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

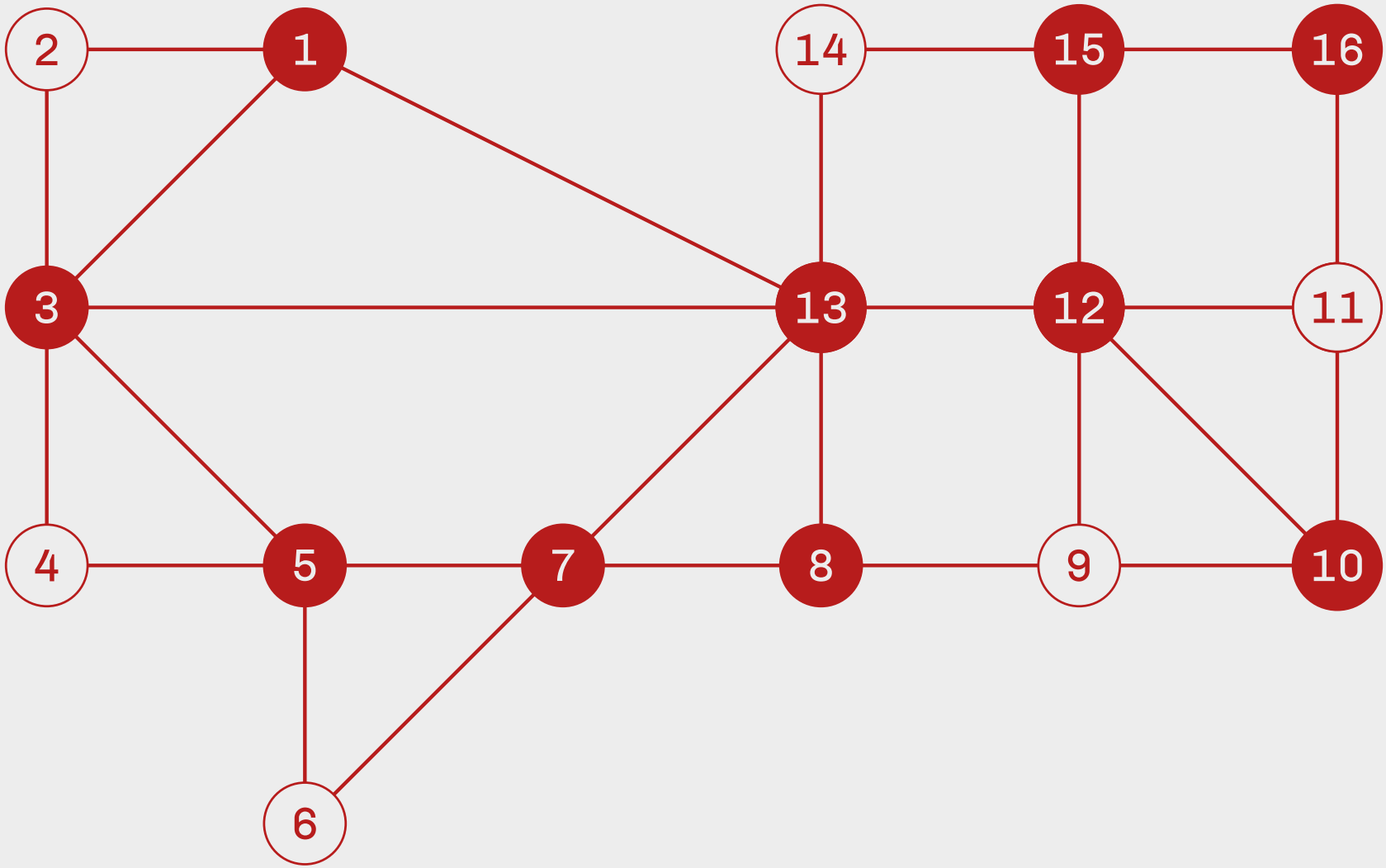
Grupo 2

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

```
1  /**
2   * Función de selección. Devuelve el nodo con mayor número
3   * de incidencias de la lista de candidatos, cuyo tamaño es N.
4   */
5  int FSeleccion(int* LC, int N) {
6    int pos_max = 0;
7    for (int i = 1; i < N; i++) {
8      if (LC[i] > LC[pos_max])
9        pos_max = i;
10   }
11   return pos_max;
12 }
```

Pseudocódigo

N ≡ número de nodos del grafo
M ≡ número de nodos del recubrimiento
L ≡ matriz de adyacencia
T[1, . . . , M] ≡ vector de nodos del recubrimiento
V[1, . . . , N] ≡ 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
```

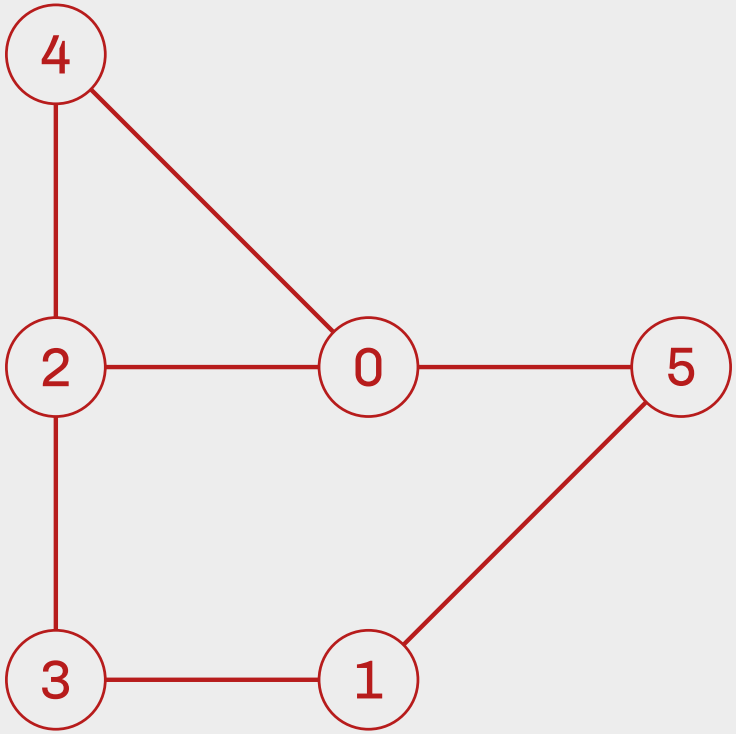
Algoritmo

```
1  Solucion RecubrimientoGrafoGreedy(const Problema& p) {
2
3    Solucion sol; // Solución a devolver
4    int num_nodos = p.getNumNodos();  // Número de candidatos sin utilizar
5    int incidencias[num_nodos];  // Vector de incidencias de cada nodo (LC
         )
6    int pos_max;
7
8    // Inicializar la lista de candidatos (LC)
9    for (int i = 0; i < num_nodos; ++i)
10     incidencias[i] = p.getNumIncidencias(i);
11
12   // Nodo con más incidencias
13   pos_max = FSeleccion(incidencias, num_nodos);
```

Algoritmo

```
1     while (incidencias[pos_max] > 0) { // Mientras el vector de
          incidencias no sea {0,0,...,0}
2       // Añadir el nodo con más incidencias a la solución (siempre es
            factible)
3       sol.addNodo(pos_max);
4
5       // Eliminar el nodo de la LC
6       incidencias[pos_max] = 0;
7
8       // Decrementar número de incidencias de nodos conectados con el
            seleccionado
9       for (int j = 0; j < num_nodos; ++j) {
10        if (p.estanConectados(pos_max,j) && incidencias[j] > 0)
11          --incidencias[j];
12      }
13
14      // Seleccionar nodo con más incidencias
15      pos_max = FSeleccion(incidencias, num_nodos);
16    }
17
18    return sol;
19  }
```

Ejemplo paso a paso



Inicialización

N = num_nodos = 6

M = sol.coste = 0

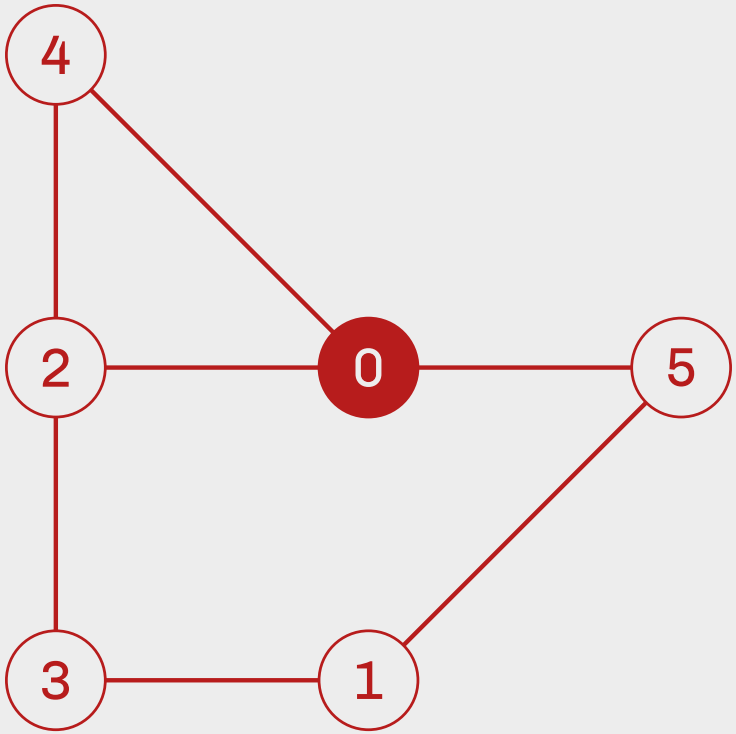
$$L = p.matriz\_adyacencia = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$V = incidencias = \begin{pmatrix} 3 \\ 2 \\ 3 \\ 2 \\ 2 \\ 2 \end{pmatrix}$$

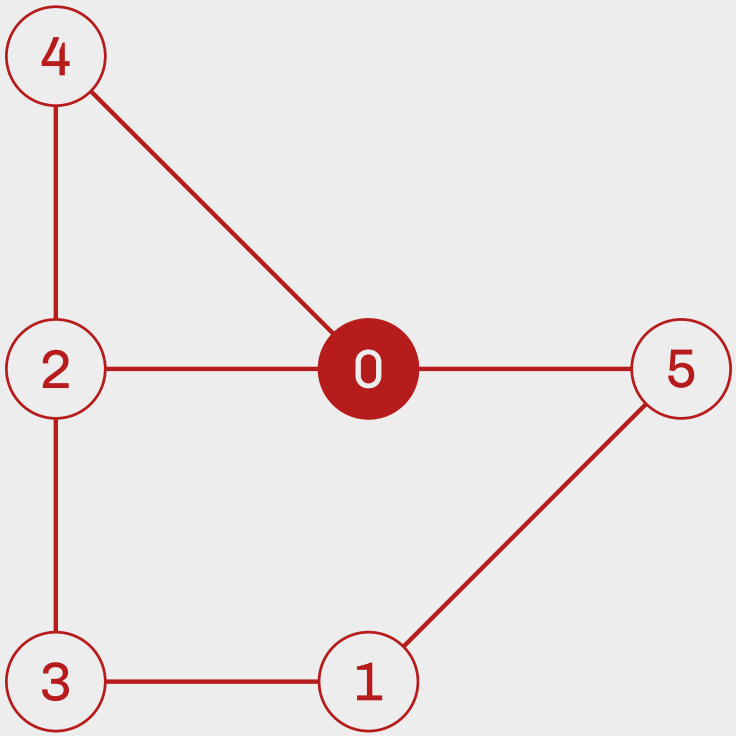
p = pos_max = 0

Bucle while

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



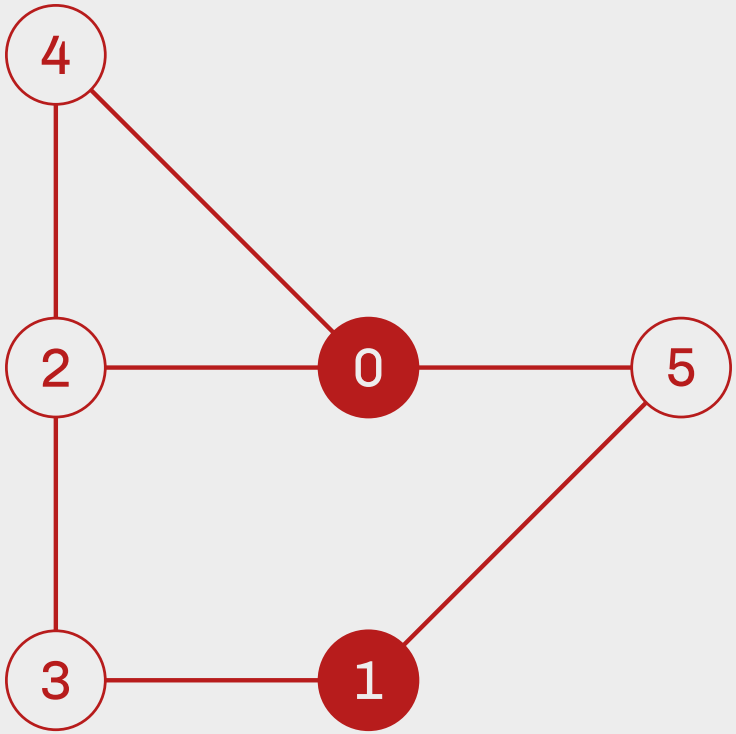
$$V = \text{incidencias} = \begin{pmatrix} 3 \\ 2 \\ 3 \\ 2 \\ 2 \\ 2 \end{pmatrix} \Longrightarrow V = \text{incidencias} = \begin{pmatrix} 0 \\ 2 \\ 2 \\ 2 \\ 1 \\ 1 \end{pmatrix}$$



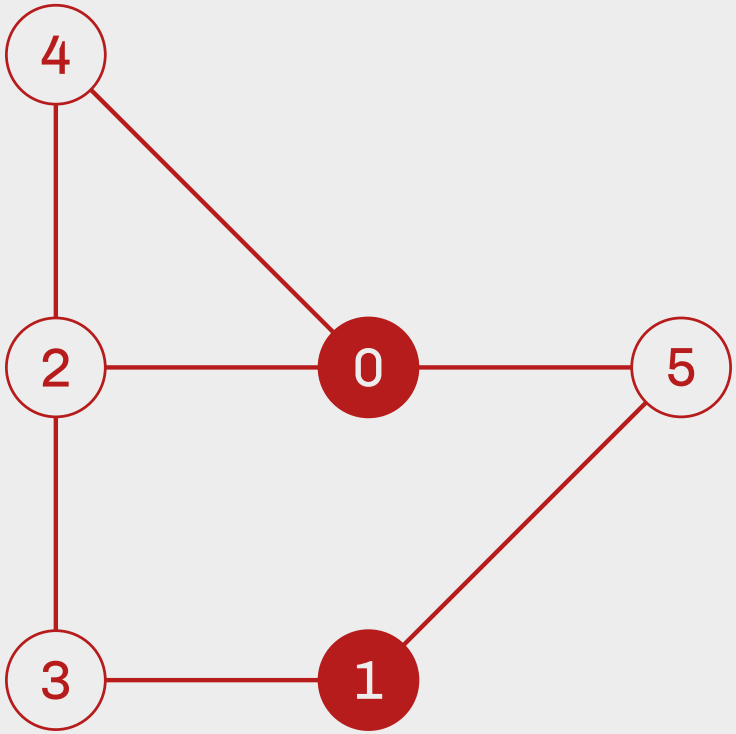
p = pos_max = 1

Bucle while

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

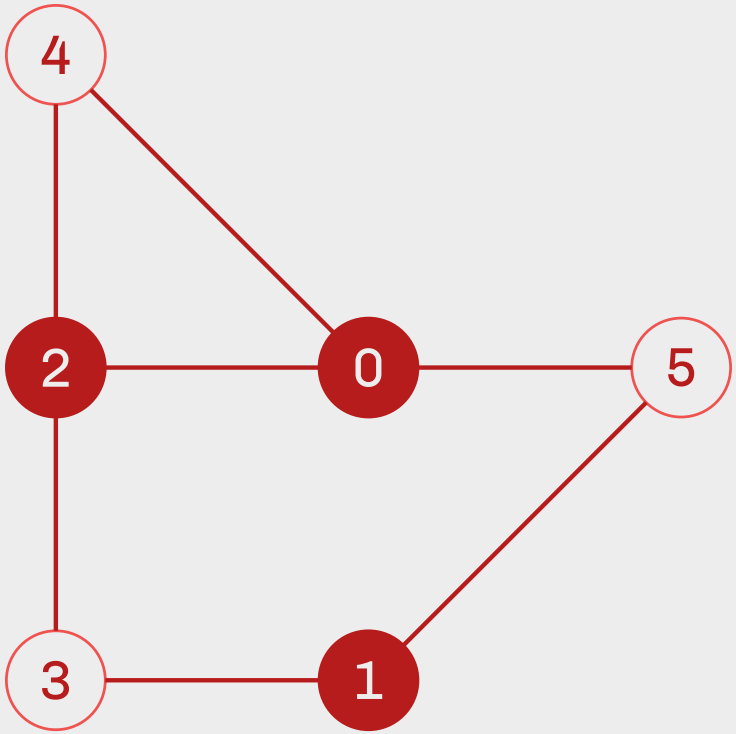


$$V = \text{incidencias} = \begin{pmatrix} 0 \\ 2 \\ 2 \\ 2 \\ 1 \\ 1 \end{pmatrix} \Longrightarrow V = \text{incidencias} = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

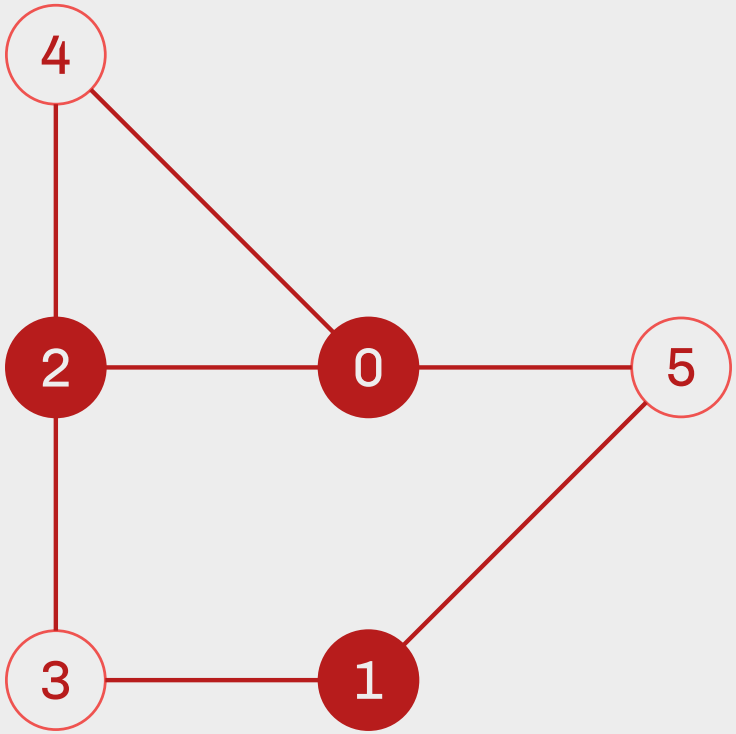


Bucle while

incidencias[2] = 2 > 0
T = {0, 1, 2}
M = sol.coste = 3



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

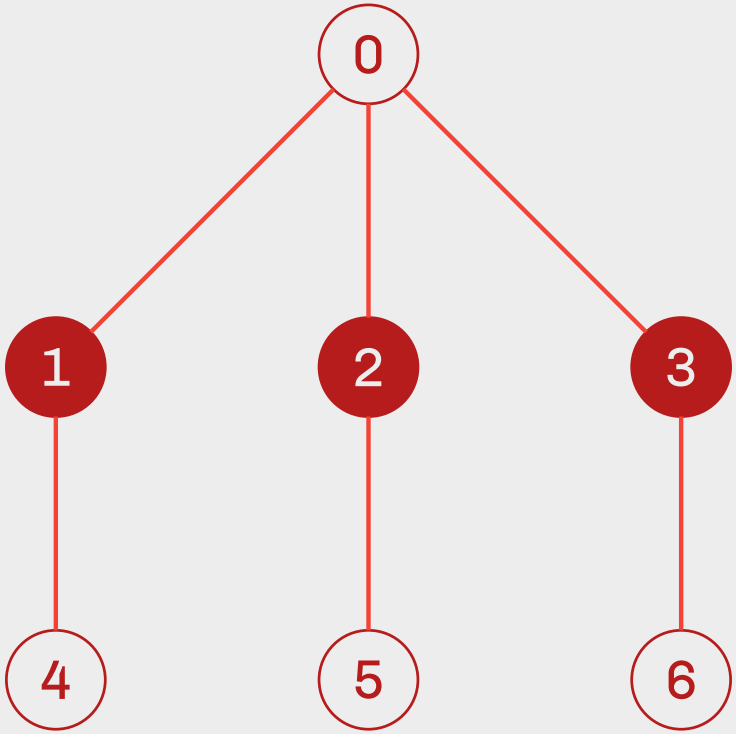


Eficiencia teórica

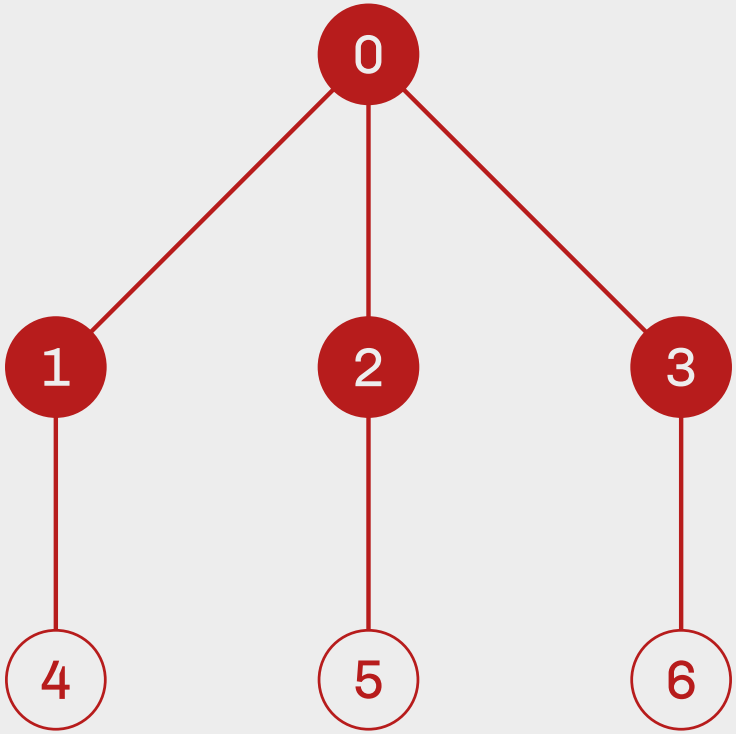
Algoritmo de orden cuadrático: $O(|V|^2)$

```
1     while (incidencias[pos_max] > 0) { // Mientras el vector de
          incidencias no sea {0,0,...,0}
2       // Añadir el nodo con más incidencias a la solución (siempre es
            factible)
3       sol.addNodo(pos_max);
4
5       // Eliminar el nodo de la LC
6       incidencias[pos_max] = 0;
7
8       // Decrementar número de incidencias de nodos conectados con el
            seleccionado
9       for (int j = 0; j < num_nodos; ++j) {
10        if (p.estanConectados(pos_max,j) && incidencias[j] > 0)
11          --incidencias[j];
12      }
13
14      // Seleccionar nodo con más incidencias
15      pos_max = FSeleccion(incidencias, num_nodos);
16    }
```

Contraejemplo



Contraejemplo



Caso real

Pueblos y hospitales