Algorítmica: práctica 3

Encontrar un recubrimiento minimal de un grafo

Sofía Almeida Bruno
Antonio Coín Castro
María Victoria Granados Pozo
Miguel Lentisco Ballesteros
José María Martín Luque

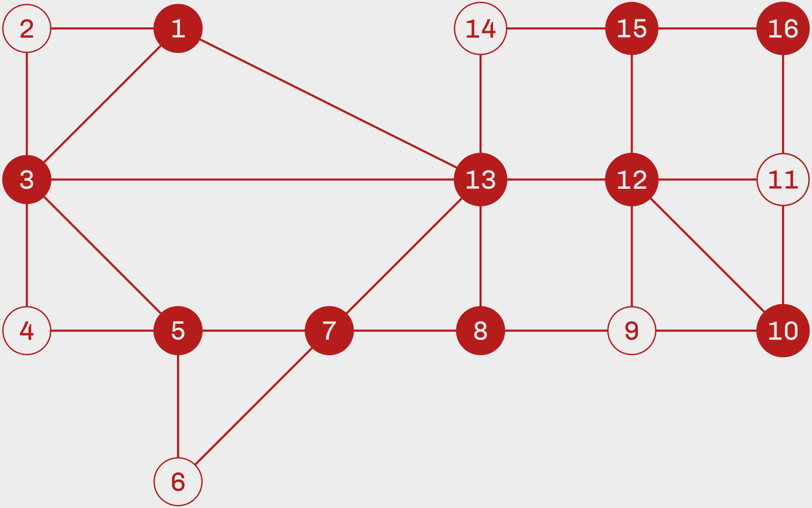
Grupo 2

25 de mayo de 2017

Objetivo

Encontrar un recubrimiento minimal del grafo no dirigido $G = (V, E)$. Un conjunto $U \subseteq V$ es un recubrimiento de G si cada arista en E incide en, al menos, un vértice o nodo de U. La solución que proporcionamos es el conjunto de nodos que forman el recubrimiento junto con el coste (número de nodos).

Ejemplo



Componentes Greedy

- *Lista de candidatos:* nodos del grafo
- *Lista de candidatos utilizados:* nodos considerados
- *Función solución:* no haya ninguna arista sin considerar
- *Criterio de factibilidad:* el nodo no está en la lista de candidatos utilizados
- *Función objetivo:* recubrimiento de coste mínimo
- *Función de selección:* nodo en el que inciden más aristas

Código fuente 1: Función de selección

```c
/**
 * Función de selección. Devuelve el nodo con mayor número
 * de incidencias de la lista de candidatos, cuyo tamaño es N.
 */
int FSeleccion(int* LC, int N) {
  int pos_max = 0;
  for (int i = 1; i < N; i++) {
    if (LC[i] > LC[pos_max])
      pos_max = i;
  }
  return pos_max;
}
```

Pseudocódigo

$N \equiv$ número de nodos del grafo
$M \equiv$ número de nodos del recubrimiento
$L \equiv$ matriz de adyacencia
$T[1, \ldots, M] \equiv$ vector de nodos del recubrimiento
$V[1, \ldots, N] \equiv$ vector de incidencias

```
function RecubrimientoGrafoGreedy (L[1, . . . , N][1, . . . , N] )
    T = ∅                                                        ▹ Recubrimiento
    for i = 1, . . . , N do
        V[i] ← Σ_{j=1}^{N} L[i][j]                               ▹ Número de incidencias del nodo i
    end for
    p ← Nodo con más incidencias (V[p] ≥ V[i]  ∀i)               ▹ Función de selección

    while V[p] > 0 do                                            ▹ Función solución
        T ← T ∪ {p}
        V[p] ← 0
        for j = 1, . . . , N do
            if Están conectados p y j and V[j] > 0  then
                V[j] ← V[j] – 1
            end if
        end for
        p ← Nodo con más incidencias (V[p] ≥ V[i]  ∀i)           ▹ Función de selección
    end while

    return T
end function
```

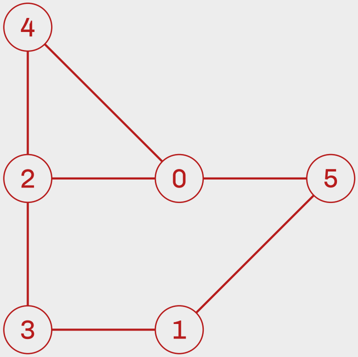
Algoritmo

```cpp
Solucion RecubrimientoGrafoGreedy(const Problema& p) {

  Solucion sol; // Solución a devolver
  int num_nodos = p.getNumNodos();  // Número de candidatos sin utilizar
  int incidencias[num_nodos];  // Vector de incidencias de cada nodo (LC)
  int pos_max;

  // Inicializar la lista de candidatos (LC)
  for (int i = 0; i < num_nodos; ++i)
    incidencias[i] = p.getNumIncidencias(i);

  // Nodo con más incidencias
  pos_max = FSeleccion(incidencias, num_nodos);
```

Algoritmo

```
  while (incidencias[pos_max] > 0) { // Mientras el vector de incidencias no sea {0,0,...,0}
    // Añadir el nodo con más incidencias a la solución (siempre es factible)
    sol.addNodo(pos_max);

    // Eliminar el nodo de la LC
    incidencias[pos_max] = 0;

    // Decrementar número de incidencias de nodos conectados con el seleccionado
    for (int j = 0; j < num_nodos; ++j) {
      if (p.estanConectados(pos_max,j) && incidencias[j] > 0)
        --incidencias[j];
    }

    // Seleccionar nodo con más incidencias
    pos_max = FSeleccion(incidencias, num_nodos);
  }

  return sol;
}
```

Ejemplo paso a paso



Inicialización

N = num_nodos = 6

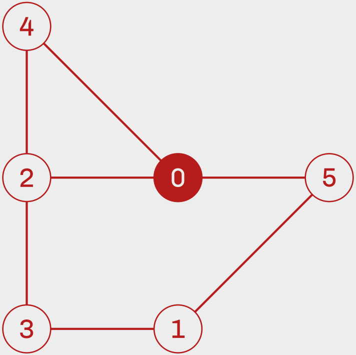
M = sol.coste = 0

$$L = \text{p.matriz\_adyacencia} = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

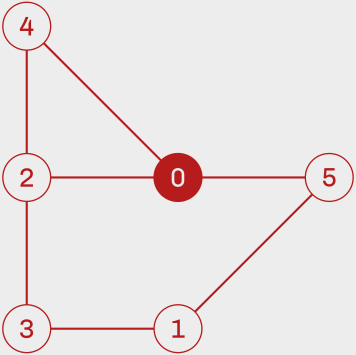
$$V = \text{incidencias} = \begin{pmatrix} 3 \\ 2 \\ 3 \\ 2 \\ 2 \\ 2 \end{pmatrix} \qquad p = \text{pos\_max} = 0$$

Bucle while

incidencias[0] = 3 > 0
T = {0}
M = sol.coste = 1



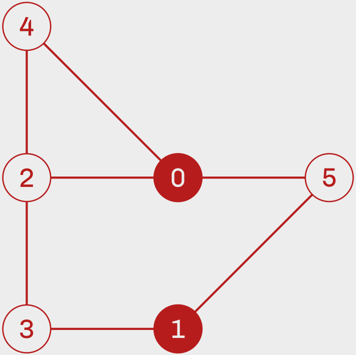
$$V = \text{incidencias} = \begin{pmatrix} 3 \\ 2 \\ 3 \\ 2 \\ 2 \\ 2 \end{pmatrix} \Longrightarrow V = \text{incidencias} = \begin{pmatrix} 0 \\ 2 \\ 2 \\ 2 \\ 1 \\ 1 \end{pmatrix}$$



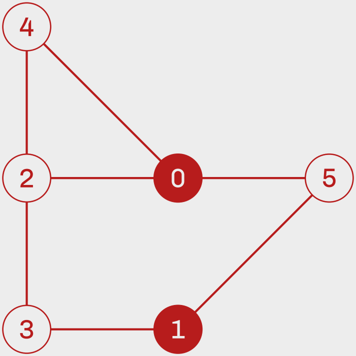
$p = pos\_max = 1$

Bucle while

incidencias[1] = 2 > 0
T = {0, 1}
M = sol.coste = 2



$$V = \text{incidencias} = \begin{pmatrix} 0 \\ 2 \\ 2 \\ 2 \\ 1 \\ 1 \end{pmatrix} \Longrightarrow V = \text{incidencias} = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

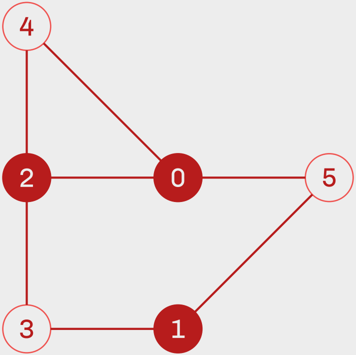


Bucle while

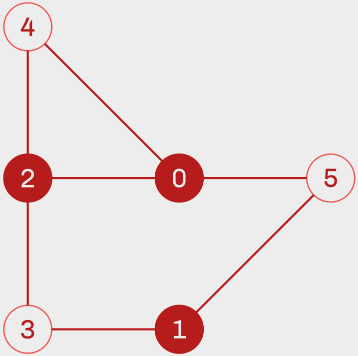
incidencias[2] = 2 > 0

T = {0, 1, 2}

M = sol.coste = 3



$$V = \text{incidencias} = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 1 \\ 1 \\ 0 \end{pmatrix} \Longrightarrow V = \text{incidencias} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

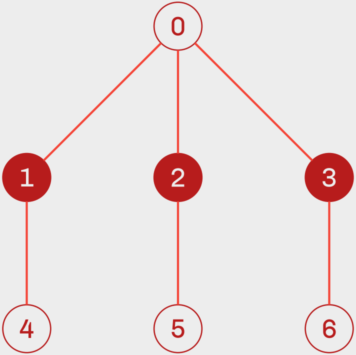


Eficiencia teórica

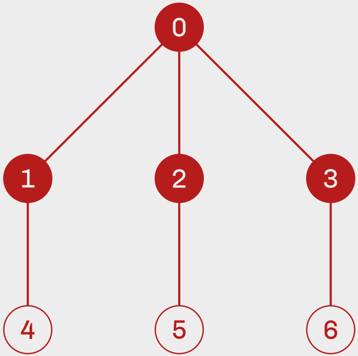
Algoritmo de orden cuadrático: $O(|V|^2)$

```
while (incidencias[pos_max] > 0) { // Mientras el vector de incidencias no sea {0,0,...,0}
  // Añadir el nodo con más incidencias a la solución (siempre es factible)
  sol.addNodo(pos_max);

  // Eliminar el nodo de la LC
  incidencias[pos_max] = 0;

  // Decrementar número de incidencias de nodos conectados con el seleccionado
  for (int j = 0; j < num_nodos; ++j) {
    if (p.estanConectados(pos_max,j) && incidencias[j] > 0)
      --incidencias[j];
  }

  // Seleccionar nodo con más incidencias
  pos_max = FSeleccion(incidencias, num_nodos);
}
```

Contraejemplo



Contraejemplo



Caso real

Pueblos y hospitales