Semana 5 - Ejercicios de grupo

Métodos algorítmicos en resolución de problemas II
Facultad de Informática - UCM

	Nombres y apellidos de los componentes del grupo que participan	**ID Juez**
1	Jorge Arevalo Echevarria	MAR205
2	Daniel Hernández Martínez	MAR246
3	Miguel Verdaguer Velazquez	MAR285
4		

Instrucciones:

1. Para editar este documento, es necesario hacer una copia de él. Para ello:
    - Alguien del grupo inicia sesión con la cuenta de correo de la UCM (si no la ha iniciado ya) y accede a este documento.
    - Mediante la opción *Archivo* → *Hacer una copia*, hace una copia del documento en su propia unidad de *Google Drive*.
    - Abre esta copia y, mediante el botón *Compartir* (esquina superior derecha), introduce los correos de los demás miembros del grupo para que puedan participar en la edición de la copia.

2. La entrega se realiza a través del Campus Virtual. Para ello:
    - Alguien del grupo convierte este documento a PDF (dándole como nombre el número del grupo, 1.pdf, 2.pdf, etc…). Desde *Google Docs*, puede hacerse mediante la opción *Archivo* → *Descargar* → *Documento PDF*.
    - Esta misma persona sube el fichero PDF a la tarea correspondiente del *Campus Virtual*. Solo es necesario que uno de los componentes del grupo entregue el PDF.

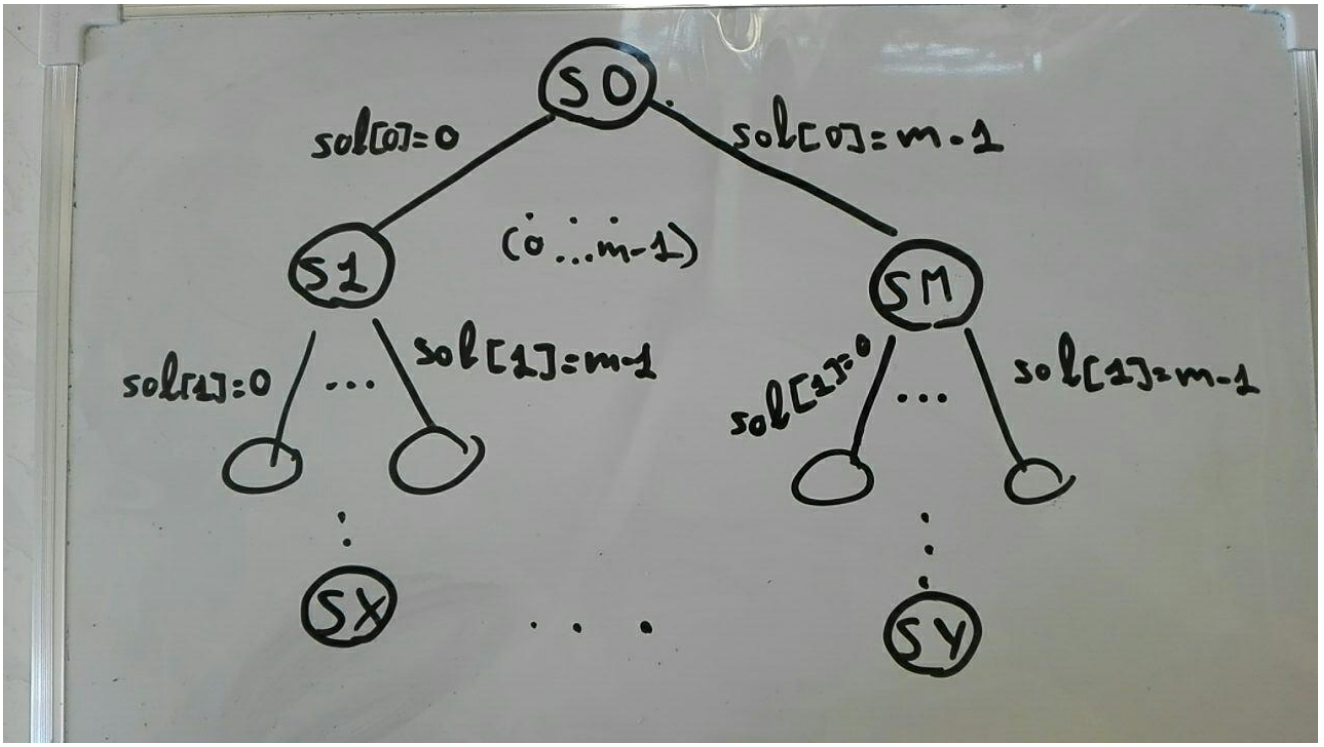
Compra de la semana

El objetivo de hoy es resolver el **problema 16 Compra de la semana**, del juez automático. Ahí podéis encontrar el enunciado completo del problema.

El problema consiste en comprar *n* productos en *m* supermercados teniendo en cuenta que todos los productos están disponibles en todos los supermercados y que han de comprarse a lo sumo tres productos en cada supermercado ($n \leq 3m$). Conociendo el coste de cada uno de los productos en cada uno de los supermercados se desea obtener el coste mínimo de la compra que satisface las restricciones del problema.

Solución: (Explicad cómo representáis la solución al problema, cuáles son las restricciones explícitas e implícitas del problema y representad mediante un dibujo el espacio de soluciones indicando cuál es su tamaño. Explicad qué marcadores vais a utilizar para mejorar el coste del test de factibilidad. Proporcionad y comparad dos posibles beneficios estimados que puedan ser utilizados como prioridad de la cola (en la implementación utilizad la que consideréis mejor).)

Representación de la solución: Una lista de longitud n, y cada elemento indica en que supermercado(0...m-1) se ha comprado el producto n.



El árbol de soluciones tiene profundidad N y factor de ramificación M. El tamaño del árbol es N^M.

Restricción explícita: cada elemento de la solución tiene que ser un entero entre 0 y m-1, que indica en que supermercado se ha comprado dicho producto.
Restricción implícita: que cada valor entre 0 y m-1 no se repita más de tres veces en la solución , es decir, que no se haya comprado más de 3 productos en un supermercado.

Marcador coste test de factibilidad: Una lista de contadores de producto por supermercado [0..m-1]

Dos posibles funciones de estimación:

- El coste acumulado por los productos comprados hasta el que estamos analizando.
- Sumar al coste acumulado el menor precio de cada producto que nos falte por analizar. Esto sin preocuparnos de que se cumpla la restricción de que no se compren más de 3 productos en cada supermercado. Esto es cota inferior ya que si algún producto se compra en un supermercado en el que no tenga el precio mínimo, el precio total será mayor que el de la estimación. Además si todos los productos restantes se compran en supermercados con el precio mínimo, el total coincidirá con la estimación, pero nunca podrá ser menor. Por esto, y debido a que el valor de esta estimación se acerca más al precio final que la primera, esta función es mejor para utilizarse como estimación y será la que usaremos en nuestra implementación.

Solución: (Escribid aquí las explicaciones necesarias para contar de manera comprensible la implementación del algoritmo de ramificación y poda. Incluid la definición del tipo de los nodos, y el código. Extended el espacio al que haga falta.)

```
#include <vector>
#include <iostream>
#include <queue>
#include <limits>
using namespace std;

struct Nodo{
  vector<int> sol;
  int k;
  double coste; // coste acumulado
  double coste_estimado; // estimación, prioridad
  bool operator<(Nodo const& otro) const{
    return otro.coste_estimado< coste_estimado;
  }
};


int min(vector<int> v) {
  int min = numeric_limits<int>::infinity();
  for(int i: v) {
    if(i < min) min = i;
  }
  return min;
}
```

```
double calculo_estimacion(vector<int> const& sumaminsupers, Nodo const& X) {
  return sumaminsupers[X.k]+X.coste;
}

vector<int> precalculo(vector<vector<int>> const& supermercados, int N, int M) {
  //Sacamos el minimo precio de cada producto
  vector<int> sumamins = vector<int>(N);
  vector<int> mins = vector<int>(N, numeric_limits<double>::infinity());
  for(int i = 0; i < N; i++) {
    for(int j = 0; j < M; j++) {
      mins[i] = min(supermercados[j][i], mins[i]);
    }
  }
  //Acumulamos los minimos de atras hacia delante
  sumamins[mins.size()-1] = mins[mins.size()-1];
  for(int i = mins.size()-2;i >= 0;i--) {
    sumamins[i] = mins[i] + sumamins[i+1];
  }
}

//supermercados son los valores iniciales y sumaminsupers es la suma acumulada de los minimos precios de cada producto, contadores es el marcados, de longiud M e inicializado a 0
void supermercados_rp(vector<vector<int>> const& supermercados,vector<int> & sol_mejor, double& coste_mejor, vector<int> & contadores, vector<int> const& sumaminsupers, int M, int N) {
  Nodo Y;
  Y.sol = std::vector<int>(N, -1);
  Y.k = -1;
  Y.coste= 0;
  Y.coste_estimado= calculo_estimacion(sumaminsupers, Y);
  priority_queue<Nodo> cola;
  cola.push(Y);
  coste_mejor= numeric_limits<double>::infinity();
  while(!cola.empty() && cola.top().coste_estimado < coste_mejor) {
    Y = cola.top(); cola.pop();
    Nodo X(Y);
    ++X.k;
    for(int i = 0; i < M;i++) {
      if(contadores[i] < 3) {
      X.sol[X.k] = i;
      X.coste = Y.coste + supermercados[i][X.k];
      X.coste_estimado= calculo_estimacion(sumaminsupers, X);
      if(X.coste_estimado < coste_mejor)
            if(X.k == N-1) {
              sol_mejor= X.sol;
              coste_mejor= X.coste;
            } else
              cola.push(X);
      }
    }
```

✓ 1

2

OJO, si hacen aquí la copia del nodo (que está OK) entonces hay que desmarcar porque si no se acumulan los cambios en los hijos

debe ir en el nodo

```
  }
}
```

Resolved el problema completo del juez, que ahora debe ser ya muy sencillo.
Número de envío: