PRÁCTICA 1: ALGORITMIA PARA PROBLEMAS DIFÍCILES

Javier Fumanal Idocin, NIP: 684229

CONTENIDO

FICHEROS ENTREGADOS

En la entrega se incluyen:

- Una carpeta src con todos los ficheros de código fuente en Java.
- Una carpeta pruebas con tres ficheros .txt con distintos ficheros de prueba.
- Un script ejecutar.sh que compila el código en un fichero P1.jar y ejecuta dos casos de prueba.
- Fichero LEEME con información sobre las opciones de ejecución del programa.

ANÁLISIS DEL PROBLEMA

El problema consiste trazar un corte entre los productos de Amazon para separarlos en dos grupos, de modo que la suma de las ventas conjuntas que se hacen en productos de ambos conjuntos sea mínima. Para resolverlo, se plantea cada producto como un nodo en un grafo no dirigido, que se conecta a cada otro producto con el que haya sido comprado conjuntamente. El peso de esta conexión es igual al número de veces que tales productos han sido comprados juntos.

Una vez construido el grafo, se puede identificar el problema fácilmente como un problema de corte mínimo, y se utiliza tanto el algoritmo de Stein como el de Karger-Stein para resolverlo.

DISEÑO DEL PROGRAMA

Para resolver el problema, se ha estructurado el programa en 3 partes: un paquete grafo, donde creamos un grafo con las prestaciones que necesitamos; un paquete random para probar distintos generadores de números aleatorios y un paquete main desde el cargar los ficheros de productos y ejecutar el algoritmo de resolución del problema.

PAQUETE GRAFO

El paquete grafo consta de tres clases: Grafo, Identificable y Nodo.

- Identificable: es una interfaz con tres métodos. Uno para que devuelve un identificador único del objeto, otro para combinar dos identificables y un tercero para crear un identificable a partir de otro. Estas son las tres condiciones que necesitamos para aplicar el algoritmo Karger y Karger-Stein en el grafo. (La clase Producto implementa esta interfaz).
- Nodo: es una clase que contiene un elemento que debe implementar Identificable. Además, posee una tabla hash que lista las conexiones que posee este nodo con el resto de nodos del grafo.

- Grafo: es un grafo no dirigido y con pesos de nodos con objetos Identificables. Permite combinar nodos, y manipula las conexiones entre nodos dejando siempre un estado coherente entre las mismas. También permite acceder a los nodos que lo conforman de forma aleatoria.

PAQUETE RANDOM

Para el paquete random se han creado tres clases, cada una de ellas genera números aleatorios utilizando un generador distinto: el generador estándar de Java (clase VanillaRandom), el de la biblioteca de criptografía de Java (CryptoRandom) y una versión más segura en esta misma librería (SecuredRandom).

El último de estos generadores no está disponible en versiones anteriores a Java 1.8.

PAQUETE MAIN

En el paquete main hay dos clases: Main y Producto.

- Producto: clase que implementa a identificable y que guarda los datos relacionados con cada producto.
- Main: clase que contiene los métodos relacionados con la lectura de los ficheros de prueba y los algoritmos Karger y Karger-Stein. Permite escoger entre el uno o el otro mediante un argumento por la línea de comandos. También permite escoger qué generador aleatorio de número queremos utilizar.

ALGORITMOS

KARGER

Para implementar el algoritmo de Karger se han implementado las operaciones básicas que requiere el mismo: combinar nodos y elección aleatoria de los mismos, en la clase Grafo.

Para combinar nodos, elegimos uno de los dos y añadimos todas las conexiones que posee uno en el otro. A su vez, conectamos todos aquellos que estaban conectados con este con el otro, de modos que el nodo resultante esté conectado de forma bidireccional con todos los nodos con los que estaba conectado anteriormente y con los del otro nodo. Una vez hecho eso, utilizamos la operación añadir() de los objetos Identificables para combinar los elementos contenidos en ambos nodos. Una vez hecho esto, borramos el nodo que hemos elegido del grafo, borrándolo de la lista de nodos y eliminando todas sus conexiones hacia y desde el resto de nodos.

Para elegir el nodo inicial, hacemos una elección aleatoria. Una vez tenemos este nodo, queremos escoger para mezclarlo con él a otro conectado con él. Como queremos minimizar además el peso del corte, buscamos darle más posibilidad de mezcla a aquellos nodos cuya conexión con este tenga más peso (que en este contexto serían los productos más comprados conjuntamente con el elegido). Para ello, creamos una lista de “participaciones”.

Una lista de participaciones consiste en una lista de enteros, que son los identificadores de los nodos que son candidatos a la mezcla. Cada identificador aparece en la lista un número de veces igual al peso de su conexión con el nodo elegido. De la lista, se elige al azar un elemento de la misma, que será el nodo a mezclar. De esta forma, se aumenta la posibilidad de ser elegido cuanto mayor es el peso de la conexión con el nodo elegido y viceversa.

Repetimos el proceso durante n-2 veces, hasta que sólo dos nodos finales, que serán los dos conjuntos que haya calculado

KARGER-STEIN

Karger-Stein funciona de forma similar al algoritmo anterior. En vez de reducir hasta dos conjuntos finales de elementos, se ejecutan combinaciones de nodos hasta dejar $\frac{N}{\sqrt{2}}$, donde N es el número de nodos inicial del grafo. Una vez se reduce el grafo a ese tamaño, se ejecuta recursivamente el algoritmo dos veces distintas, llegando hasta el caso base o reduciendo el grafo a dos nodos, y se comparan ambos y se devuelve el mejor. Este comportamiento permite reducir nuestra dependencia en la suerte a la hora de elegir nodo y nos permite descartar con facilidad resultados anormalmente malos que sí podrían ocurrir en el algoritmo Karger que hemos visto antes.

Para implementar el Karger-Stein necesitamos una operación adicional, que es la de crear una copia completa tanto de grafos, como de nodos e identificables. Una vez reducimos al tamaño que queremos, creamos dos copias del grafo resultante, y aplicamos el algoritmo sobre ellas de forma recursiva. Luego comparamos cuál de los dos resultados es el mejor, y devolvemos ese como resultado del algoritmo.

Para aplicar el algoritmo recursivo, elegimos un tamaño de grafo como caso base: 3. Para resolver el Corte Mínimo con este número utilizamos directamente la fuerza bruta, probamos todas las combinaciones y devolvemos el mejor resultado. En caso de que el grafo tenga ya tamaño 2, lo devolvemos sin modificar como resultado.

PRUEBAS

Para hacer las pruebas se ha utilizado el fichero de pruebas Grande.txt con 37 artículos distintos (hecho a mano). Se han probado los algoritmos Karger y Karger-Stein, con los tres generadores aleatorios distintos. Se mide el tiempo de ejecución y la calidad del corte.

Generadores	Karger (Tiempo ms)	Karger-Stein (Tiempo ms)	Karger (Coste)	Karger-Stein (Coste)
Estándar	665	504	20	7
Lib. Criptografía	1075	230	13	7
Lib. Criptografía Segura	202	227	7	7

REFERENCIAS

Sobre el algoritmo Karger y Karger-Stein:

https://en.wikipedia.org/wiki/Karger's_algorithm

http://www.cc.gatech.edu/~vigoda/7530-Spring10/Kargers-MinCut.pdf