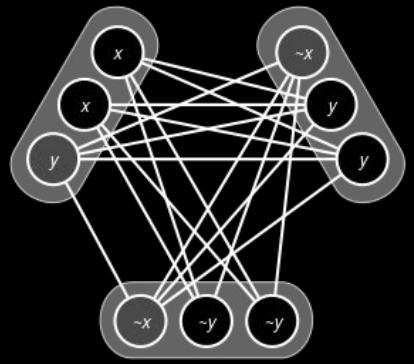
Vertex Cover

Grupo 6

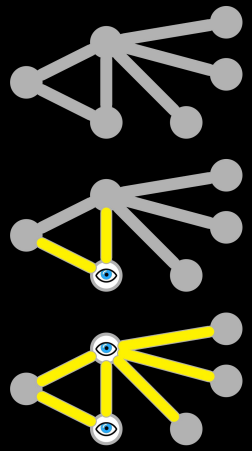
Aram Pérez Dios
Pablo Pérez González
Roberto Carrazana Pernía

Problemas involucrados

3SAT



VERTEX COVER



Transformación de 3SAT a VC

3SAT

$C = \{c_1, c_2, \dots, c_m\}$ $\quad |c_i| = 3$

$U = \{u_1, u_2, \dots, u_n\}$

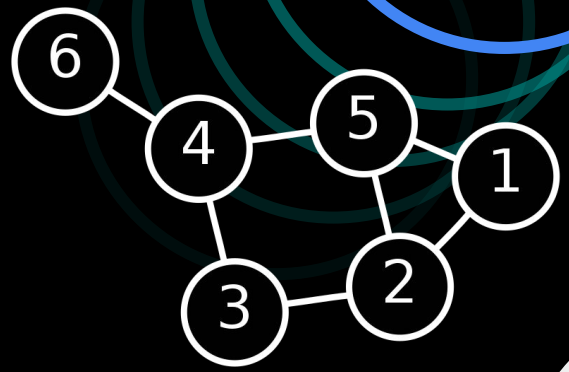
$x_1 \vee x_2 \vee x_6$

$\neg x_1 \vee x_4 \vee x_3$

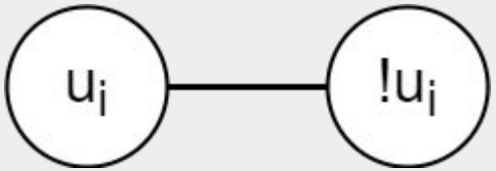
$\neg x_3 \vee x_1 \vee x_7$

Vertex Cover

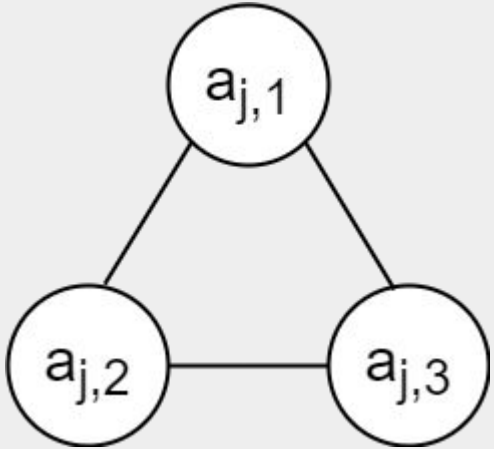
$G(V, E)$



Transformación de 3SAT a VC



True Setting Components



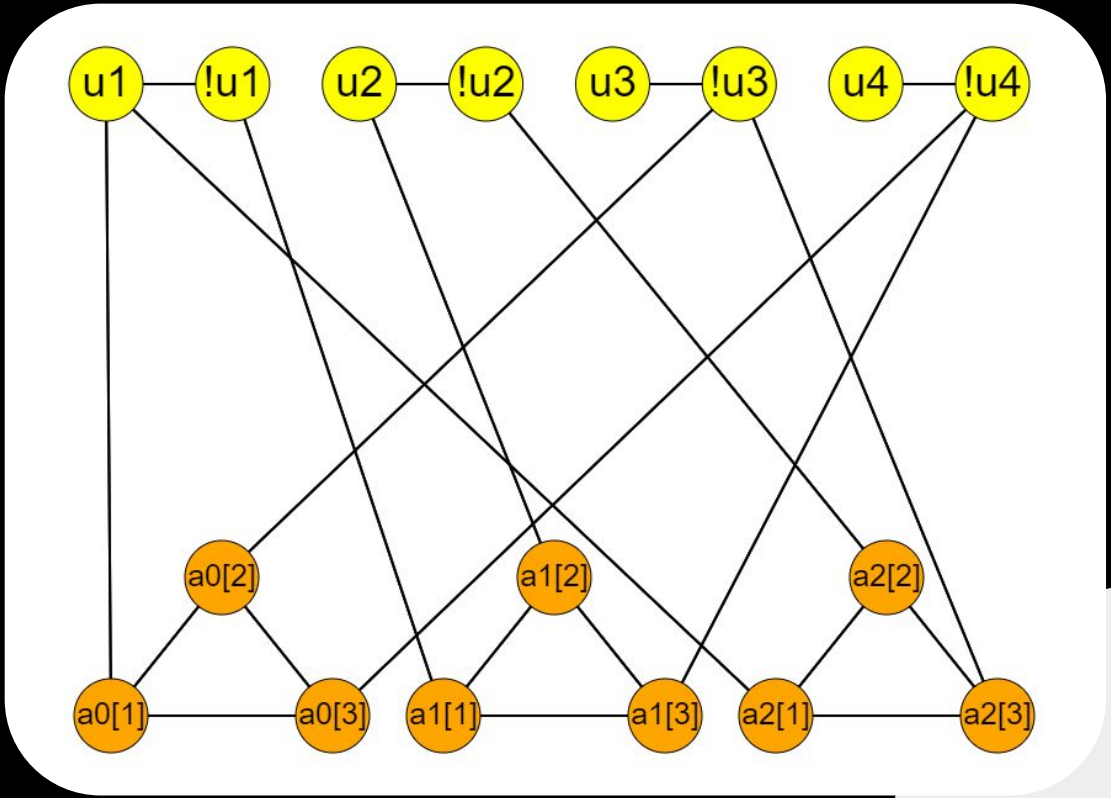
Satisfaction Testing Component

Transformación de 3SAT a VC

$u_1 \vee \neg u_3 \vee \neg u_4$

$\neg u_1 \vee u_2 \vee \neg u_4$

$u_1 \vee \neg u_2 \vee \neg u_3$



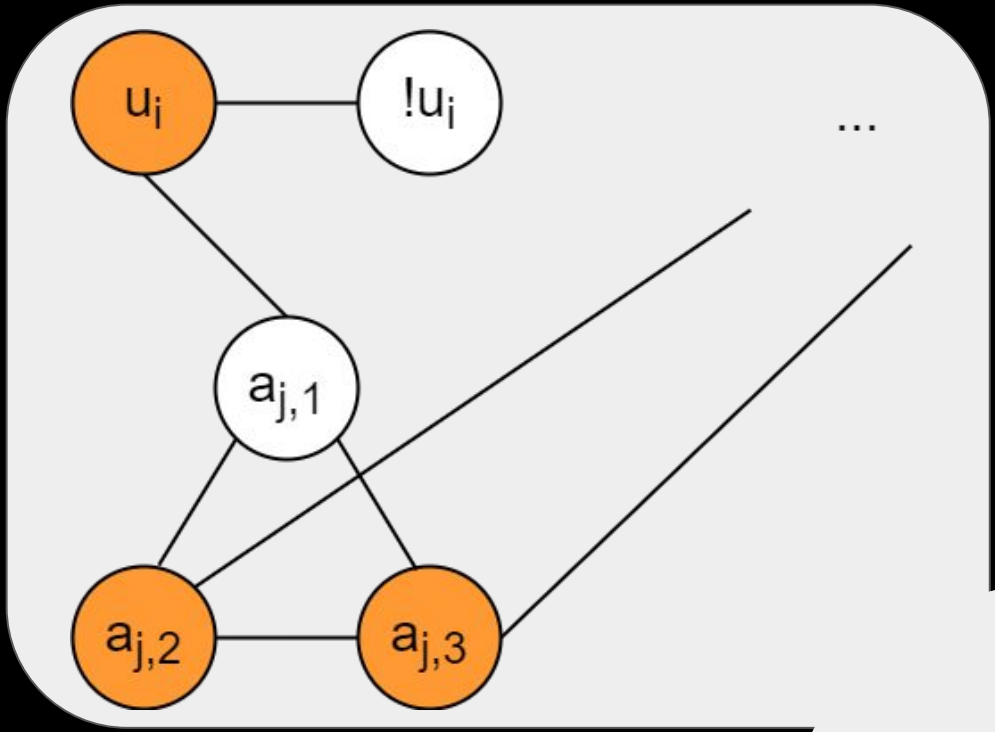
Satisfactibilidad tras la transformación

$$K = n + 2m$$

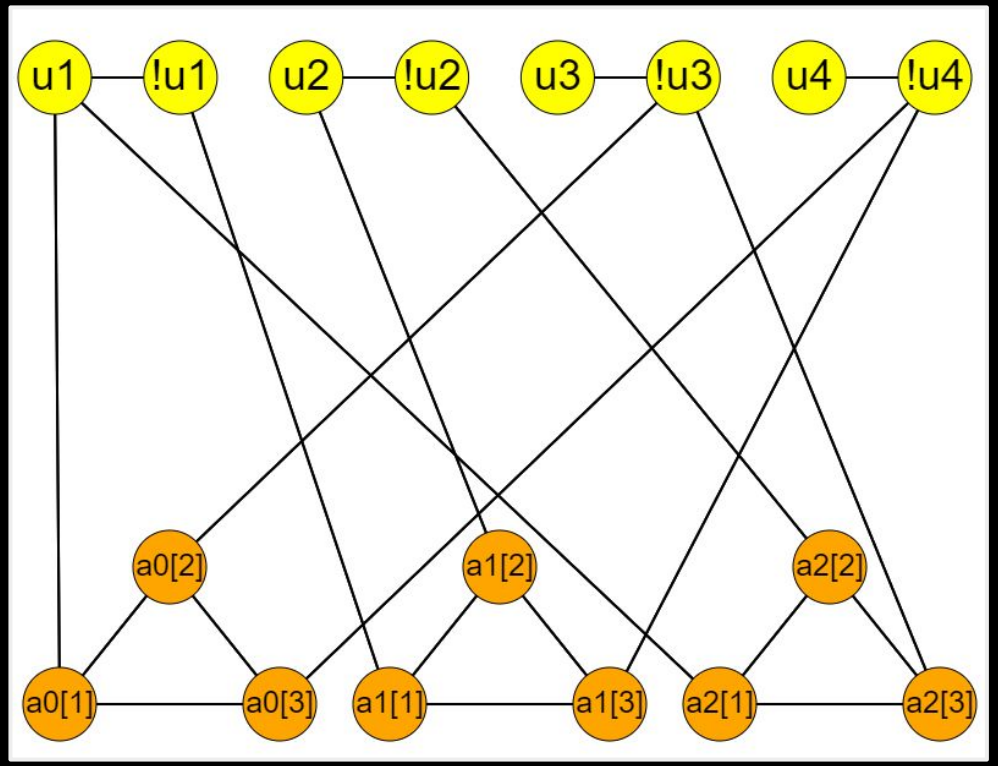
$$V' \leq K \leq V$$

$$m = |C|$$

$$n = |U|$$

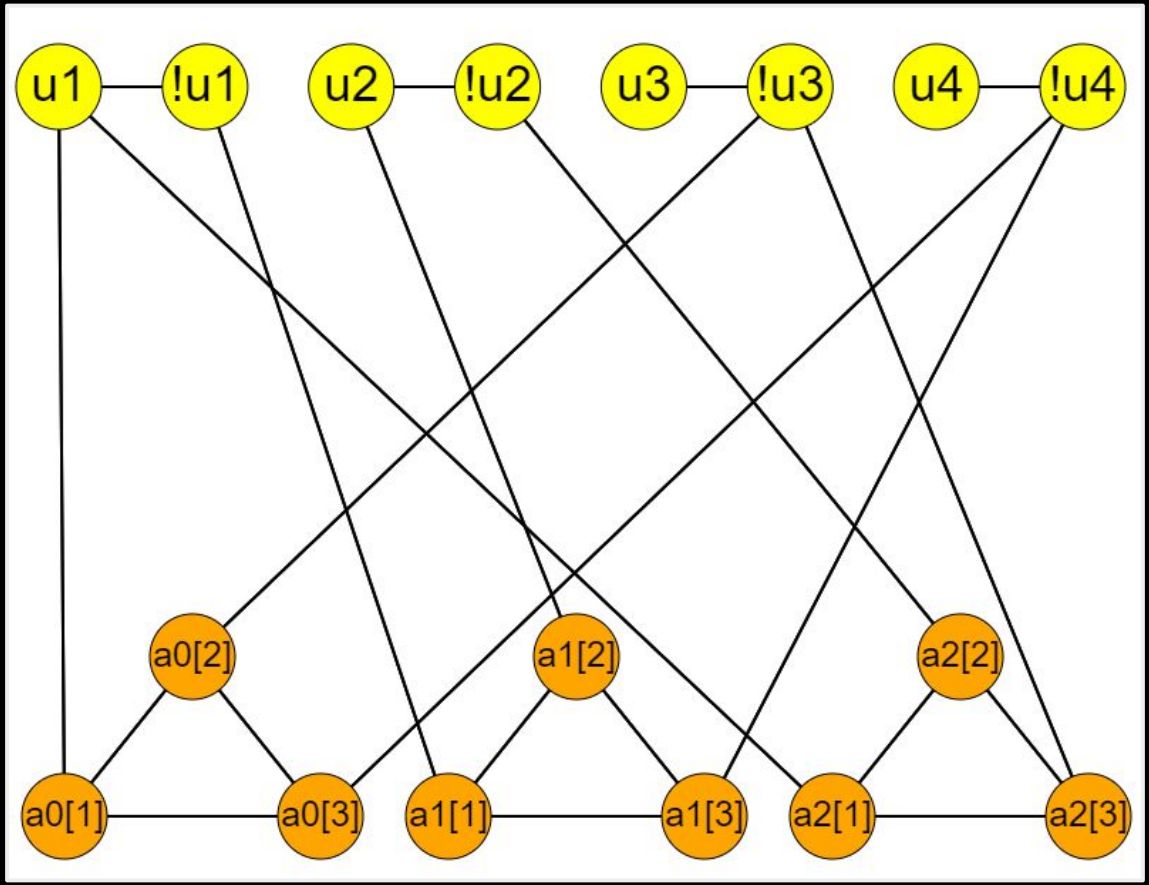


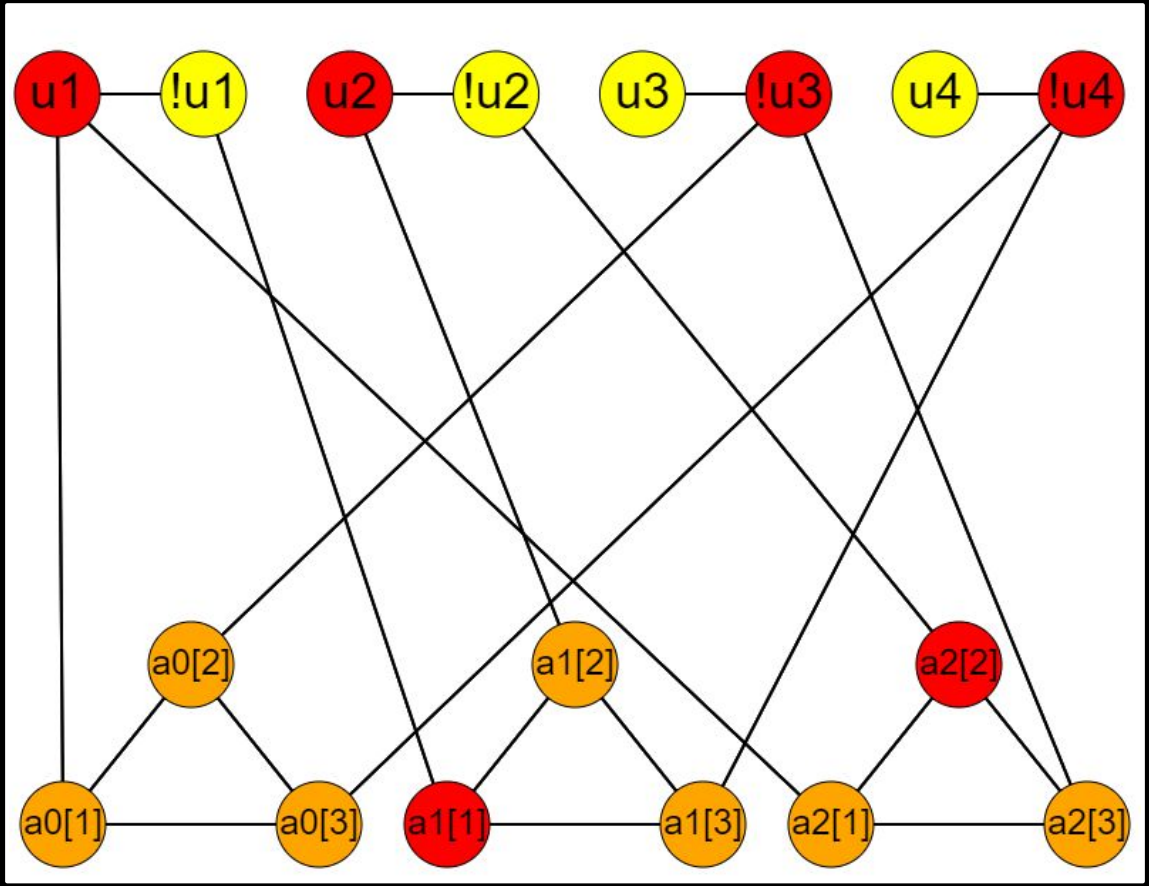
Ejemplo

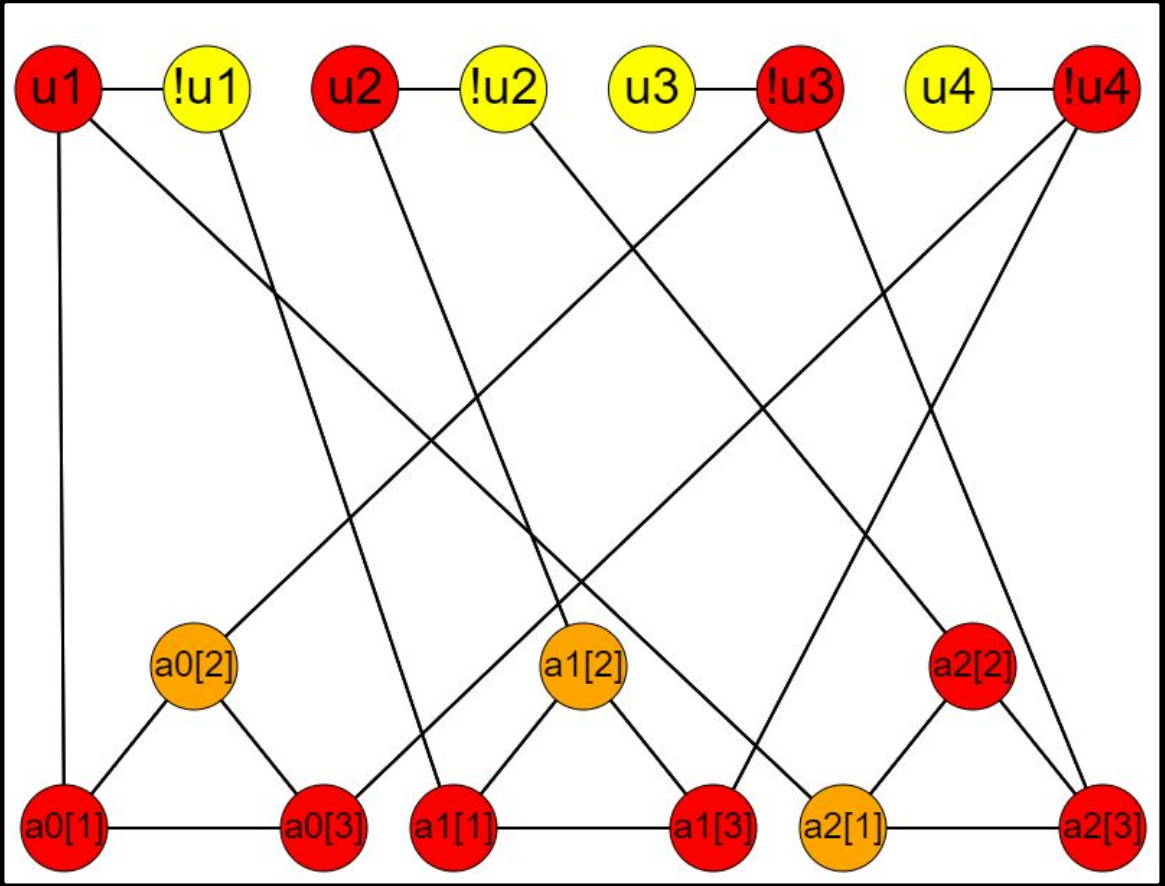


Para conseguir un VC, serán necesarios dos pasos:

1. Incluir uno de cada par de literales.
2. Incluir dos nodos de cada gadget.
3. $|V'| \leq K$


u1
!u1
u2
!u2
u3
!u3
u4
!u4
a0[2]
a1[2]
a2[2]
a0[1]
a0[3]
a1[1]
a1[3]
a2[1]
a2[3]


u1
!u1
u2
!u2
u3
!u3
u4
!u4
a0[2]
a1[2]
a2[2]
a0[1]
a0[3]
a1[1]
a1[3]
a2[1]
a2[3]



V’ = { u1, u2, !u3, !u4, a0[1], a0[3], a1[1], a1[3], a2[2], a2[3] }

|V’|= 10

Implementación

- JavaScript
- Clase ThreeSAT. Lectura de ficheros.
- Clase VertexCover que almacena el grafo.
- Representación por consola o HTML Canvas.

Entrada (JSON)

```json
{
    "literalsCount":3,
    "literals": ["u1","u2","u3"],
    "clausesCount": 3,
    "clauses": ["u1 !u2 !u3", "!u1 u2 !u3", "u1 !u2 u3"]
}
```

Clase ThreeSAT

```
/** @desc Clase ThreeSAT */
export class ThreeSAT {

  /** @desc Constructor de la clase */
  constructor() {
    this.literals = [];
    this.clauses = [];
  }

  /**
   * @desc Método para crear una entrada de problema 3SAT desde un objeto
   * @param {Object} threeSATData - objeto 3SAT leido desde documento JSON
   */
  createFromObject(threeSATData) {···
  }
```

```
/** @desc Clase Clause */
export class Clause {

  /** @desc Constructor de la clase */
  constructor(literals = []) {
    this.literals = literals;
  }
}
```

Clase VertexCover

```javascript
/** @desc Clase VertexCover */
export class VertexCover {

  /**
   * @desc Constructor de la clase VertexCover
   * @param {ThreeSAT} threeSAT - entrada de un problema ThreeSAT
   */
  constructor(threeSAT) {
    this.threeSAT = threeSAT;
    this.graph = new Graph();
    this.#createLiterals();
    this.#createClauses();
  }

  /** @desc Método para crear los literales incluidos en el VertexCover */
  #createLiterals() {
    for (const literal of this.threeSAT.literals) {
      this.graph.addVertex(literal);
      this.graph.addVertex('!' + literal);
      this.graph.addEdge(literal, '!' + literal);
    }
  }

  /** @desc Método para crear las cláusulas */
  #createClauses() {
    let clauseNumber = 0;
    for (const clause of this.threeSAT.clauses) {
      this.graph.addVertex(`a${clauseNumber}[1]`);
      this.graph.addVertex(`a${clauseNumber}[2]`);
      this.graph.addVertex(`a${clauseNumber}[3]`);

      this.graph.addEdge(`a${clauseNumber}[1]`, `a${clauseNumber}[2]`);
      this.graph.addEdge(`a${clauseNumber}[1]`, `a${clauseNumber}[3]`);
      this.graph.addEdge(`a${clauseNumber}[2]`, `a${clauseNumber}[3]`);

      // Cada a[i][j] se conecta al literal correspondiente de la clausula
      this.graph.addEdge(`a${clauseNumber}[1]`, clause.literals[0]);
      this.graph.addEdge(`a${clauseNumber}[2]`, clause.literals[1]);
      this.graph.addEdge(`a${clauseNumber}[3]`, clause.literals[2]);

      clauseNumber++;
    }
  }
}
```

Clase Graph

```javascript
/** @desc Clase Graph */
export class Graph {

  /**
   * @desc Constructor de la clase Graph
   * @param {Number} numberOfVertices - cantidad de nodos
   */
  constructor(numberOfVertices = 0){
    this.numberOfVertices = numberOfVertices;
    this.adjacentList = new Map();
  }

  /**
   * @desc Método para añadir un nuevo nodo
   * @param {String} newVertex - etiqueta del nuevo vértice
   */
  addVertex(newVertex){
    this.adjacentList.set(newVertex, []);
  }

  /**
   * @desc Método para añadir una nueva arista
   * @param {String} vertex - etiqueta del vértice que se quiere conectar
   * @param {String} newVertex - etiqueta del vértice al que se conecta
   */
  addEdge(vertex, newVertex){
    this.adjacentList.get(vertex).push(newVertex);
    this.adjacentList.get(newVertex).push(vertex);
  }
```

Fin