si para cada estación $\left(\sum_{i=1} z=\begin{Bmatrix} np_i \times x_i, & i \geq es_i \wedge i < el_i \\ 0, & \neg(i \geq es_i \wedge i < el_i) \end{Bmatrix}\right) \leq n$

Para transformar el problema en un problema de minimización, en primer lugar se define el beneficio de un pedido i como $b_i = (el_i - es_i) \times np_i$ y, el objetivo es maximizar el beneficio, es decir, se minimiza el beneficio negativo:

$min - \sum_{i=1}^{p} b_i \times x_i$ , siendo los nodos factibles.

**Definición de la función de coste**

$$c(x) = \begin{cases} -\sum_{i=1}^{p} b_i \times x_i, & si\ x\ nodo\ solución\ factible \\ \infty, & si\ x\ nodo\ solución\ no\ factible \\ min(c(x_{izq}), c(x_{der})), & si\ x\ nodo\ no\ hoja \end{cases}$$

### Definición de la función de estimación

```
tipo pedido = registro
                    es, el, np, b: entero
              freg;

variables pedidos: vector[1..p] de pedido;
          x: vector[1..p] de entero

funcion cota(pas: vector[0..m] de entero; ben, obj: entero) devuelve entero
{pas = número de pasajeros actuales en cada estación; ben = beneficio actual; obj =
índice del primer objeto a considerar}
variables auxPas: vector[0..m] de entero
principio
  si obj > p entonces devuelve ben
  sino
    para i:=0 hasta m hacer
      si i >= pedidos[obj].es and i < pedidos[obj].el entonces
        auxPas[i] := pas[i] + pedidos[obj].np
      sino
        auxPas[i] := pas[i]
      fsi
    fpara;
    si existe i = 0..m: auxPas[i] > n entonces devuelve cota(pas, ben, obj+1)
    sino
      si forall i = 0..m: auxPas[i] = n entonces devuelve ben + pedidos[obj].b
      sino
        devuelve cota(auxPas, ben + pedidos[obj].b, obj+1)
      fsi
    fsi
  fsi
fin

funcion calcularPas(j: entero) devuelve vector[0..m] de entero
{j = índice del último objeto a considerar}
variables pas: vector[0..m] de entero
principio
  para i := 0 hasta m hacer
    pas[i] := 0
  fpara;
  para i := 1 hasta j hacer
    si x[i] = 1 entonces
      para k := pedidos[i].es hasta pedidos[i].el-1 hacer
        pas[k] := pas[k] + pedidos[i].np
      fpara
    fsi
  fpara
fin
```

$$\hat{c}(x)=\begin{cases} -cota\left(calcularPas(j), \sum_{i=1}^{j} b_i \times x_i, j+1\right), & si\ x\ nodo\ factible \\ \infty, & si\ x\ nodo\ no\ factible \end{cases}$$

**Definición de la función de poda u**
Si x es un nodo de nivel j, con $1 \leq j \leq p$, se han asignado ya valores a $x_i$, $1 \leq i \leq j$, por tanto:

$$c(x) \leq -\sum_{i=1}^{j} b_i \times x_i = u$$

# Decisiones tomadas

Cada bloque se resuelve de manera independiente y secuencial (se podrían calcular en paralelo si importara el tiempo de ejecución total, pero en este caso el tiempo a considerar es el de resolución de cada bloque).

La información del tren correspondiente a cada bloque se almacena en un TAD de tipo Tren.

Los pedidos forman una lista enlazada, que previamente ha sido ordenada, y cada pedido almacena información del pedido correspondiente, un puntero al tren de su bloque y un puntero al siguiente pedido.

Se ha utilizado el heap de mínimos de la práctica anterior (adaptado para tipos de datos genéricos) para ordenar los pedidos de cada bloque por su beneficio y de manera descendente.

Además, se ha utilizado el heap de mínimos para expandir siempre el nodo con mayor beneficio. Sólo se insertan en el heap los nodos factibles que no han sido podados.
En el caso de que sea un nodo hoja (posible solución) se inserta en otro heap de soluciones, en el que su cima será la mejor solución cuando ya no queden más nodos en el heap de nodos factibles.

El árbol (binario) de exploración está formado por nodos que a su vez son del tipo árbol e incluyen datos correspondientes al pedido y otros datos precalculados (ej. beneficio) para acelerar los cálculos posteriores.
Cada nodo tiene acceso a su padre y a sus hijos (en el caso de que el nodo haya sido expandido).