

# Lab08: Árbol de expansión mínimo y minisat

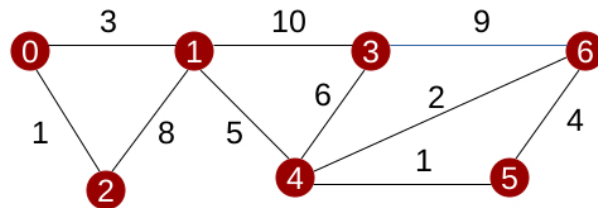
## 1. Encontrar un árbol de expansión mínimo

El algoritmo de Prim permite hallar el árbol minimal de cualquier grafo con pesos, no dirigido y conexo. Hay que seguir los siguientes pasos:

- Marcamos un nodo cualquiera, será el nodo de partida.
- Seleccionamos la arista de menor peso incidente en el nodo marcado anteriormente, y marcamos el otro nodo en el que incide.
- Repetimos el paso anterior siempre que la arista elegida enlace un nodo marcado y otro que no lo esté.
- El proceso termina cuando tenemos todos los nodos del grafo marcados.

### Ejemplo

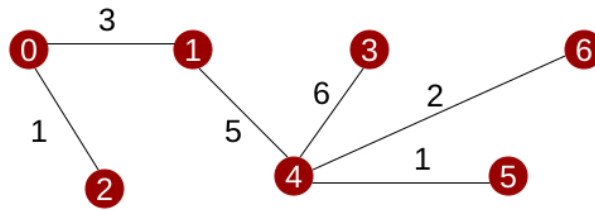
Vamos a construir el árbol de expansión mínimo para el siguiente grafo:



Siguiendo el algoritmo de Prim:

- Elegimos, por ejemplo, el nodo 0 y lo marcamos.
- Elegimos la arista con menor peso incidente en 0, la  $(0, 2)$ . Marcamos el otro nodo en el que incide, el 2.
- Elegimos la arista con menor peso incidente en un nodo marcado y otro que no lo esté, la  $(0, 1)$ . Marcamos el nodo no marcado, el 1.
- Elegimos la arista con menor peso incidente en un nodo marcado y otro que no lo esté, la  $(1, 4)$ . Marcamos el nodo no marcado, el 4.
- Elegimos la arista con menor peso incidente en un nodo marcado y otro que no lo esté, la  $(4, 5)$ . Marcamos el nodo no marcado, el 5.
- Elegimos la arista con menor peso incidente en un nodo marcado y otro que no lo esté, la  $(4, 6)$ . Marcamos el nodo no marcado, el 6.
- Elegimos la arista con menor peso incidente en un nodo marcado y otro que no lo esté, la  $(4, 3)$ . Marcamos el nodo no marcado, el 3.
- Finalizamos dado que tenemos marcados los 7 nodos del grafo.

Por tanto el árbol de expansión mínimo resultante sería:



### Para realizar el ejercicio:

Dispones del fichero `lab08_arbol.py`, donde tienes que implementar la función `prim_algorithm`. Dicha función recibe un grafo no dirigido conexas y con pesos, que son números reales, y devuelve un árbol de expansión mínimo del grafo.

## 2. MiniSAT

La carpeta `instances` en la carpeta `Code for Students` contiene un juego de pruebas extraídas de (<https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html>).

Cada instancia está en un fichero de texto en formato DIMACS (estándar para la mayoría de los SAT-solvers). Este formato es similar al nuestro pero requiere una cabecera inicial con la palabra “`p cnf`” y el número de variable y cláusulas de la fórmula. Además cada cláusula se especifica en una única línea que siempre termina con un 0.

Vamos a ver cómo funciona **minisat**: un SAT-solver pequeño y fácil de usar (<http://minisat.se/>).

La primera fórmula del fichero `instancias`, que se llama `1-unsat.cnf`, es insatisfactible. Teclea en un terminal:

```
minisat 1-unsat.cnf
```

En caso de que la fórmula sea satisfactible, puedes obtener una asignación a las variables que la haga cierta. Si escribes

```
minisat 2-sat.cnf asig-2.txt,
```

se creará un fichero `asig-2.txt` donde se encuentra la asignación.

Prueba todas las instancias del fichero `instances` para familiarizarte con el SAT-solver `minisat`.