Recubrimiento de un grafo

## Índice

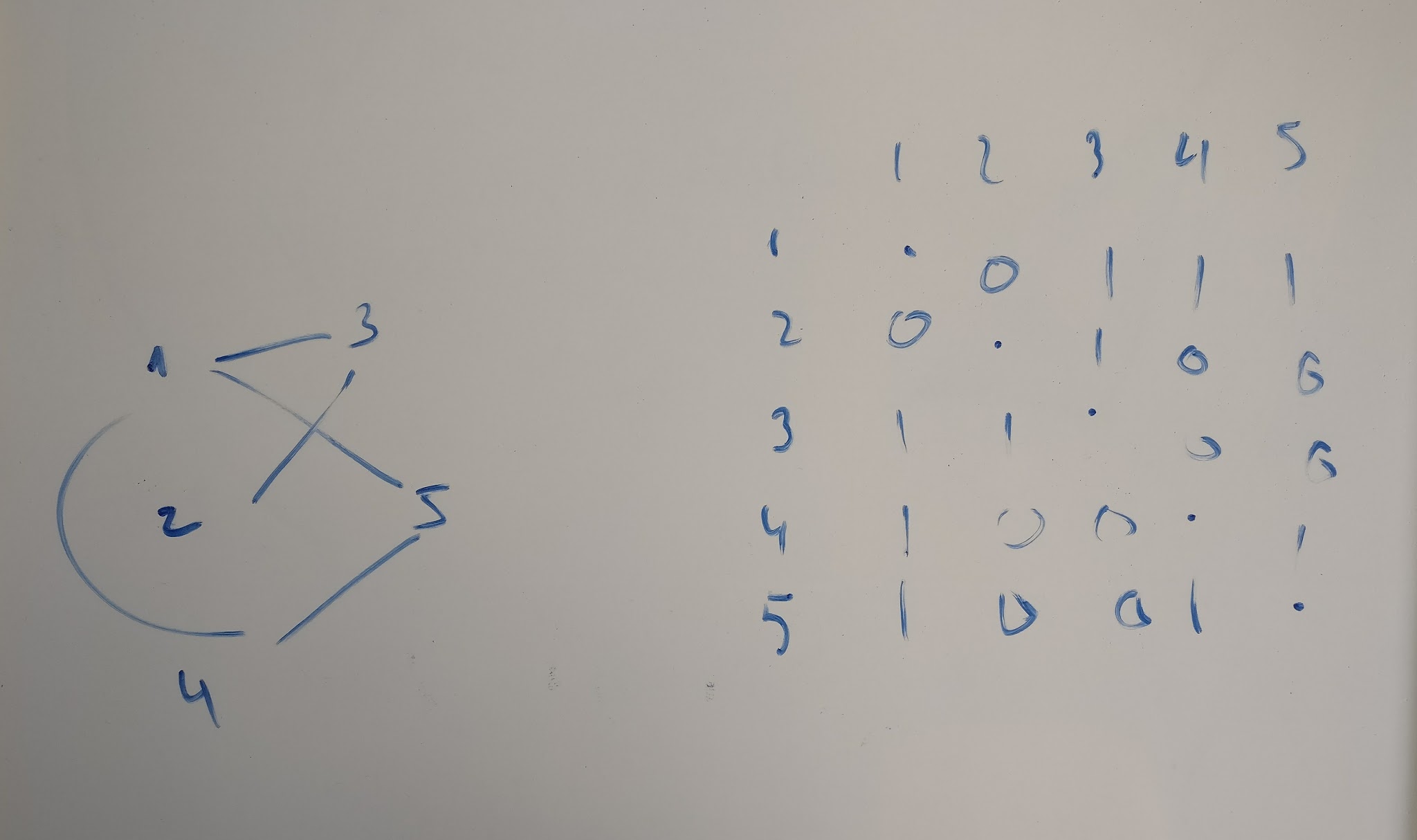
1. Análisis del problema
2. Solución
3. Eficiencia
4. Conclusiones

# Análisis del problema

En esta práctica se nos presentaba el problema del recubrimiento de un grafo. Dicho problema consta en seleccionar una serie de nodos pertenecientes a un grafo para recubrirlos y que al finalizar todas las aristas consten al menos con un nodo recubierto.

# Diseño de la solución

Para solventar este problema hemos optado por representar el grafo como una matriz triangular la cual representará qué nodos están unidos.



Para la matriz hemos usado un vector de vectores y hemos añadido el tamaño de 8 para que sea como el tablero real de ajedrez.

Para trabajar con el problema hemos creado un tipo de dato llamado “coordenada” que está compuesto por dos enteros que indican una posición de fila y columna, otro llamado “Nodo“ que representará un nodo ya sea padre o hijo generado .

# Explicación algoritmo



Lo primero que haremos en el algoritmo será crear una lista con todos los nodos del grafo y para ello recorreremos la matriz mencionada anteriormente y rellenaremos las aristas de los nodos para después añadir el nodo a la lista de candidatos.

Una vez tenemos todos los nodos los ordenaremos la lista para situar al principio aquellos nodos cuyo número de aristas sea superior.

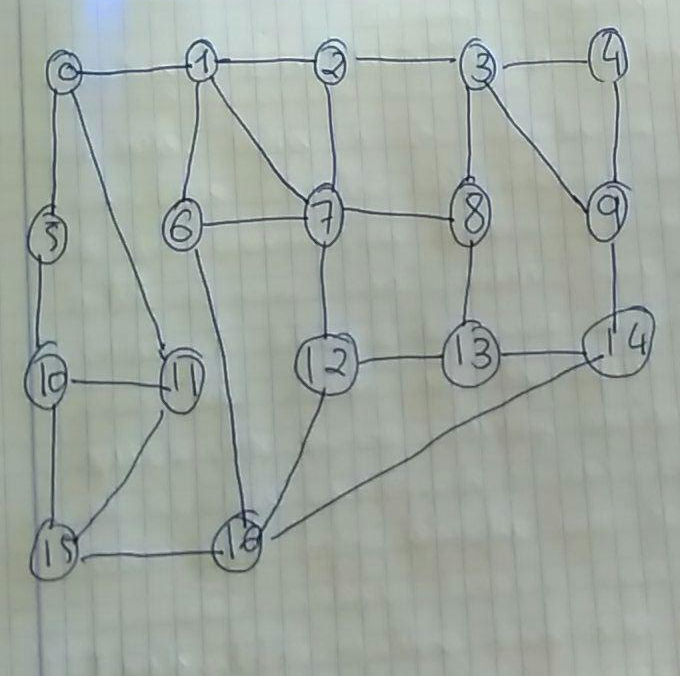
Para seleccionar los nodos que recubriremos entraremos en un bucle cuya condición de salida será superar el número de aristas totales de nuestro grafo y lo primero que haremos es comprobar para cada nodo las aristas que usa.

En un bucle comprobaremos si nuestra lista de aristas usadas contiene a una arista en particular del nodo que estamos comprobando, si no la contiene la añadiremos a nuestra lista y si ya está añadida aumentaremos el contador de aristas.

Al comprobar las aristas de un nodo comprobaremos si el contador de aristas es igual al total de aristas que tiene un nodo, y si el contador es igual al total de aristas de ese nodo quiere decir que todas las aristas ya están usadas por lo que no convendría introducir ese nodo. Es decir, si un nodo tiene alguna arista que no se haya usado formará parte de la solución.

Al finalizar con todos los nodos o sobrepasemos el total de aristas tendremos en un vector de enteros los nodos que forman la solución a nuestro problema.

# Funcionamiento algoritmo



En nuestro algoritmo rellenaremos la lista de candidatos con los nodos que vemos y al ordenarla con los nodos con mayor número de aristas en primera posición quedará la siguiente lista:

7 - 16 - 3 - 1 - 15 - 14 - 13 - 12 - 11 - 10 - 9 - 8 - 6 - 2 - 0 - 5 - 4

5 aristas - 7

4 aristas - 16, 3 y 1

3 aristas - 15, 14, 13, 12, 11, 10, 9, 8, 6, 2 y 0

2 aristas - 4 y 5

Al comprobar si las aristas del 7 están en nuestra lista de aristas usadas vemos que no hay ninguna por lo que entrarán todas:

Aristas Usadas:

7 - 1 7 - 8

7 - 2 7 - 12

7 - 6

Añadimos el nodo a la solución: 7

Comprobamos ahora para el siguiente nodo (16).

Ninguna de sus aristas está usada por lo que aristas usadas será:

7 - 1 7 - 8 16 - 12

7 - 2 7 - 12 16 - 14

7 - 6 16 - 6 16 - 15

Nodos solución: 7 - 16

**Siguiente nodo: 3**

Ninguna de sus aristas está usada. Aristas usadas:

7 - 1 7 - 8 16 - 12

7 - 2 7 - 12 16 - 14

7 - 6 16 - 6 16 - 15

3 - 2 3 - 4 3 - 8

3 - 9

Se añade a la solución.

Nodos solución: 7 - 16 - 3

**Siguiente nodo: 1**

La arista 7 - 1 está usada, por lo que se aumenta su contador de aristas usadas: 1

Las otras aristas se añaden:

7 - 1 7 - 8 16 - 12

7 - 2 7 - 12 16 - 14

7 - 6 16 - 6 16 - 15

3 - 2 3 - 4 3 - 8

3 - 9 1 - 0 1 - 2

1 - 6

Se añade a la solución dado que no todas sus aristas están usadas.

Nodos solución: 7 - 16 - 3 - 1

**Siguiente nodo: 15**

La arista 15 - 16 está usada, su contador se incrementa a: 1

Se añaden las otras aristas:

7 - 1 7 - 8 16 - 12

7 - 2 7 - 12 16 - 14

7 - 6 16 - 6 16 - 15

3 - 2 3 - 4 3 - 8

3 - 9 1 - 0 1 - 2

1 - 6 15 - 10 15 - 11

Se añade a la solución dado que no todas sus aristas están usadas-

Nodos solución: 7 - 16 - 3 - 1 - 15

**Siguiente nodo: 14**

Tiene la arista 16 - 14 usada por lo que su contador se queda en: 1

Aristas usadas:

7 - 1 7 - 8 16 - 12

7 - 2 7 - 12 16 - 14

7 - 6 16 - 6 16 - 15

3 - 2 3 - 4 3 - 8

3 - 9 1 - 0 1 - 2

1 - 6 15 - 10 15 - 11

14 - 9 14 - 13

Se añade a la solución dado que no todas sus aristas están utilizadas.

Nodos solución: 7 - 16 - 3 - 1 - 15 - 14

**Siguiente nodo: 13**

La arista 14 - 13 ya está usada por lo que su contador aumenta a: 1

Se añaden las otras aristas:

7 - 1 7 - 8 16 - 12

7 - 2 7 - 12 16 - 14

7 - 6 16 - 6 16 - 15

3 - 2 3 - 4 3 - 8

3 - 9 1 - 0 1 - 2

1 - 6 15 - 10 15 - 11

14 - 9 14 - 13 13 - 12

13 - 8

Se añade a la solución dado que no todas sus aristas están utilizadas.

Nodos solución: 7 - 16 - 3 - 1 - 15 - 14 - 13

**Siguiente nodo: 12**

Las aristas usadas son 12 - 16, 12 - 7 y 12 - 3 que es igual al número total de aristas (3 aristas) por lo que este nodo no se añade a la solución y las aristas usadas permanecen igual.

Nodos solución: 7 - 16 - 3 - 1 - 15 - 14 - 13

**Siguiente nodo: 11**

La arista 15 -11 ya está usada y su contador aumenta a: 1

Se añaden las otras aristas:

7 - 1 7 - 8 16 - 12

7 - 2 7 - 12 16 - 14

7 - 6 16 - 6 16 - 15

3 - 2 3 - 4 3 - 8

3 - 9 1 - 0 1 - 2

1 - 6 15 - 10 15 - 11

14 - 9 14 - 13 13 - 12

13 - 8 11 - 10 11 - 0

Se añade a la solución dado que no todas sus aristas están utilizadas.

Nodos solución: 7 - 16 - 3 - 1 - 15 - 14 - 13 - 11

**Siguiente nodo: 10**

Las aristas usadas son 11 - 10 y 15 - 10 por lo que su contador aumenta a: 2

Se añade la arista que no está usada:

7 - 1 7 - 8 16 - 12

7 - 2 7 - 12 16 - 14

7 - 6 16 - 6 16 - 15

3 - 2 3 - 4 3 - 8

3 - 9 1 - 0 1 - 2

1 - 6 15 - 10 15 - 11

14 - 9 14 - 13 13 - 12

13 - 8 11 - 10 11 - 0

10 - 5

Se añade a la solución dado que no todas sus aristas están utilizadas.

Nodos solución: 7 - 16 - 3 - 1 - 15 - 14 - 13 - 11 - 10

**Siguiente nodo: 9**

Las aristas usadas son 14 - 9 y 3 - 9, su contador aumenta a: 2

Se añade la arista que no está usada:

7 - 1 7 - 8 16 - 12

7 - 2 7 - 12 16 - 14

7 - 6 16 - 6 16 - 15

3 - 2 3 - 4 3 - 8

3 - 9 1 - 0 1 - 2

1 - 6 15 - 10 15 - 11

14 - 9 14 - 13 13 - 12

13 - 8 11 - 10 11 - 0

10 - 5 9 - 4

Se añade a la solución dado que no todas sus aristas están utilizadas.

Nodos solución: 7 - 16 - 3 - 1 - 15 - 14 - 13 - 11 - 10 - 9

**Siguiente nodo: 8**

Todas sus aristas ya están usadas por lo que no se añade a la solución y la lista de aristas usadas permanece igual.

Nodos solución: 7 - 16 - 3 - 1 - 15 - 14 - 13 - 11 - 10 - 9

**Siguiente nodo: 6**

Todas sus aristas ya están usadas por lo que no se añade a la solución y la lista de aristas usadas permanece igual.

Nodos solución: 7 - 16 - 3 - 1 - 15 - 14 - 13 - 11 - 10 - 9

**Siguiente nodo: 2**

Todas sus aristas ya están usadas por lo que no se añade a la solución y la lista de aristas usadas permanece igual.

Nodos solución: 7 - 16 - 3 - 1 - 15 - 14 - 13 - 11 - 10 - 9

**Siguiente nodo: 0**

Las aristas 1 - 0 y 11 - 0 están usadas por lo que su contador aumenta a: 2

Se añade la arista que no está usada:

7 - 1 7 - 8 16 - 12

7 - 2 7 - 12 16 - 14

7 - 6 16 - 6 16 - 15

3 - 2 3 - 4 3 - 8

3 - 9 1 - 0 1 - 2

1 - 6 15 - 10 15 - 11

14 - 9 14 - 13 13 - 12

13 - 8 11 - 10 11 - 0

10 - 5 9 - 4 0 - 5

Se añade a la solución dado que no todas sus aristas están utilizadas.

Nodos solución: 7 - 16 - 3 - 1 - 15 - 14 - 13 - 11 - 10 - 9 - 0

**Siguiente nodo: 5**

Todas sus aristas ya están usadas por lo que no se añade a la solución y la lista de aristas usadas permanece igual.

Nodos solución: 7 - 16 - 3 - 1 - 15 - 14 - 13 - 11 - 10 - 9 - 0

**Siguiente nodo: 4**

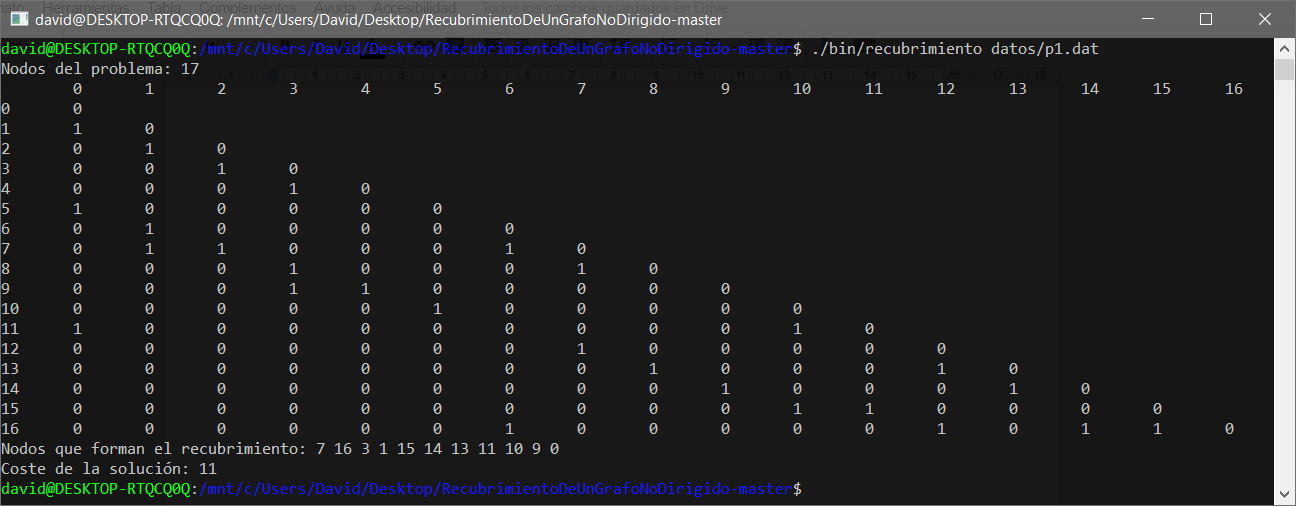
Todas sus aristas ya están usadas por lo que no se añade a la solución y la lista de aristas usadas permanece igual.

Nodos solución: 7 - 16 - 3 - 1 - 15 - 14 - 13 - 11 - 10 - 9 - 0

---------------------------------------------------------------------------------------------------------------------------

Acaba el algoritmo y obtenemos el siguiente vector con los nodos solución:

**7 - 16 - 3 - 1 - 15 - 14 - 13 - 11 - 10 - 9 - 0**

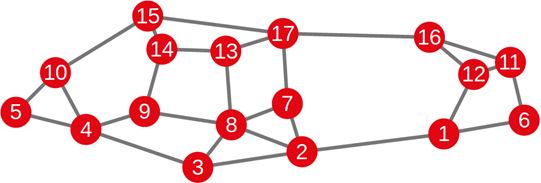


# Caso real

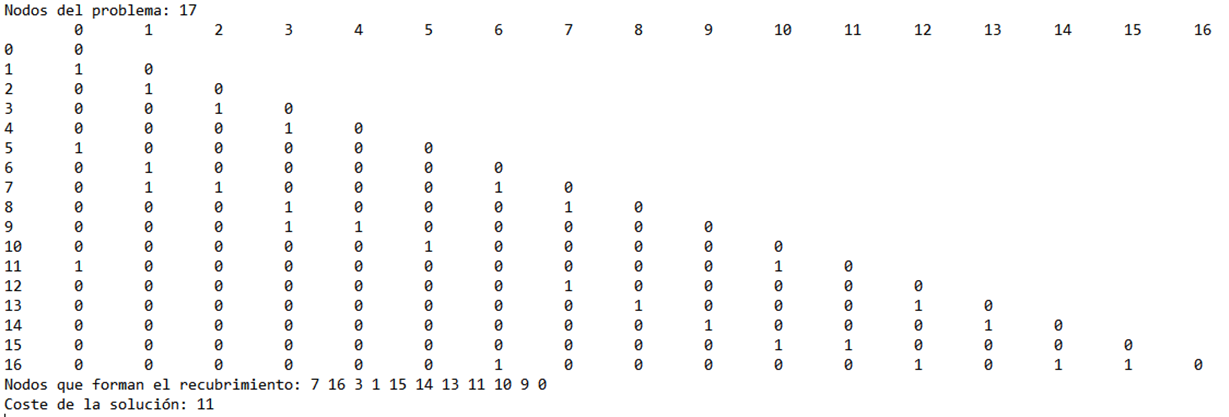
Tenemos una ciudad con calles y bajo sus calles están las tuberías que hacen llegar el agua a los vecinos de los edificios colindantes en esa calle.

Necesitamos cavar el menor número de pozos para poder surtir a todos los vecinos del pueblo de agua.

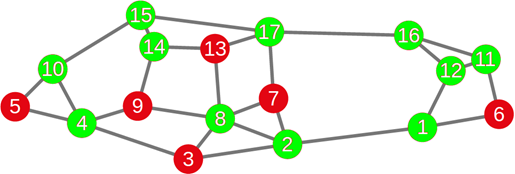
Lo gris son las calles de la ciudad y lo rojo el máximo de pozos a cavar.



Aplicamos nuestro algoritmo para ver el mínimo de pozos:

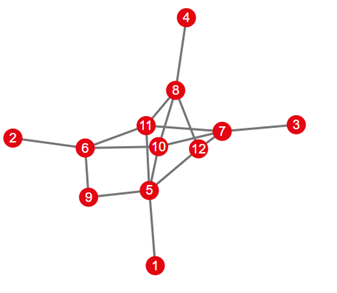


La solución que nos aporta es la siguiente

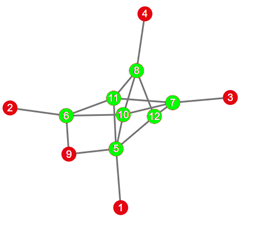


Como vemos es una buena solución y a priori no se nos ocurre ninguna más óptima, parece que nuestro algoritmo funciona como se espera.

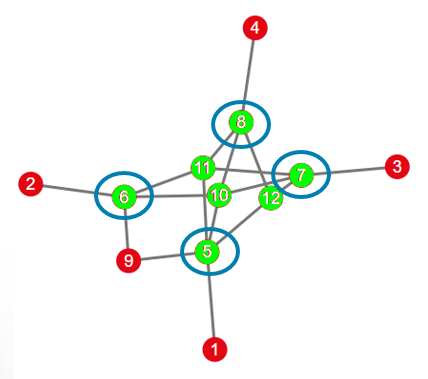
Sin embargo hemos encontrado un grafo para el cual nuestro algoritmo no consigue el óptimo, es el siguiente:



Si introducimos la matriz de adyacencia del grafo y aplicamos nuestro algoritmo, obtenemos la siguiente solución:



Pero podemos comprobar sin mucho esfuerzo que no es la más óptima ya que encontramos una solución donde todos los edificios de la ciudad estarían abastecidos de agua si cavamos únicamente 4 pozos. Lo vemos en la siguiente imagen: (la solución óptima esta rodeada en azul)



# Eficiencia teórica

# //OPERACIÓN

# **while(aristas\_usadas.size() < num\_aristas){**

# n = candidatos.front();//seleccionamos el primer candidato, puesto que la lista está ordenada el primero siempre será el más prometedor

# **for (int i = 0; i < n.aristas.size(); ++i){**

# **if (!EstaArista(aristas\_usadas, n.aristas[i]))**

# aristas\_usadas.push\_back(n.aristas[i]);

# else n\_aristas++;//aristas que ya están usadas en la estructura "aristas\_usadas", del nodo que estoy analizando en el bucle.

# }

# if (!(n\_aristas == n.aristas.size()))//!(si todas las aristas del nodo que estoy analizando (n) están usadas, no meto ese nodo).

# sol.aniadirNodo(n.nodo);

# candidatos.pop\_front();//borramos el candidato analizado.

# n\_aristas = 0;

# }

# Como podemos ver en el código las líneas en azul estas líneas valen 1 y las rojas en el peor de los casos valen n.

Las líneas por fuera del for valen 5. Dentro del for la comprobación de la condición del if siempre se hace la cual implica recorrer un vector que en el peor de los casos es n igual que el número de iteraciones del for, de manera que al estar el if dentro del for el for en total valdrá n2 y el while vale n, asi que T(n)=2n3 + 5 que en el peor de los casos es O(n3)

# Como el tamaño de las aristas suele ser mediano o pequeño se tiene que el tiempo promedio del algoritmo se puede comportar como un O(n2).

# Instrucciones compilación y ejecución

Acceder a la carpeta y ejecutar el makefile.

Una vez hecho esto ejecutar el programa llamado recubrimiento alojado en /bin y pasar como parámetro la matriz situada en /datos.

**Ej:** *$ make*

*$ ./bin/recubrimiento datos/p1.dat*