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	Borja Aday Guadalupe Luis	MAR241
2	Óscar Morujo Fernández	MAR262
3	Roberto Plaza Hermira	MAR267
4	Pablo Martínez	MAR258

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).)

Sea N el número de productos a comprar y M el número de supermercados existentes.

Restricciones implícitas:

- Como máximo se pueden comprar 3 productos en cada supermercado. De ello se deriva (N <= 3M).

Restricciones explicitas:

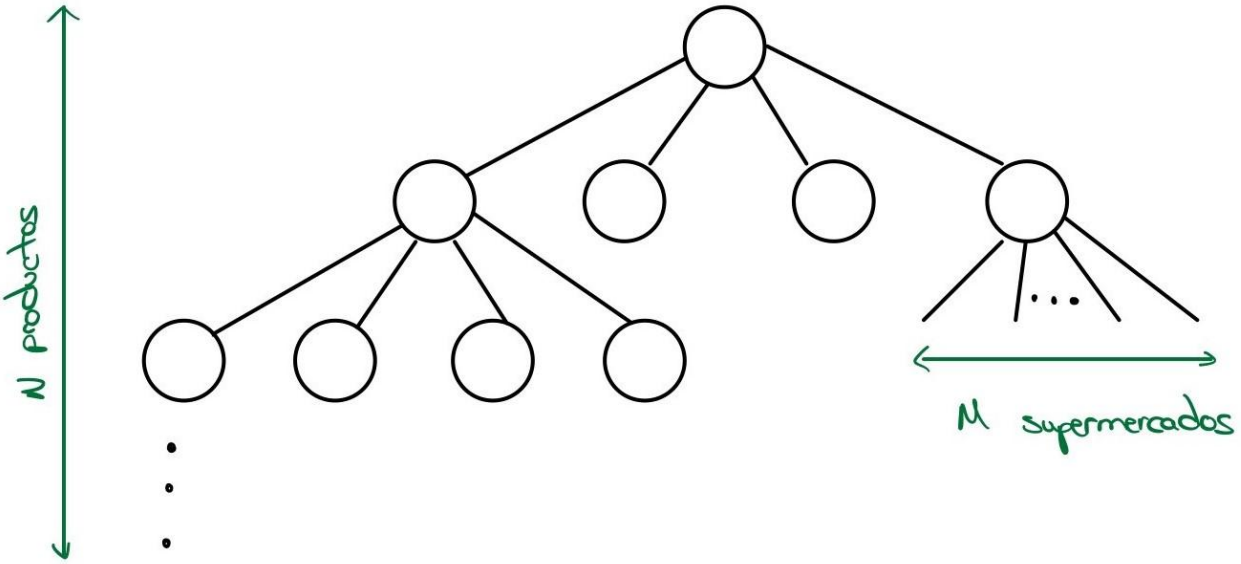
- El vector de soluciones será de tamaño N, para cada producto se almacena el supermercado en el que se ha comprado.
- Para mejorar la eficiencia del test de factibilidad tambien usaremos un vector de tamaño m como marcaje, por ejemplo marcaje[i] nos indicara el número de productos que se pueden comprar en el supermecado i. Inicialmente marcaje[i] = 3 para todo 0 <= i < M.
- De ello se deriva que en el vector de soluciones no podemos encontrar más de 3 veces el mismo número, dado que indicaría que hemos comprado más de 3 productos en un mismo supermercado.

Espacio de soluciones :

- El tamaño máximo del espectro de soluciones (número de nodos) es: $\frac{M^{N+1}-1}{M-1}$.
- De cada nodo salen como máximo M ramas (cada supermercado en el que puedo comprar el siguiente producto).

- Esto se debe a que en el caso peor tendremos un árbol M-ario completo. En un caso promedio, es probable que en algún punto del árbol se haya descartado algún supermercado por el hecho de que ya se hayan comprado 3 productos en él, por lo que esa rama no sería factible.

- El árbol de soluciones tiene altura N (número de productos).



Beneficios estimados:

Para este ejercicio, al tratarse de una minimización necesitamos encontrar una cota inferior.

- Cota inferior 1: Coste actual + 0.

Sabemos que es una cota inferior, pero es muy optimista, damos por hecho que no vamos a gastar ni un céntimo más en comprar los N - k - 1 productos restantes.

- Cota inferior 2: Coste actual + (N - k - 1) * minCoste.

Sabemos que esto sigue siendo una cota inferior, pues damos por hecho que los N - k - 1 productos restantes cuestan todos lo mismo que el producto más barato en cualquier supermercado. Esta es una cota más ajustada.

- Cota inferior 2: Coste actual + mínimo[i] $k < i < N$.

Esta cota es mucho más realista, para cada producto que nos falta asumimos que vamos a poder comprar el más barato, esta será la mayor cota inferior de las 3 propuestas por lo que usaremos esta cota para la poda.

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.)

```
struct Nodo {
    vector<int> sol;
    int k;
    vector<int> marcaje;
    double beneficio; // valor acumulado
    double beneficio_est; // prioridad
    bool operator<(Nodo const& otro) const {
        return otro.beneficio_est < beneficio_est;
    }
};

int beneficio_estimado(vector<int> minimos, Nodo Y) {
    if (Y.k == -1) return minimos[minimos.size() - 1];
    return Y.beneficio + minimos[minimos.size() - 1] - minimos[Y.k];
}
```

En el nodo necesitamos tener guardada distinta información:

- K: Índice del objeto sobre el que estamos decidiendo en qué supermercado comprarlo.
- Marcaje: Vector que nos indica cuantos objetos podemos comprar en cada supermercado.
- Beneficio: Valor actual de la solución parcial.
- Beneficio estimado: Aproximación a la solución final dada la solución parcial (se usará para la posterior poda).
- Definimos la prioridad entre nodos en función del beneficio estimado, y como queremos minimizar, damos prioridad en la cola de nodos al que tenga menor coste estimado.

Para la mejora del cálculo del coste estimado, hemos utilizado el vector de mínimos, en dicho vector, la posición mínimos[i] nos indica el coste estimado de comprar los productos desde el 0 hasta el i. (Entendemos que los objetos están numerados del 0 al N - 1).

De este modo:

- cuando la K del nodo es 0, el coste estimado es mínimos[N - 1].
- Cuando la K != 0, el coste estimado es el coste real actual + mínimos[N - 1] + mínimos[Y.k].
  Esto se debe a que los primeros k ya se han tenido en cuenta en el coste real y solo hay que tener en cuenta los mínimos desde el K hasta el N - 1.

Con esta mejora hemos conseguido que el cálculo del coste estimado sea constante, habiendo realizado un pequeño cálculo previo, el vector de mínimos.

```
for (int i = 0; i < mSuper; i++) {
    for (int j = 0; j < nProd; j++) {
        cin >> datos[i][j];
        if (datos[i][j] < minimos[j]) minimos[j] = datos[i][j];
    }
}

for (int i = 0; i < minimos.size() - 1; i++) minimos[i + 1] += minimos[i];
```

```
void compra_rp(vector<vector<int>> const& objetos, int m,
    vector<int> sol_mejor, vector<int> minimos, int& beneficio_mejor) {
    int N = minimos.size();
    // se genera la raíz
    Nodo Y;
    Y.marcaje = vector<int>(m, 3);
    Y.sol = vector<int>(N);
    Y.k = -1;
    Y.beneficio = 0;
    Y.beneficio_est = beneficio_estimado(minimos, Y);
    priority_queue<Nodo> cola;
    cola.push(Y);
    beneficio_mejor = INT_MAX;

    // búsqueda mientras pueda haber algo mejor
    while (!cola.empty() && cola.top().beneficio_est < beneficio_mejor) {

        Y = cola.top();
        cola.pop();
        Nodo X(Y);
        ++X.k;

        // probamos a comprar el objeto en cada supermercado.
        for (int i = 0; i < m; i++) {
            if (X.marcaje[i] > 0) {
                X.marcaje[i]--;
                X.sol[X.k] = i;
                X.beneficio = Y.beneficio + objetos[i][X.k];
                X.beneficio_est = beneficio_estimado(minimos, X);
                if (X.beneficio_est < beneficio_mejor) {
                    if (X.k == N - 1) {
                        if (X.beneficio < beneficio_mejor) {
                            sol_mejor = X.sol;
                            beneficio_mejor = X.beneficio;
                        }
                    }
                    else cola.push(X);
                }
                X.marcaje[i]++;
            }
        }

    }
}
```

En cuanto al código solo cabe destacar que para cada nodo generamos los M nodos siguientes (comprar el objeto siguiente en cada uno de los supermercados posibles) dándole el marcaje que necesita a cada nodo.

Cuando ya hemos comprado tres productos en un supermercado, su marcaje será igual a 0 y por tanto no seguirá generando hijos, tal y como nos indica las restricciones implícitas.

La poda se realiza al comparar que el beneficio estimado sea menor que el mejor que tengo hasta el momento, pues en caso de que el estimado sea mayor al mejor que tengo guardado, sabemos que podemos descartar ese nodo ya que la estimación es una cota inferior y nuestro coste será mayor que el estimado, y por tanto mayor que el mejor que tenemos en ese momento. Es por ello que esa rama entera puede ser podada.