**Pontificia Universidad Católica Madre y Maestra**

**Facultad de Ciencias de la ingeniería,**

**Departamento de Ingeniería en Sistemas y Computación**

**Diseño y análisis de Algoritmos**



Trabajo:

T1-Corte Mínimo

Materia:

Diseño y análisis de Algoritmos

ST-ISC-405-T-001

Estudiantes:

Francis Cáceres

2012-0559

Profesor:

Juan R. Núñez

Introducción

Se nos pide codificar y ejecutar el algoritmo de contracción aleatorizado para el problema de corte mínimo y utilizarlo en grafo para calcular el corte min. (SUGERENCIA: Tenga en cuenta que debes averiguar una implementación de contracción de aristas. Inicialmente, es posible que desee hacer esto, ingenuamente, la creación de un nuevo grafo a partir del viejo cada vez que hay una contracción de aristas. Pero también se debe pensar más en implementaciones más eficiente... i) (ADVERTENCIA:.. favor asegúrese de ejecutar el algoritmo muchas veces con diferentes semillas aleatorias, y recordar el corte más pequeño que alguna vez se encontró). De una respuesta de este valor encontrado.

El algoritmo Karger es un algoritmo probabilístico para el problema de la reducción mínima. Este es un algoritmo utilizando una fuente de peligros, para producir una solución correcta con una buena probabilidad.

Desarrollo

Tenemos un archivo el cual contiene la representación como lista de adyacencia de un grafo no dirigido sencillo. Hay 200 vértices etiquetados de 1 a 200. La primera columna del archivo representa la etiqueta vértice, y la fila en particular (las otras entradas, excepto la primera columna) le dice a todos los vértices de los que el vértice es adyacente a. Así, por ejemplo, la sexta fila se ve así: "6 155 56 52 120 ......". Esto sólo significa que el vértice con etiqueta 6 es vecino de (es decir, tiene arista con) los vértices con etiquetas 155,56,52,120, ......, etc…

El algoritmo se basa en un procedimiento de contracción borde. Este procedimiento es, dado un grafo G y una e = borde de fusionar a y b para formar un solo tapa ab cuyos vecinos son la concatenación de los vecinos de a y b. El gráfico formado se denota G / e. Esta transformación puede revelar múltiples bordes, que está trabajando con dispositivos multiplicadores. El punto importante es que un corte en el gráfico después de la contracción ya era un corte en el gráfico original.

El algoritmo es simplemente contratar a un borde dibujado de manera uniforme en el gráfico con el contrato, y repetir el proceso hasta que sólo tienen dos picos. Los dos picos representan la partición de vértices y los bordes entre ellos se encuentran los bordes de corte.

Para solucionar el problema planteado se utilizó el algoritmo de minCut:

public static int minCut( Graph gr ) {  
  
 Random rnd = new Random();  
  
 while( gr.vertices.size() > 2 ) {  
 Edge edge = gr.edges.remove( rnd.nextInt( gr.edges.size() ) );  
 Vertex v1 = *cleanVertex*( gr, edge.ends.get( 0 ), edge );  
 Vertex v2 = *cleanVertex*( gr, edge.ends.get( 1 ), edge );  
 //contract  
 Vertex mergedVertex = new Vertex( v1.lbl );  
 *redirectEdges*( gr, v1, mergedVertex );  
 *redirectEdges*( gr, v2, mergedVertex );  
  
 gr.addVertex( mergedVertex );  
 }  
 return gr.edges.size();  
}

Y para la contracción:

private static void redirectEdges( Graph gr, Vertex fromV, Vertex toV ) {  
 for (Iterator<Edge> it = fromV.edges.iterator(); it.hasNext(); ) {  
 Edge edge = it.next();  
 it.remove();  
 if( edge.getOppositeVertex( fromV ) == toV ) {  
 //remove self-loop  
 toV.edges.remove( edge );  
 gr.edges.remove( edge );  
 } else {  
 edge.replaceVertex( fromV, toV );  
 toV.addEdge( edge );  
 }  
 }  
}

Con la ayuda de otras funciones. Para un mejor vistazo al código completo puede visitar este link (El código fuente está basado en Java):

<https://github.com/SuitMonkey/tarea1/blob/master/src/MinCut.java>

Para la solución lo corrí 25 veces y este fue el resultado:

