Ficha de descripción del problema mediante Programación Dinámica

- **Descripción:** Escoger el número de gramos a seleccionar de entre los ingredientes disponibles, sabiendo que solo podemos seleccionar un máximo de 1000 gramos, que hay que minimizar el coste total (cada ingrediente tiene coste) y que con la selección de ingredientes que realicemos hay que cumplir un mínimo de nutrientes existentes (cada ingrediente posee una cantidad de nutrientes por gramo).

- **Tipos:** S - SoluciónAlimentos
  A - Integer.

- **Propiedades Compartidas:**
  - ProblemaAlimentos po.
    - List<Integer> minNut.
    - Integer caMax.
    - List<IngredienteActivo> ing.
      - List<Double> nut.
      - Double co.
  - Integer ingN = ing.size()
  - Integer minNutN = minNut.size()

- **Propiedades individuales:**
  - Integer index.
  - Integer caAc.
  - Double coAc.
  - List<Double> nutAc.
  - Integer caRe = po.caMax − caAc.

- **Solución:** SoluciónAlimentos.

- **Tamaño:** po.ingN − index + 1

- Solución parcial: (a, v)

  Siendo v el coste total de los ingredientes.

  Las alternativas representan el número de gramos escogidos del ingrediente po.ing(index).

- Alternativas: $A = [0, caR]$

  La alternativa a representa el número de gramos que escogemos del ingrediente po.ing(index).

- Instanciación: Inicial = (0, 0, 0.0, nutAc)

  Siendo nutAc una List<Double> con po.minNutN "ceros".

- Caso base: index = po.ingN.

- Solución caso base:

  cumpleMinimos(): función que devuelve true o false si llegados a un caso base se cumple que cada elemento nutAc es mayor o igual que el correspondiente elemento de po.minNut, o no.

  - Si (cumpleMinimos): (null, 0.).
  - Si (!cumpleMinimos): null.

- Número de subproblemas: 1 (Reducción).

- Subproblema:

  $p = (index, caAc, coAc, nutAc) \xrightarrow{a} p_a = (index+1, caAc+a, NextCoAc, NextNutAc)$.

  $NextCoAc = coAc + a * po.ing(index).co.$

  $NextNutAc =$ nutAc actualizada, añadiendo los nutrientes que aparecen al escoger a gramos del ingrediente po.ing(index).

- Esquema Recursivo:

$$sp(index, caAc, coAc, nutAc) = \begin{cases} null, & \text{si } !cumpleMinimos() \\ (null, 0.), & \text{si } cumpleMinimos() \\ \min_{a \in A} \{ cS(a, sp(index+1, caAc+a, Next(coAc, NextNutAc)) \} & \text{, en otro caso} \end{cases}$$

- $sA(a, (a', v)) = (a, v + a \times po.img(index).co)$
- sP: Elige la solución parcial con menor valor de v.
- objetivo: V : coAc.
- Solución reconstruida:
  - $sr(a, v) =$ Solucion.create(), es un caso base.
  - $sr(a, s) =$ s.addIngrediente(index, a)
    s.setCosteTotal(coAc + V), es caso recursivo.

- Complejidad: $O(mp)$

- Se adjuntan grafos: con filtro y sin filtro.
  - Para poder generarlo, tuve que reducir la cantidad máxima a 636, ya que la solución es la misma y así podemos obtener un grafo más pequeño con lo más representativo.

Ficha de descripción del problema mediante Backtracking.

- **Descripción:** Escoger el número de gramos a seleccionar de entre los ingredientes disponibles, sabiendo que solo podemos seleccionar un máximo de 1000 gramos, que hay que minimizar el coste total (cada ingrediente tiene coste) y que con la selección de ingredientes que realicemos hay que cumplir un mínimo de nutrientes existentes (cada ingrediente posee una cantidad de nutrientes por gramo).

- **Tipos:** S- Solución Alimentos
  A- Integer.

- **Propiedades compartidas:**
  - Problema Alimentos po.
    - List<Integer> minNut.
    - Integer caMax.
    - List<IngredienteActivo> ing.
      - List<Double> nut
      - Double co.
    - Integer ingN = ing.size().
    - Integer minNutN = minNutN.size()

- **Propiedades individuales:**
  - Integer index.
  - Integer caAc.
  - Double coAc.
  - List<Double> nutAc.
  - Integer caRe = po.caMax - caAc.
  - List<Integer> la.
  - Integer laN = la.size().

- **Solución:** Solución Alimentos.
- **Tamaño:** po.ingN - index + 1.

- Alternativas: $A = [0, caR]$

La alternativa a representa el número de gramos escogidos del ingrediente po.ing(index).

- Instanciación: Inicial = (0, 0, 0.0, nutAc, [])

Siendo nutAc una List<Double> con po.minNutN "ceros".

- Estado final: index = po.ingN.

- Avanza:

(index, caAc, coAc, nutAc, la) → a → (index+1, caAc+a, NextCoAc, NextNutAc, lata)

- NextCoAc = coAc + a * po.ing(index).co
- NextNutAc = nutAc actualizada, añadiendo los nutrientes que aparecen al escoger a gramos del ingrediente po.ing(index).

- Retrocede:

(index, caAc, coAc, nutAc, la) → a → (index-1, caAc-a, NextCoAc, NextNutAc, la-a)

- NextCoAc = coAc - a * po.ing(index).co.
- NextNutAc = nutAc actualizada, eliminando los nutrientes que desaparecen al eliminar a gramos del ingrediente po.ing(index).

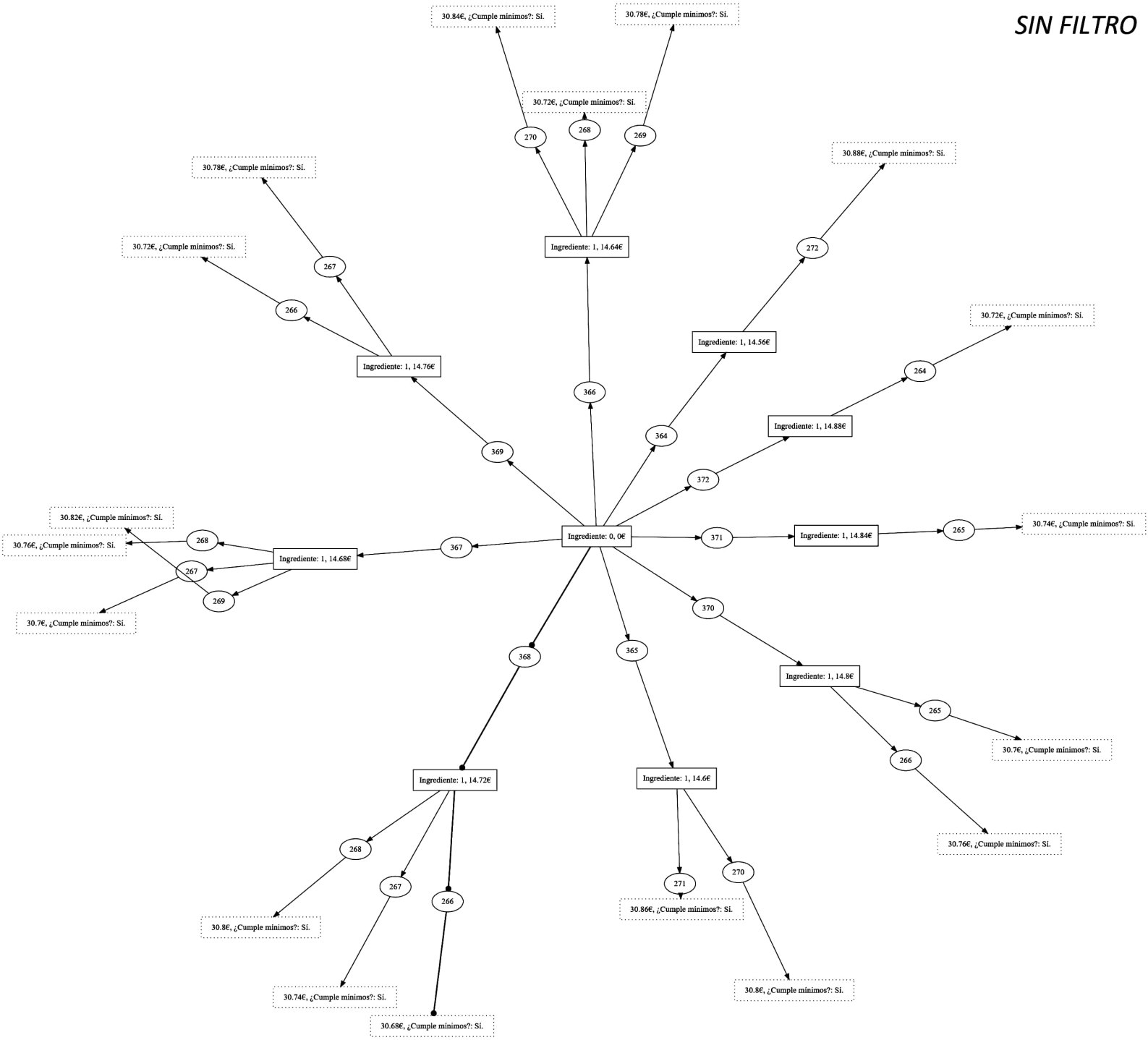
- Solución (para el estado final):

```
S = Solucion.create().
for(int i = 0; i < laN; i++){
    s.addIngrediente(i, la(i))
}
s.setCosteTotal(coAc)
```

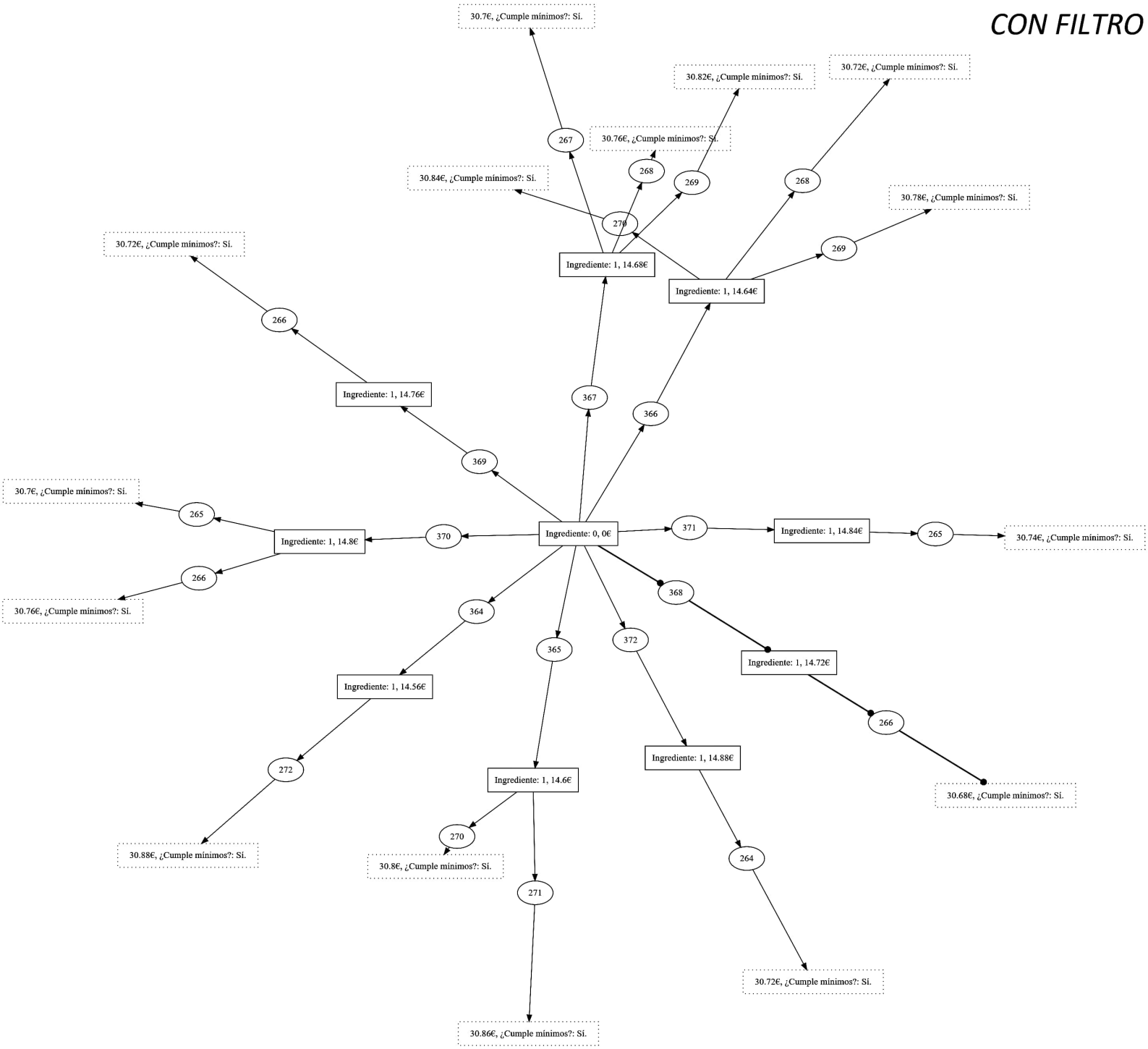
- Si (cumpleMinimos()): S
- Si (!cumpleMinimos()): null

- Objetivo: V: coAc.

SIN FILTRO


Ingrediente: 0, 0€
366
Ingrediente: 1, 14.64€
270
30.84€, ¿Cumple minimos?: Sí.
268
30.72€, ¿Cumple minimos?: Sí.
269
30.78€, ¿Cumple minimos?: Sí.
369
Ingrediente: 1, 14.76€
266
30.72€, ¿Cumple minimos?: Sí.
267
30.78€, ¿Cumple minimos?: Sí.
364
Ingrediente: 1, 14.56€
272
30.88€, ¿Cumple minimos?: Sí.
372
Ingrediente: 1, 14.88€
264
30.72€, ¿Cumple minimos?: Sí.
371
Ingrediente: 1, 14.84€
265
30.74€, ¿Cumple minimos?: Sí.
367
Ingrediente: 1, 14.68€
268
30.82€, ¿Cumple minimos?: Sí.
30.76€, ¿Cumple minimos?: Sí.
267
269
30.7€, ¿Cumple minimos?: Sí.
370
Ingrediente: 1, 14.8€
265
30.7€, ¿Cumple minimos?: Sí.
266
30.76€, ¿Cumple minimos?: Sí.
368
Ingrediente: 1, 14.72€
268
30.8€, ¿Cumple minimos?: Sí.
267
30.74€, ¿Cumple minimos?: Sí.
266
30.68€, ¿Cumple minimos?: Sí.
365
Ingrediente: 1, 14.6€
271
30.86€, ¿Cumple minimos?: Sí.
270
30.8€, ¿Cumple minimos?: Sí.

CON FILTRO


Ingrediente: 0, 0€
367
366
371
370
369
364
365
372
368
Ingrediente: 1, 14.68€
Ingrediente: 1, 14.64€
Ingrediente: 1, 14.8€
Ingrediente: 1, 14.76€
Ingrediente: 1, 14.84€
Ingrediente: 1, 14.56€
Ingrediente: 1, 14.6€
Ingrediente: 1, 14.88€
Ingrediente: 1, 14.72€
267
268
269
270
266
265
272
271
264
30.7€, ¿Cumple mínimos?: Sí.
30.82€, ¿Cumple mínimos?: Sí.
30.72€, ¿Cumple mínimos?: Sí.
30.76€, ¿Cumple mínimos?: Sí.
30.84€, ¿Cumple mínimos?: Sí.
30.78€, ¿Cumple mínimos?: Sí.
30.74€, ¿Cumple mínimos?: Sí.
30.76€, ¿Cumple mínimos?: Sí.
30.68€, ¿Cumple mínimos?: Sí.
30.88€, ¿Cumple mínimos?: Sí.
30.8€, ¿Cumple mínimos?: Sí.
30.86€, ¿Cumple mínimos?: Sí.