# TEORÍA DE ALGORITMOS
# RELACIÓN DE PROBLEMAS
# ALGORIMOS *BACKTRACKING* Y *BRANCH&BOUND*

1. Un instructor de esquí dispone de n pares de esquís para sus n alumnos. Obligatoriamente cada alumno debe recibir un par de esquís, que han de adecuarse lo máximo posible a su altura. El problema del instructor es asignar los esquís a los alumnos de forma que se minimice la suma de los valores absolutos de las diferencias entre las alturas de los alumnos y las longitudes de los respectivos esquís asignados. Proponer un algoritmo que resuelva este problema de forma óptima y aplicarlo sobre el siguiente ejemplo.

   Altura: 178 168 190 170
   Longitud: 183 188 168 175

2. Resolver el problema de la mochila 0-1 con los siguientes valores: Tamaño de la mochila M = 61; número de objetos n = 5; matriz de pesos W = (1, 11, 21, 23, 33); matriz de ganancias P = (11, 21, 31, 33, 43).

   Representar el árbol de estados que se obtendría al utilizar las técnicas de Backtraking y Branch and Bound respectivamente. ¿Qué funciones de acotación utilizarías para reducir el espacio de búsqueda?

   Nota: En ambos casos, numerar los nodos según el orden en que son expandidos y comentar los criterios que se siguen para la expansión o poda de los nodos.

3. Diseñar un algoritmo BB para resolver el problema de asignación que tiene asociada la siguiente matriz:

| | **f1** | **f2** | **f3** | **f4** |
|---|---|---|---|---|
| **d1** | 94 | 1 | 54 | 68 |
| **d2** | 74 | 10 | 88 | 82 |
| **d3** | 62 | 88 | 8 | 76 |
| **d4** | 11 | 74 | 81 | 21 |

   Enunciado al problema anterior: Tenemos un conjunto de Datos (d1, d2, ..., d4) a almacenar y un conjunto de Ficheros en los que se pueden almacenar (f1, f2, ..., f4). El coste de almacenar cada dato en un fichero viene determinado por el número de pasos que hay que dar hasta encontrar la posición en la que dicho dato viene almacenado. Se pretende asignar los datos a los ficheros con mínimo costo global.

4. Resolver el problema del viajante de comercio utilizando la técnica BB donde tenemos 5 ciudades y la matriz de distancias (simétrica) es:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| A | 0 | 3 | 10 | 11 | 7 |
| B | | 0 | 8 | 12 | 9 |
| C | | | 0 | 9 | 4 |
| D | | | | 0 | 5 |

5. Cuestiones teóricas:

   a) Indicar las similitudes y diferentes entre Backtraking y Branch and Bound.
   b) Representación de los algoritmos Backtraking. Restricciones Implícitas y Explícitas. Comentarlo utilizando como ejemplo la resolución del problema de la suma de subconjuntos.

6. Una matriz booleana M[1..n,1..n] representa un laberinto cuadrado donde las casillas adyacentes a una dada se encuentran en la misma fila o columna. Si M[1,j] es cierto, se puede pasar por la casilla (i,j); si M[i,j] es falso no se puede pasar por la casilla (i,j).

   La siguiente figura nos muestra un ejemplo, siendo I la casilla de inicio, S la casilla de salida, y X las casillas con valor falso.

| | | | | | |
|---|---|---|---|---|---|
| I | ↓ | | | | |
| X | ↓ | | X | | X |
| ← | ↓ | X | X | | |
| ↓ | X | X | X | | X |
| ↓ | X | ↑ | → | ↓ | X |
| ↓ | → | → | X | S | X |

a) Dar las restricciones implícitas y explícitas del problema anterior de forma que nos aseguren un árbol de estados finito. Representar el árbol de estados para el laberinto de la figura.

b) Diseñar un algoritmo Backtraking que partiendo de la casilla de inicio y llegando a la salida encuentre un camino en el laberinto.

c) Indicar cómo se podría resolver este problema utilizando una técnica BB con criterio LC-Search (el nodo a expandir en cada caso es el más prometedor). Representar el orden en que se expanden los nodos en el ejemplo anterior.

1. Asignar n pares de esquís a n alumnos.

- Suponemos un vector de longitudes $l = (l_1, \ldots, l_n)$ con $l_i$ representando la longitud del esquí $i$.
- Suponemos un vector de alturas $a = (a_1, \ldots, a_n)$ con $a_i$ representado la altura del alumno $i$.

- Representaremos la solución mediante un árbol combinatorio:

$X = (x_1, \ldots, x_n)$ donde $x_i \in \{1, \ldots, n\}$ y representa el esquí que se asigna al alumno $i$. Es decir, las posiciones representan a los alumnos y los contenidos a los esquís asignados. $\Rightarrow (1,3,2) \neq (2,1,3)$.

n! nodos hoja

- Hijos de un nodo $(x_1, \ldots, x_k)$:

$(x_1, \ldots, x_k, x_{k+1})$ tq. $x_{k+1} \neq x_i \ \forall i \in \{1, \ldots, k\}$ (única restricción implícita).

Es decir, en el nivel 0 tenemos la raiz del árbol (se consideran todos los esquís), en el nivel 1 se consideran todos los esquís menos 1, en el nivel 2 todos menos 2 y así sucesivamente.

- Función objetivo (a minimizar):

$$F(X) = \sum_{i=1}^{n} |l_{x_i} - a_i|$$

Aunque este problema se puede resolver aplicando Backtracking, en este caso, al no haber muchas restricciones implícitas no se eliminan muchas ramas del árbol solución. Para poder podar más ramas se va a considerar un enfoque Branch & Bound. Para facilitar las cosas se considerará que el estimador para la ramificación será la cota local. La cota global se actualizará únicamente cuando se encuentre una nueva solu ción si su coste es mejor que el indicado por dicha cota global.

- Cota local o inferior $CI(j)$ para un nodo hijo $j$:
  - Coste Decis. Tomadas:

$$CDT(j) = \sum_{i=1}^{k+1} | l_{x_i} - a_i |$$

  - Coste Decis. No Tomadas (estimado):

$$CNT(j) = \sum_{i=k+2}^{n} \min(|l_c - a_i|) \quad \forall \ c \in \{1, \ldots, n\}.$$

Es decir, se asigna a los alumnos que quedan el mejor esquí posible. No es tan buena como asignar el mejor esquí de los que quedan, pero es muy eficiente.

$$CI(j) = CDT(j) + CNT(j)$$

- Estimador: $EC(j) = CI(j)$.
- Cota global o superior $CS$: se actualiza cada vez que se encuentre una solución mejor. El valor inicial podría ser infinito. Pero se usará el coste de la solución encontrada por un algoritmo Greedy. El Greedy consistirá en asignar a cada alumno el mejor esquí disponible y eliminarlo de la lista de candidatos.
- Se utilizará una estrategia LC-LIFO.

- Para facilitar los cálculos, al principio del algoritmo se calcularán las diferencias entre la altura de cada alumno y la longitud del mejor esquí posible para dicho alumno. Estos valores se almacenarán en el array MDEst (mejor distancia estimada).
- La lista de esquís que todavía no han sido asignados se almacenará para cada nodo, junto con la solución parcial en el mismo vector.

```
EsquisBB (l, a : array [1..n] of integer; var s : nodo);
    MDEst : array [1..n] of real;
    // Se almacena el coste de asignar el mejor esquí a cada alumno
    // en MDEst
    Para i := 1..n hacer
        MDEst [i] := abs (l [1] - a[i]);
        Para j := 2..n hacer
            Si (abs (l [j] - a [i]) < MDEst [i]) entonces
                MDEst [i] := abs (l[j] - a [i]);
            FinSi;
        FinPara;
    FinPara;
    s := EsquisGreedy (l, a, s);
    C := s.coste;
    LNV := {NodoInicial (l, a, MDEst)};
    Mientras LNV ≠ ∅ hacer
        x := selecciónar (LNV);
        LNV := LNV - {x};
        Si (x.CI < C) entonces
            Para i := x.nivel+1..n hacer //esquis restantes desde nivel+1
                y := Generar (x, i, l, a, MDEst);
                Si (y.nivel = n) y (y.coste < s.coste) entonces
                    s := y;
                    C := y.coste;
                Sino Si (y.nivel < n) y (y.CI < C) entonces
                    LNV := LNV + {y};
                FinSi;
            FinSi;
        FinPara;
    FinSi;
    FinMientras;
FinEsquisBB;

NodoInicial (l, a : array [1..n] of integer; MDEst : array [1..n] of real) : nodo;
    res.nivel := res.cdt := res.cnt := res.coste := 0;
    Para i := res.nivel+1..n hacer
        res.tupla [i] := i; // se incluyen los esquis libres a partir de nivel.
        res.cnt += MDEst [i];
    FinPara;
    res.EC := res.CI := res.cdt + res.cnt;
    Devolver res;
FinNodoInicial;
```

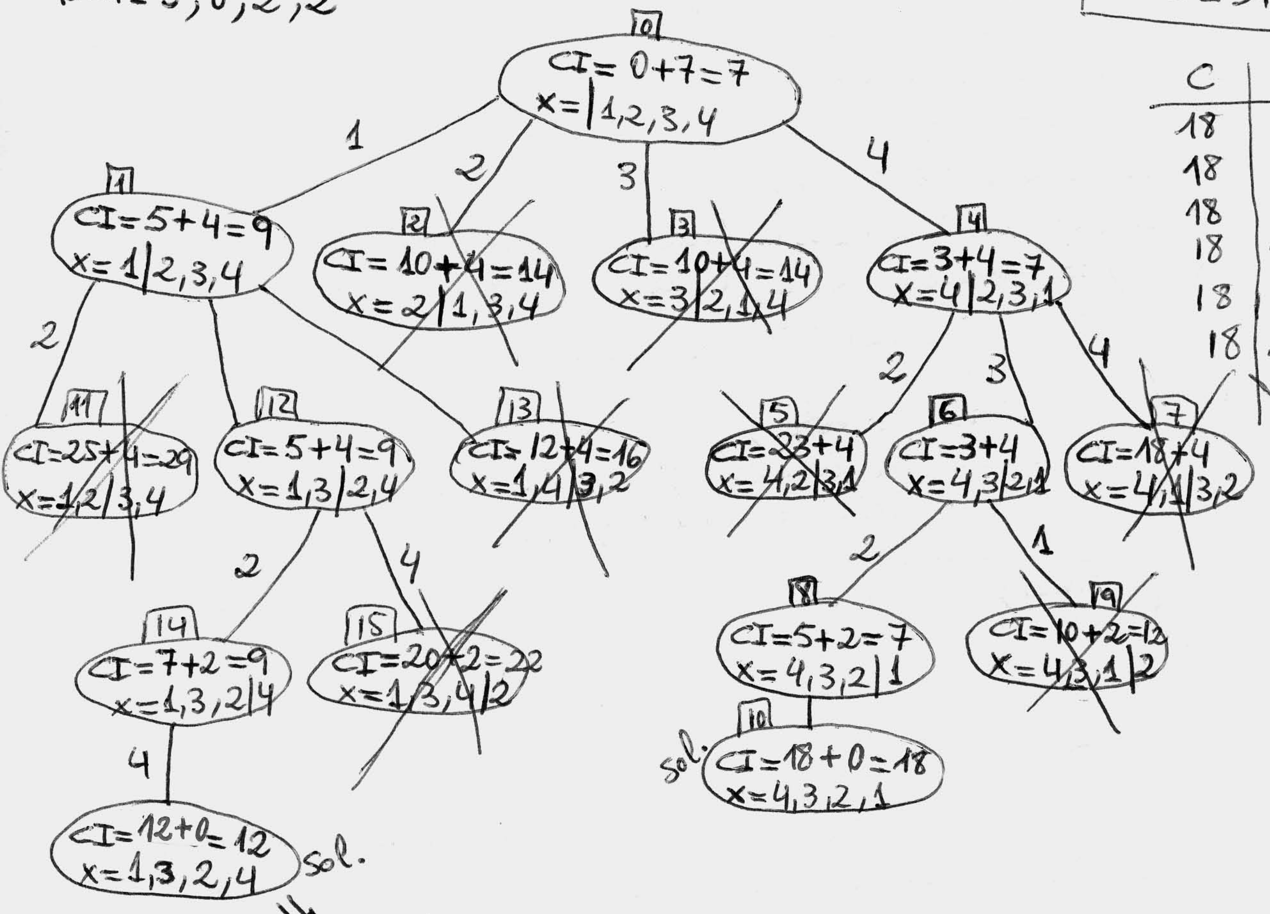
```
Generar (x: nodo; esqui: integer; l, a: array [1..n] of integer;
        MDEst: array [1..n] of real): nodo;
    res.tupla := x.tupla;
    res.nivel := x.nivel + 1; res.cnt := x.cnt; res.cdt := x.cdt;
    res.tupla[res.nivel] := x.tupla[esqui]; // coloca el esquí en siguiente posición
    res.tupla[esqui] := x.tupla[res.nivel]; // repone el esquí libre "sobrescrito"
    res.cnt -= MDEst[res.nivel];
    res.cdt += abs(l[res.tupla[res.nivel]] - a[res.nivel]);
    res.EC := res.CI := res.cdt + res.cnt;
    res.coste := res.cdt;
    Devolver res;
FinGenerar;
```

EJEMPLO: Altura : 178, 168, 190, 170

Longitud : 183, 188, 168, 175

MDEst = 3, 0, 2, 2

Sol. Greedy:
$s = 4, 3, 2, 1$
$C = 3+0+2+13 = 18$




| C | LNV |
|---|---|
| 18 | 0 |
| 18 | 4, 1, 3, 2 |
| 18 | 6, 1, 3, 2 |
| 18 | 8, 1, 9, 3, 2 |
| 18 | 1, 9, 3, 2 |
| 18 | 12, 9, 3, 2, 13 |
| 18 | 14, 9, 3, 2, 13 |
| 12 | 9, 3, 2, 13 |
| 12 | 3, 2, 13 |
| 12 | 2, 13 |
| 12 | 13 |
| 12 | |

$\Rightarrow$ $s = 1, 3, 2, 4$
$C = 12$

---

## ② BACKTRACKING.

- Representación de la solución mediante árbol binario:

$X = (x_1, \ldots, x_n)$ donde $x_i \in \{0, 1\} \Rightarrow$ No se repiten soluciones.

- Hijos de un nodo $(x_1, \dots, x_k)$:

$$(x_1, \dots, x_k, 0) \text{ y } (x_1, \dots, x_k, 1) \quad \text{tq.} \quad \sum_{i=1}^{k+1} x_i \cdot w_i <= M$$

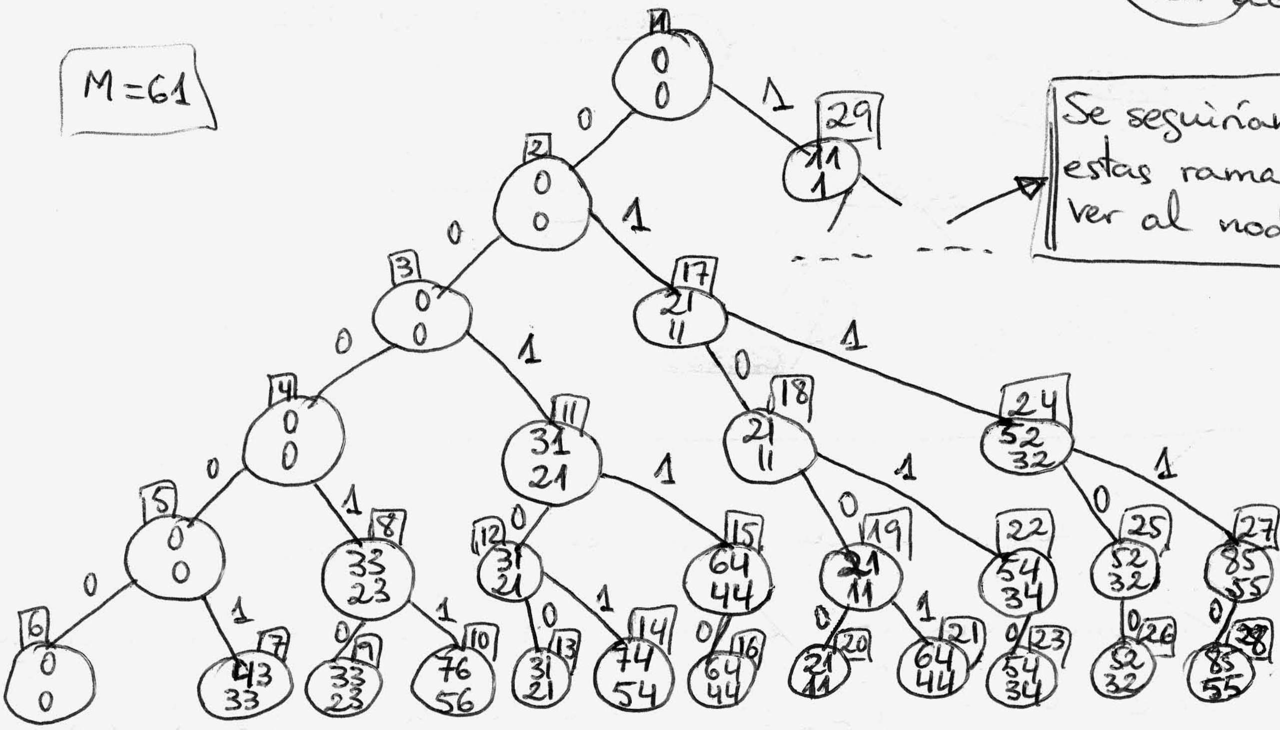
Las restricciones implícitas están consideradas por la representación y esta condición.

- Función objetivo (a maximizar):

$$F(x) = \sum_{i=1}^{n} x_i \cdot P_i$$

Cada nodo se representará como sigue → Beneficio acumulado / Peso acumulado.

$\boxed{M=61}$

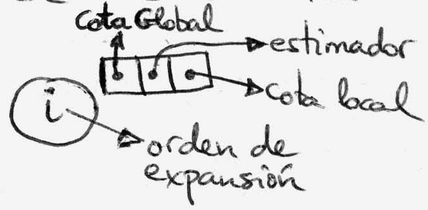


Se seguirían generando estas ramas hasta volver al nodo raíz.

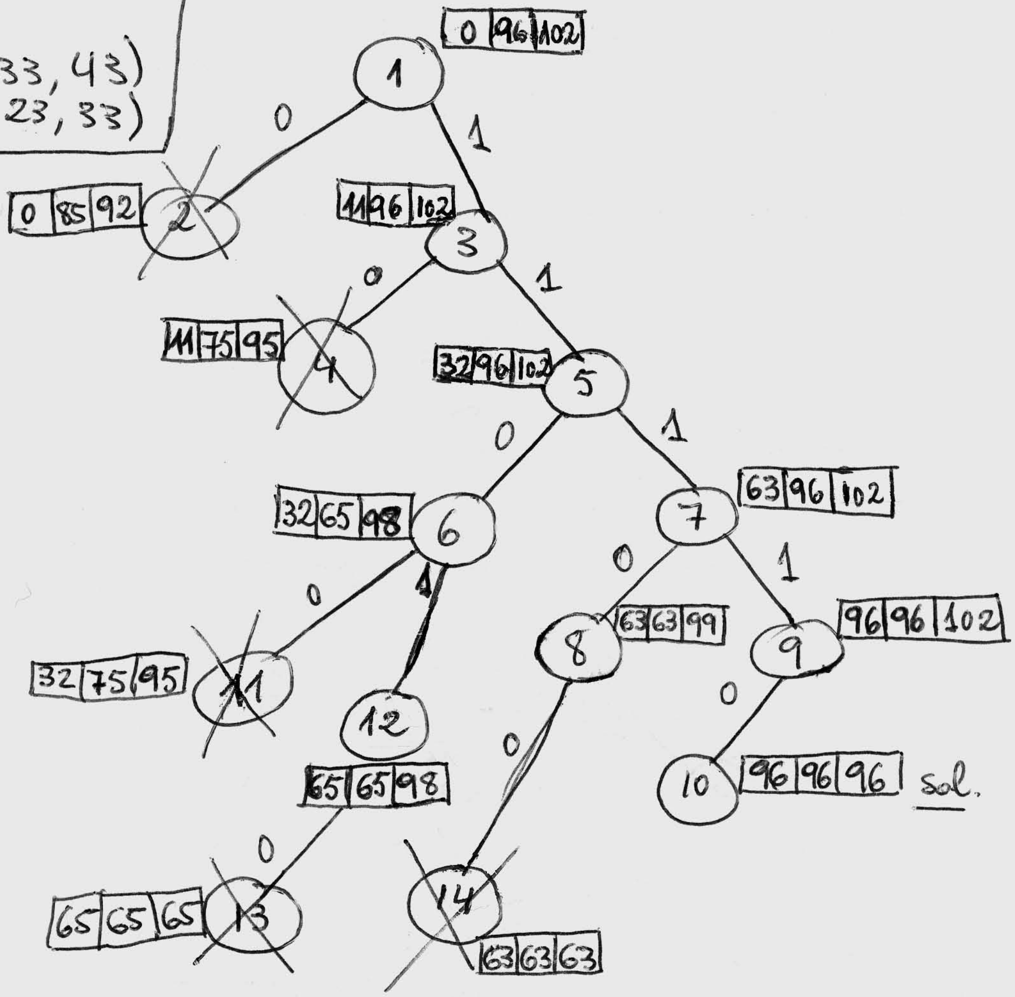
BRANCH & BOUND

Se utilizará el algoritmo descrito en las transparencias de la asignatura. No obstante, habría que describir las cotas, etc., para resolver el ejercicio si se tratase de un examen.

Representación de un nodo →



Ordenados por densidad decreciente:
$P = (11, 21, 31, 33, 43)$
$W = (1, 11, 21, 23, 33)$




③ Asignar n conjuntos de datos a n ficheros.

- Suponemos una matriz de costos $A$, con $a_{ij}$ representando el coste de asignar el conjunto de datos i al fichero j:

$$A = \begin{pmatrix} a_{11}, & \dots, & a_{1n} \\ \vdots & & \vdots \\ a_{n1}, & \dots, & a_{nn} \end{pmatrix}$$

- Representaremos la solución mediante un árbol combinatorio: $X = (x_1, \dots, x_n)$ donde $x_i \in \{1, \dots, n\}$ y representa el conjunto de datos que se asignará al fichero i. Las posiciones representan los ficheros y los contenidos a los conjuntos de datos $\Rightarrow (1,3,2) \neq (2,1,3)$.

- Hijos de un nodo $(x_1, \dots, x_k)$:

  $(x_1, \dots, x_k, x_{k+1})$ tq. $x_{k+1} \neq x_i \ \forall i \in \{1, \dots, k\}$, (única restricción implícita). Se genera un árbol con $n!$ nodos hoja.

- Función objetivo (a minimizar):

$$F(X) = \sum_{i=1}^{n} A[x_i][i].$$

- El estimador será la cota local y la cota global se actualizará respecto a la mejor solución encontrada.
- Cota local o inferior $CI(j)$ para un nodo hijo $j$:
  - Coste Decisiones Tomadas:

$$CDT(j) = \sum_{i=1}^{k+1} A[x_i][i].$$

  - Coste Decisiones No Tomadas:

$$CNT(j) = \sum_{i=k+2}^{n} \min\left(A[c][i]\right) \ \forall c \in \{1, \dots, n\}.$$

  Se asigna a los ficheros que quedan el mejor conjunto posible.

$$CI(j) = CDT(j) + CNT(j).$$

- Estimador $EC(j) = CI(j)$.
- Cota global o superior $CS$: Se actualiza cada vez que encontremos una solución mejor. El valor inicial vendrá dado por un Greedy: asignar a cada fichero el mejor conjunto de datos disponible y eliminarlo de

la lista de candidatos.

- Se usará LC-LIFO.
- Al principio del algoritmo se almacenarán los costes de asignar el mejor conjunto de datos para cada fichero en el array MCEst (mejor coste estimado).
- La lista de conjuntos de datos disponibles se almacenará junto con la solución parcial en el mismo vector.

ALGORITMO:

```
DatosBB (A: array [1..n][1..n] of integer; var s: nodo);
   MCEst: array [1..n] of integer; // mejor coste estimado para un
                                   // fichero.
   Para i:=1..n hacer
        MCEst [i] := A[1][i];
        Para j:=2..n hacer
            si (A[j][i] < MCEst [i]) entonces
                MCEst [i] := A[j][i];
            FinSi;
        FinPara;
   FinPara;
   s:= Datos_Greedy (A, s);
   C:= s.coste;
   LNV:= {NodoInicial (A, MCEst)};
   Mientras LNV ≠ ∅ hacer
       x:= Seleccionar (LNV);
       LNV:= LNV - {x};
       si (x.CI < C) entonces
          Para i:=x.nivel+1 .. n hacer
              y:= Generar (x, i, A, MCEst);
              Si (y.nivel=n) Y (y.coste < s.coste) entonces
                  s:=y;
                  C:=y.coste;

        ⊛
```

(*)

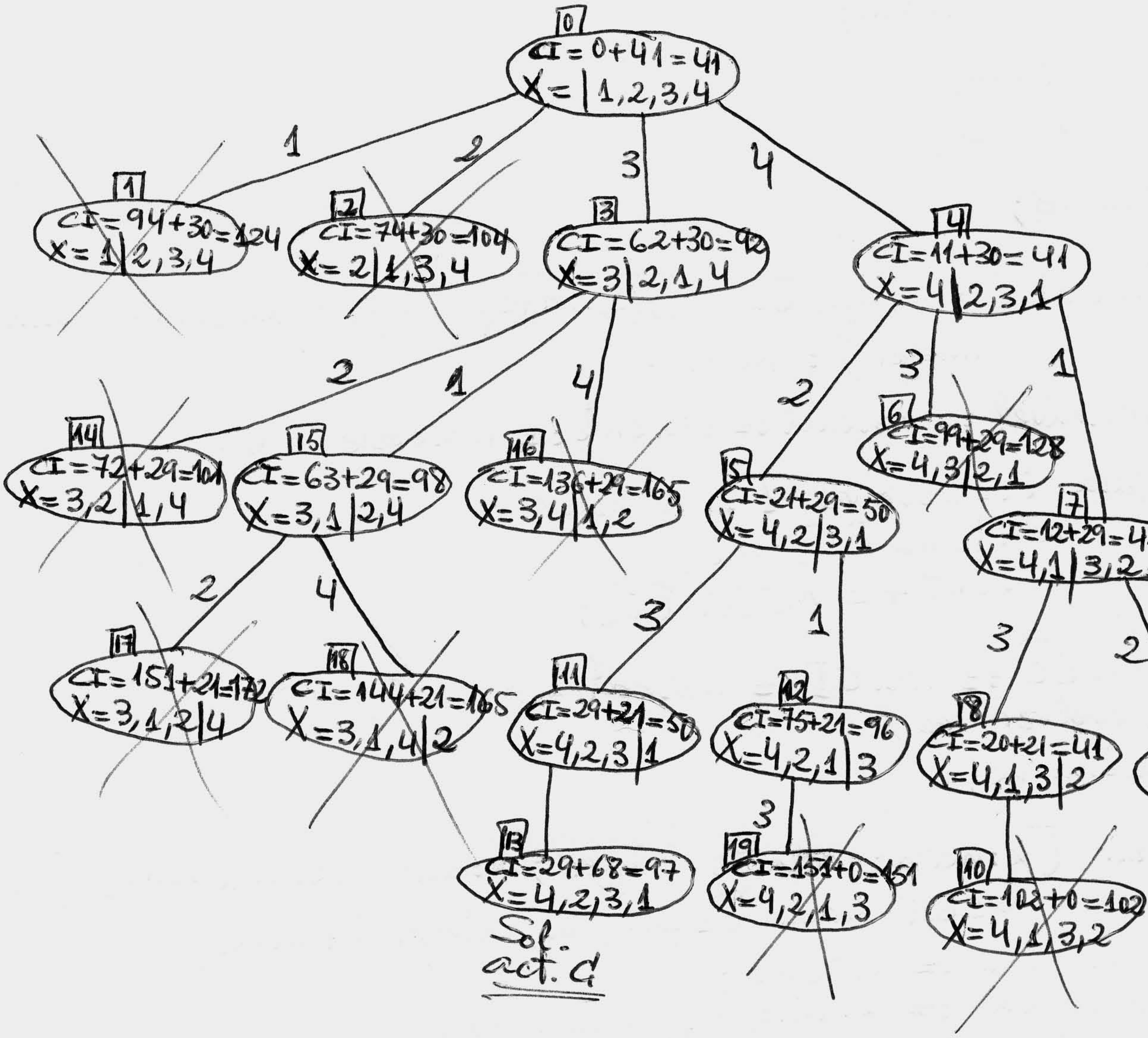
```
                sino Si (y.nivel<n) Y (y.CI<C) entonces
                        LNV := LNV+{y};
                     FinSi
                FinSi;
            FinPara;
        FinSi;
    FinMientras;
FinDatosBB;

NodoInicial (A: array [1..n][1..n] of integer; MCEst: array [1..n] of
             integer): nodo;
    res.nivel := res.cdt := res.cnt := res.coste := 0;
    Para i:=res.nivel+1 .. n hacer
        res.tupla [i] := i;
        res.cnt += MCEst [i];
    FinPara;
    res.EC := res.CI := res.cdt + res.cnt;
    Devolver res;
FinNodoInicial;

Generar (x:nodo; cdatos: integer; A:array [1..n][1..n] of integer;
         MCEst: array [1..n] of integer): nodo;
    res.tupla := x.tupla;
    res.nivel := x.nivel+1; res.cdt := x.cdt; res.cnt := x.cnt;
    res.tupla [res.nivel] := x.tupla [cdatos];
    res.tupla [cdatos] := x.tupla [res.nivel];
    res.cnt -= MCEst [res.nivel];
    res.cdt += A [res.tupla [res.nivel]] [res.nivel];
    res.EC := res.CI := res.cdt + res.cnt;
    res.coste := res.cdt;
    Devolver res;
FinGenerar;
```

EJEMPLO:

$$A=\begin{pmatrix} 94, & 1, & 54, & 68 \\ 74, & 10, & 88, & 82 \\ 62, & 88, & 8, & 76 \\ 11, & 74, & 81, & 21 \end{pmatrix}$$

Sol. Greedy:
$s = 4, 1, 3, 2.$
$C = 11+1+8+82 = 102$

MCEst = 11, 1, 8, 21.




| C | LNV |
|---|---|
| 0 | 0 |
| 102 | 4,3 |
| 102 | 7,5,3 |
| 102 | 8,5,3 |
| 102 | 5,3 |
| 102 | 11,3,12 |
| 97 | 3,12 |
| 97 | 15,12 |
| 97 | 12 |
| 97 | $\emptyset$ |