Vuelta atrás + voraces - Enero 2017

Secuencia de decisiones: $< x_1, x_2, \ldots x_p >$ donde x_i indica al amigo al que tiene que regalar la persona i.

Función objetivo: maximizar $\sum_{i=1}^{p} Afecto[i][x_i]$

Reestricciones implícitas: $(\forall i)(x_i \in \{1,2,..P\} : 1 \leq i \leq P)$

Reestricciones explícitas:

$$(\forall i)((\forall j)(i \neq j) \rightarrow x_i \neq x_j : 1 \leq j \leq P) : 1 \leq i \leq P)$$

$$(\forall i)(i \neq x_i : 1 \leq i \leq P)$$

$$(\forall i)(x_i \neq Excluir[i] : 1 \leq i \leq P)$$

Se busca una solución que siga el esquema óptimo.

Preparar-recorrido-nivel-k: $x[k] = 0$

∃-hermano-nivel-k: $x[k] < N$

Siguiente-hermano-nivel-k: $x[k] += 1$

Función solución: $k = P$

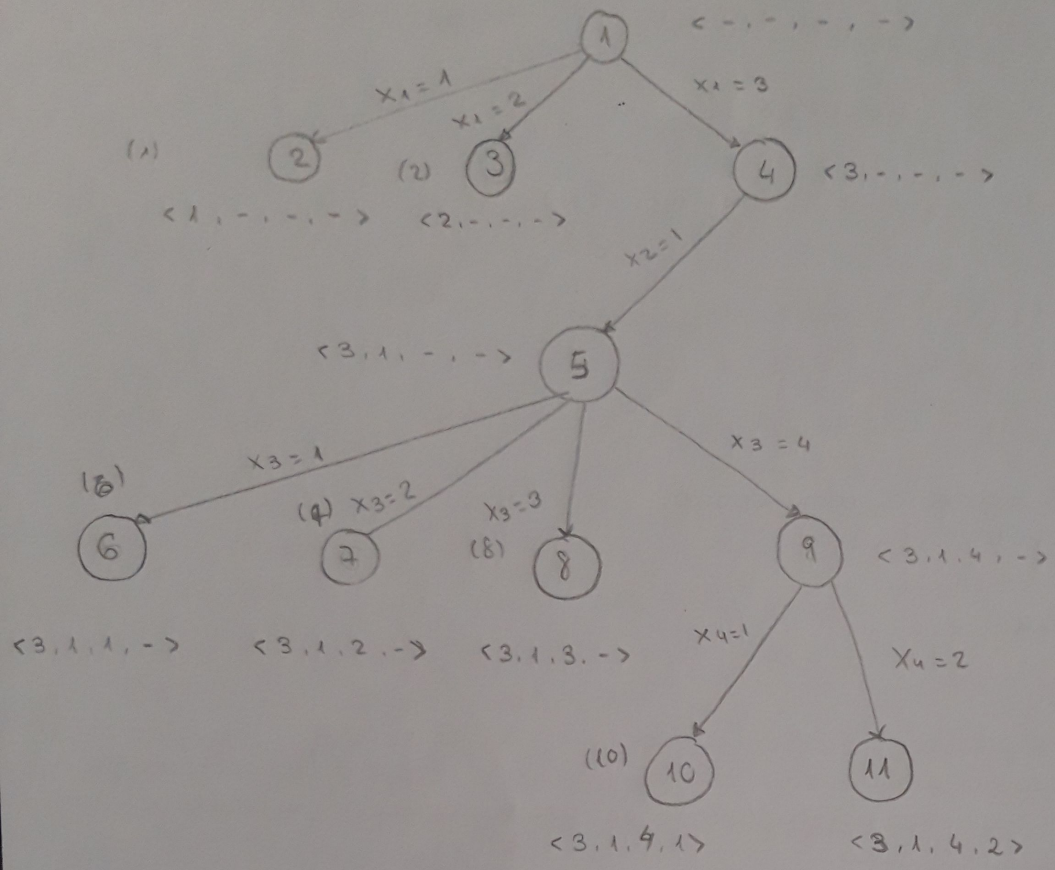
Función correcto: recibe la secuencia de decisiones x, y el valor de k, y devolverá falso si la persona k se regala a sí misma o si la persona k regala a una persona que no debe (Excluir[k]). Si el amigo asignado a la persona k aparece en el tramo de decisiones [1..k-1], devolverá falso.

```
Función correcto (Excluir[1..P]: vector de enteros; x: tupla; k: entero) retorna
                                                              (b: booleano)
var i: entero, ok: booleano fvar
i = 0
ok = verdadero
si (k = x[k] v k[k] = Excluir[k]) ok = falso; fsi
mientras (i < k-1 ∧ ok) hacer
        i = i+1;
        si (x[i] = x[k]) entonces ok = falso;
        fsi
fmientras   retorna ok
ffuncion
```

Función valor: recibe la secuencia de decisiones x y el valor de k y devuelve el valor de la función objetivo correspondiente a la secuencia de decisiones x, esto es, la suma de los afectos conseguidos con la asignación de regalos.

```
Funcion valor (Afecto[1..P][1..P]: matriz de enteros; k: entero; x: tupla)
        retorna (e: entero)
var i: entero, total: entero fvar
i=0; total=0;
mientras (i<k-1) hacer
        i = i+1;
        total = total + Afecto[i][x[i]];
fmientras
retorna total
ffuncion
```

Árbol de búsqueda P = 4 . Excluir [1..4] = {2,0,2,0}



(1) 1 no puede regalarse a sí mismo.
(2) 1 no puede a 2 (Excl.)
(6) 1 no puede regalos > 1
(7) 3 no puede a 2 regala
(8) no se puede a sí misma
(10) 1 no puede recibir > 1 regalo

Mediante Kruskal

Candidatos: siguiendo la solución del Backtracking, $\langle x_1, x_2, \ldots x_P \rangle$ donde x_i será el amigo al que regalará la persona i-ésima. Los candidatos serán las P personas. Todas ellas forman parte de la solución, el problema es determinar el orden.

Criterio de selección: en cada etapa i se seleccionará a un amigo que no haya sido escogido, distinto de la persona i-ésima y no excluido para i (Excluir [i]) cuyo afecto de la persona hacia i sea máximo.

Función de factibilidad: el criterio de selección hace que todas las propuestas sean factibles.
Función de solución: la solución cuando en el conjunto solución estén las P personas.

$$Afecto[1..4][1..4] = \begin{bmatrix} 0 & 10 & 8 & 6 \\ 9 & 0 & 7 & 5 \\ 1 & 2 & 0 & 3 \\ 6 & 5 & 4 & 0 \end{bmatrix}$$

Etapa	Candidato selec	Solución	Valor
0	--	< -, -, -, ->	0
1	3	< 3, -, -, ->	8 + 0 = 8
2	1	< 3, 1, -, ->	8 + 9 = 17
3	4	< 3, 1, 4, ->	17 + 3 = 20
4	2	< 3, 1, 4, 2>	20 + 5 = 25