UGR

Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones
Grado en Ingeniería Informática

Asignatura: Algorítmica

Práctica 3 (Primera parte): Recubrimiento de un grafo no dirigido

---

Autores:
Míriam Mengíbar Rodríguez
Néstor Rodríguez Vico
Ángel Píñar Rivas

Granada, 13 de abril de 2016

Índice:

1. Problema 4: Recubrimiento de un grafo no dirigido.

Para resolver este problema, hemos diseñado un algoritmo que se basa en el grado de cada nodo. El grado de un nodo es el número de aristas que inciden en dicho nodo.

1.1. Descripción del algoritmo para grafos.

Consideramos el grafo G=(V, E) donde V es el conjunto de sus nodos y E el conjunto de sus aristas. U será el conjunto de nodos solución a nuestro problema.
Para el diseño de nuestro algoritmo vamos a trabajar con el nodo de mayor grado. Podemos usar esta idea como función de selección del algoritmo Greedy. El por qué de esta idea es fácil de razonar, ya que a mayor grado de un nodo, mayor número de aristas incidentes y por tanto hay mayor recubrimiento.

Los elementos de nuestro algoritmo son:

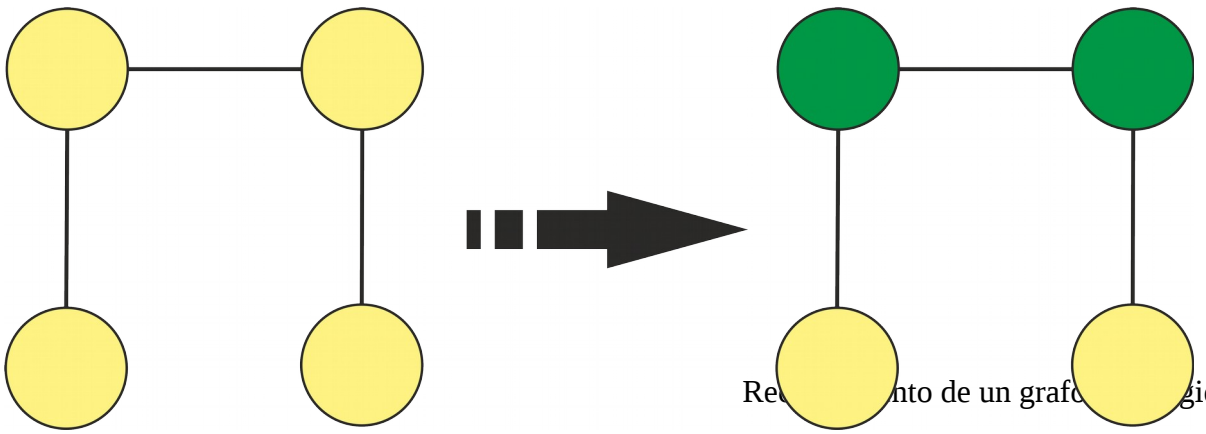
- Conjunto de candidatos: Nodos.
- Conjunto de seleccionados: Nodos de mayor grado:
- Función solución: El grafo tenga aristas.
- Función de factibilidad y función selección: Selección del nodo de mayor grado.
- Función objetivo: Conjunto de nodos en el que inciden todas las aristas del grafo.

Los pasos a seguir son:
Mientras el conjunto de las aristas no sea vacío, seleccionamos el nodo de máximo grado, v, entre los nodos restantes de V y lo añadimos a U. Para cada arista incidente, arista, en el nodo seleccionado, v, borramos la arista, arista, del conjunto E. Para evitar coger el mismo nodo de mayor grado, borramos el nodo escogido, v, del conjunto de nodos V.

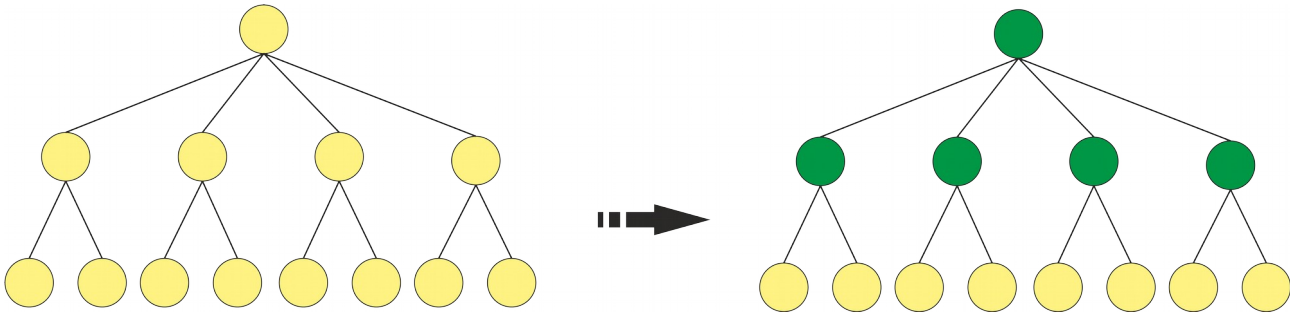
El pseudocódigo es el que sigue:

```
U = conjunto[]
while E tenga aristas {
   v = nodoMaximGrado de V;
   U.añadir(v);
   for arista ∈ aristasIncidentes de V {
      E.eliminar(arista);
   }
   V.eliminar(v);
}
```

Ejemplo:



Este algoritmo es correcto. Obtiene un recubrimiento pero no es óptimo. Un contraejemplo puede ser el que sigue:



1.2. Solución para árboles.

En el caso particular de los grafos que son árboles, podemos aprovechar la estructura que tienen para diseñar un algoritmo más específico.
En un principio, la idea era recorrer el árbol por niveles, comenzando desde el padre e ir añadiendo sus hijos. El problema es que tras la primera iteración, pasaríamos a tener varios arboles, ya que perderíamos la raíz original. Por lo tanto, cambiamos nuestro punto de vista y pensamos un algoritmo por niveles pero de forma ascendente (empezando por el último nivel hasta llegar a la raíz).

Para ello nos basaremos en el siguiente propiedad:
Sea A=(V, E) un árbol. Existe un recubrimiento minimal que no contiene a ninguna hoja de A pero sí a todo padre de una hoja.

Elementos del algoritmo para árboles:

- Conjunto de candidatos: Nodos.
- Conjunto de seleccionados: Nodos hoja del último nivel.
- Función solución: El grafo tenga más de un nodo.
- Función selección: Selección de padre de nodos hoja de último nivel.
- Función objetivo: Conjunto de nodos en el que inciden todas las aristas del grafo.
- Función de factibilidad: Nodos seleccionados tenga padre.

Vamos a calcular un recubrimiento minimal para un árbol A . Para ello usaremos la idea anterior. El algoritmo desarrollado es el siguiente:

```
//hojas_ult_nivel es un conjunto en el que almacenamos
//las hojas del ultimo nivel
U = conjunto[]
hojas_ult_nivel = conjunto[]
while A más de un nodo {
   hojas_ult_nivel = hojasUltimoNivel de A;
   for hoja ∈ hojas_ult_nivel {
      U.añadir(padre(hoja))
      A.eliminar(padre(hoja))
      A.eliminar(hoja)
```

```
    }
}
```

La idea es la siguiente; realizamos un recorrido del árbol por niveles, de forma ascendente. En cada iteración calculamos las hojas correspondientes en al nivel en el que estamos y añadimos los padres de dichas hojas al conjunto recubrimiento, U. A continuación eliminamos las hojas, sus padres y las aristas incidentes de dichos nodos, y se repite el proceso hasta que el árbol se quede sin nodos.

Esta implementación del algoritmo nos proporciona un recubrimiento minimal.