PRÁCTICA 1 – Amazon + Amazonymás

## ALGORITMIA PARA PROBLEMAS DIFÍCILES

### Francisco Ferraz 737312 – Félix Bernal

# Resumen

En esta práctica se ha implementado el algoritmo de Krager-Stein para resolver el problema de mínimo corte sobre la división de productos en dos conjuntos para minimizar las pérdidas económicas de una empresa virtual teniendo en cuenta la relación entre los mismos.

# Desarrollo

### Generador aleatorio

En primera instancia, se implementó para la primera tarea un generador de datos de prueba aleatorios sobre la idea de que dos productos podían haber sido comprados juntos o no, sin tener en cuenta el número de veces que esto había sucedido. Esto se realiza desde el fichero *matrixGenerator.cpp*, donde se genera un número K de ficheros con matrices de dimension NxN de 1s y 0s para una serie de probabilidades dadas de que dos productos hayan sido comprados juntos, probando con distintas probabilidades (0.1, 0.2, 0.3...). Las matrices son simétricas por representar la relación entre los productos (M[i][j] == M[j][i]).

### HashMap

Se ha implementado desde cero una clase HashMap para poder guardar los datos de cada producto. Su única limitación es que solo admite que todos los elementos sean del mismo tipo de dato, por lo que se recomienda utilizar el tipo *string* del estandar de C++. Su implementación se encuentra en los ficheros *hashMap.hpp* y *hashMap.cpp*.

### Análisis del problema

Para entender el problema como un problema de mínimo corte, tenemos que trasladar nuestro problema a grafos. En este caso, supondremos que los **productos** son los **vértices** del grafo y, si dos productos han sido comprados juntos (**valor 1** en la matriz) se establecerá una **arista** entre ellos.

El problema de mínimo corte establece que, para un grafo, se encuentra la separación de sus vértices en dos grupos de tal modo que se pierde el menor número de aristas. En nuestro caso, esto implica separar los productos en las dos divisiones de la empresa minimizando las pérdidas por productos relacionados.

### Algoritmo de Krager-Stein

//Completar

### Generador aleatorio con valores

De cara al apartado opcional, se ha implementado también un generador de matrices de enteros aleatorias (con valores de 0 a 10, pero valdría cualquier otro caso) para poder probar el algoritmo de Krager-Stein con matrices con valor. Está implementado en el fichero *matGenValue.cpp*.

### Modificación de Krager-Stein

La única diferencia de cara a la construcción de un grafo como los anteriores y un grafo con valores es aumentar el número de aristas entre los vértices. Es decir, si antes encontrar un valor 1 en la matriz nos hacía trazar una arista, ahora encontrar un valor 3 nos hará trazar 3 aristas. De este modo, el algoritmo resuelve automáticamente y sin modificar su comportamiento el problema.

### Justificación temporal

//Completar

### Banco de pruebas

//Completar