Universidad Nacional Mayor de San Marcos
Facultad de Ingeniería de Sistemas e Informática

ALGORÍTMICA 3

BÚSQUEDA DE PUENTES Y CICLOS EN UN GRAFO DE TAMAÑO "N"

GRUPO 5

- ❑ Bayes Enriquez, Humberto Valentín [3]
- ❑ Marquina Carihuasari, Alfonso Francisco [3]
- ❑ Molina Soto, Lesli Lisbeth [3]
- ❑ Muñoz Capcha, Alex Cristhian [3]
- ❑ Parejas Fundar, Jorge Rubén [3]
- ❑ Ramirez Martinez, Aldo Raúl [3]

Motivación

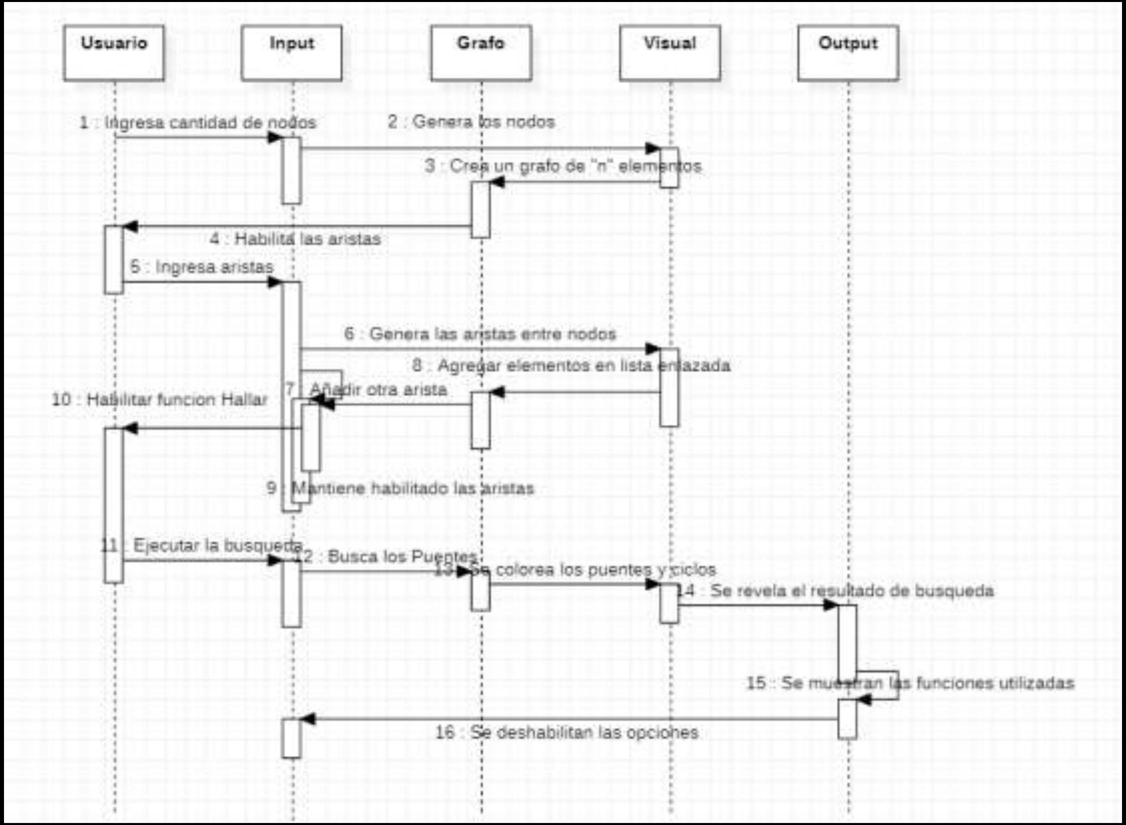
Comprender y mostrar el desempeño del algoritmo de búsqueda de puentes y ciclos en un grafo de tamaño n.

Así entender su funcionamiento y posterior aplicación en distintos casos que se requiera la ejecución de dicho algoritmo

Problema

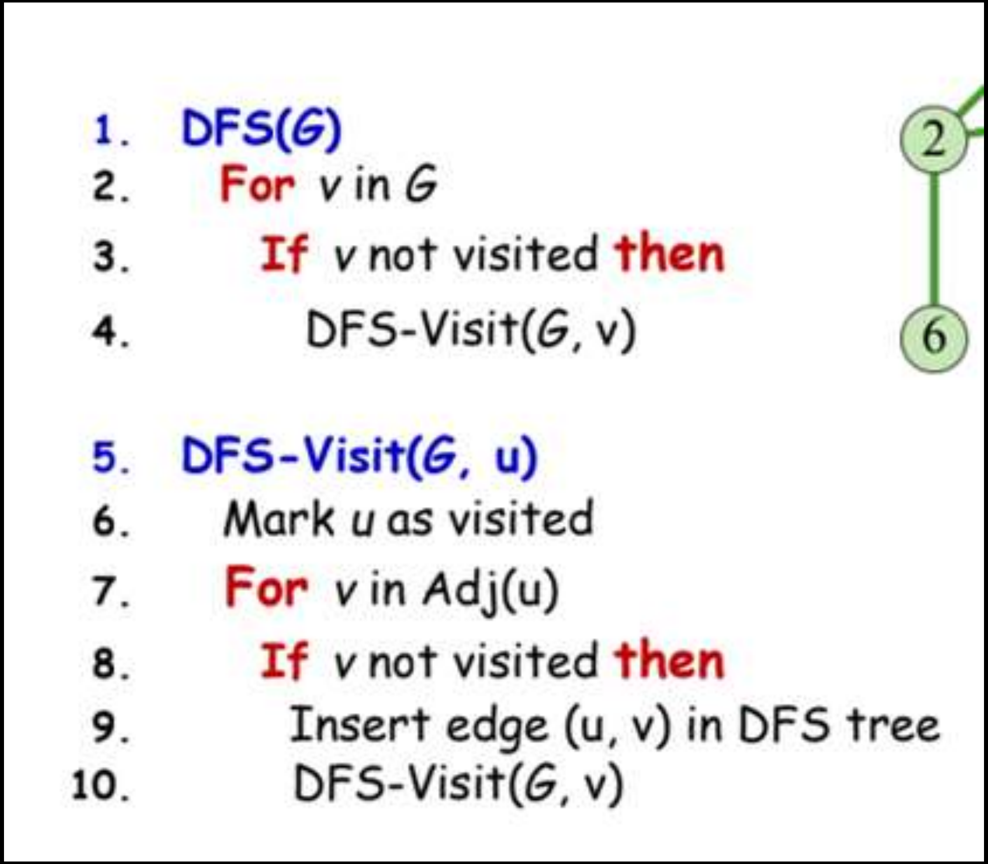
Para un grafo con n vértices y m aristas calcular y mostrar los puentes y ciclos. La entrada se compondrá de: El número de vértices n seguido del número de aristas m; que será ingresado como el inicio y final de cada arista.

Diagrama

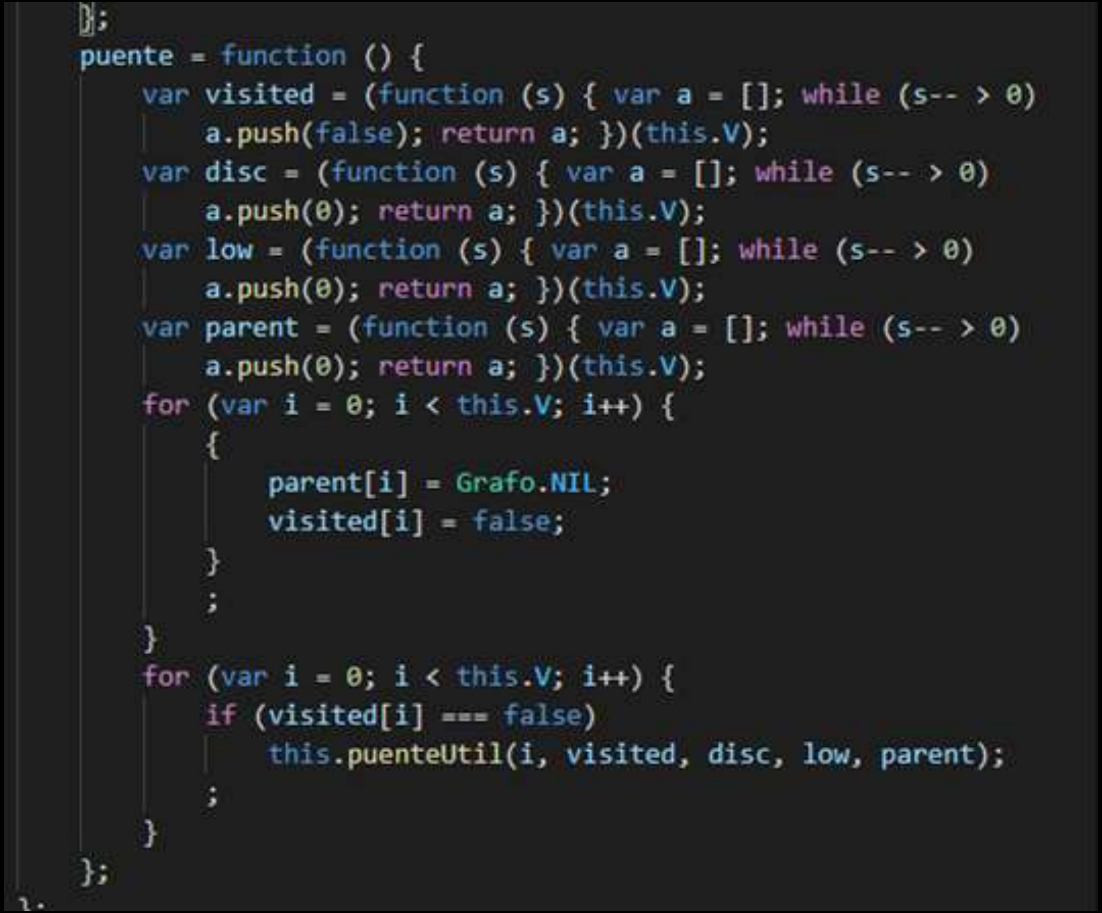

Usuario
Input
Grafo
Visual
Output
1 : Ingresa cantidad de nodos
2 : Genera los nodos
3 : Crea un grafo de "n" elementos
4 : Habilita las aristas
5 : Ingresa aristas
6 : Genera las aristas entre nodos
8 : Agregar elementos en lista enlazada
7 : Añadir otra arista
10 : Habilitar funcion Hallar
9 : Mantiene habilitado las aristas
11 : Ejecutar la busqueda
12 : Busca los Puentes
13 : Se colorea los puentes y ciclos
14 : Se revela el resultado de busqueda
15 : Se muestran las funciones utilizadas
16 : Se deshabilitan las opciones

Tecnicas de Programacion

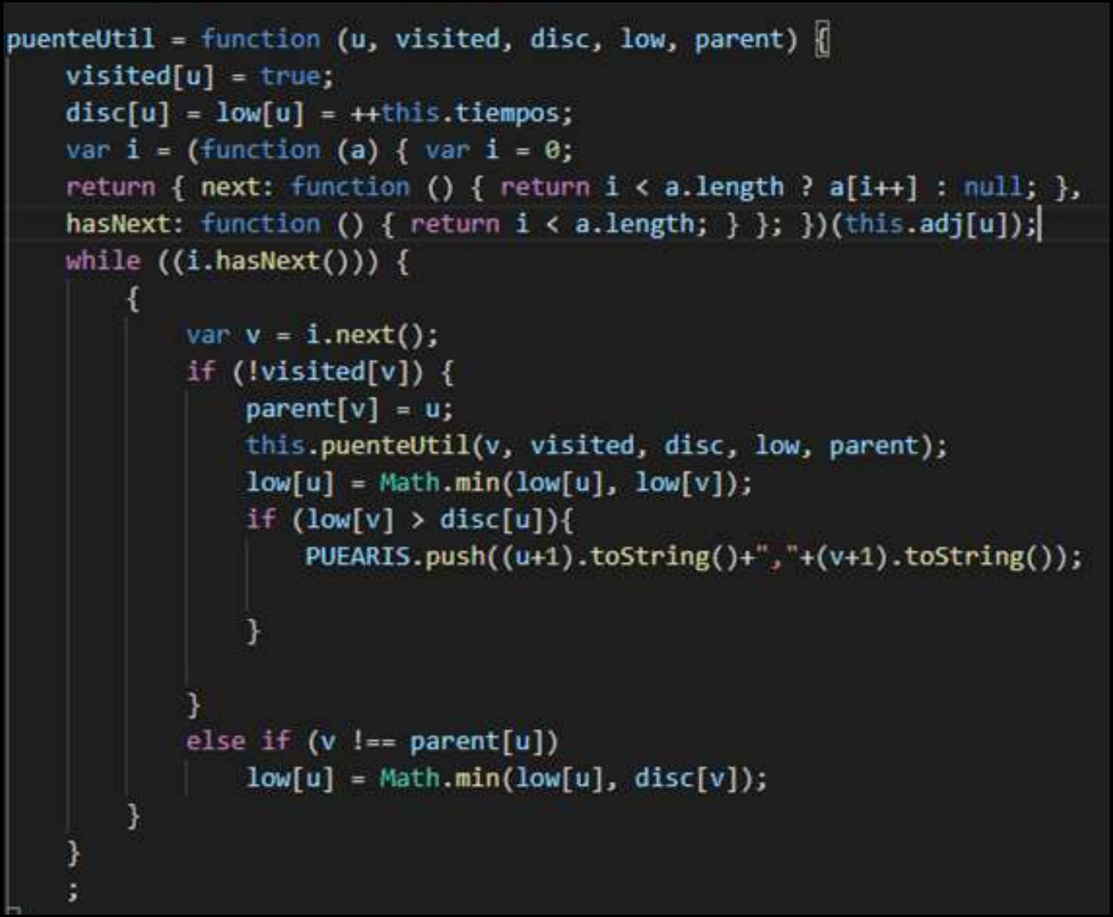
DFS :



```
1.  DFS(G)
2.    For v in G
3.      If v not visited then
4.        DFS-Visit(G, v)

5.  DFS-Visit(G, u)
6.    Mark u as visited
7.    For v in Adj(u)
8.      If v not visited then
9.        Insert edge (u, v) in DFS tree
10.       DFS-Visit(G, v)
```



```
    };
    puente = function () {
        var visited = (function (s) { var a = []; while (s-- > 0)
            a.push(false); return a; })(this.V);
        var disc = (function (s) { var a = []; while (s-- > 0)
            a.push(0); return a; })(this.V);
        var low = (function (s) { var a = []; while (s-- > 0)
            a.push(0); return a; })(this.V);
        var parent = (function (s) { var a = []; while (s-- > 0)
            a.push(0); return a; })(this.V);
        for (var i = 0; i < this.V; i++) {
            {
                parent[i] = Grafo.NIL;
                visited[i] = false;
            }
            ;
        }
        for (var i = 0; i < this.V; i++) {
            if (visited[i] === false)
                this.puenteUtil(i, visited, disc, low, parent);
            ;
        }
    };
```

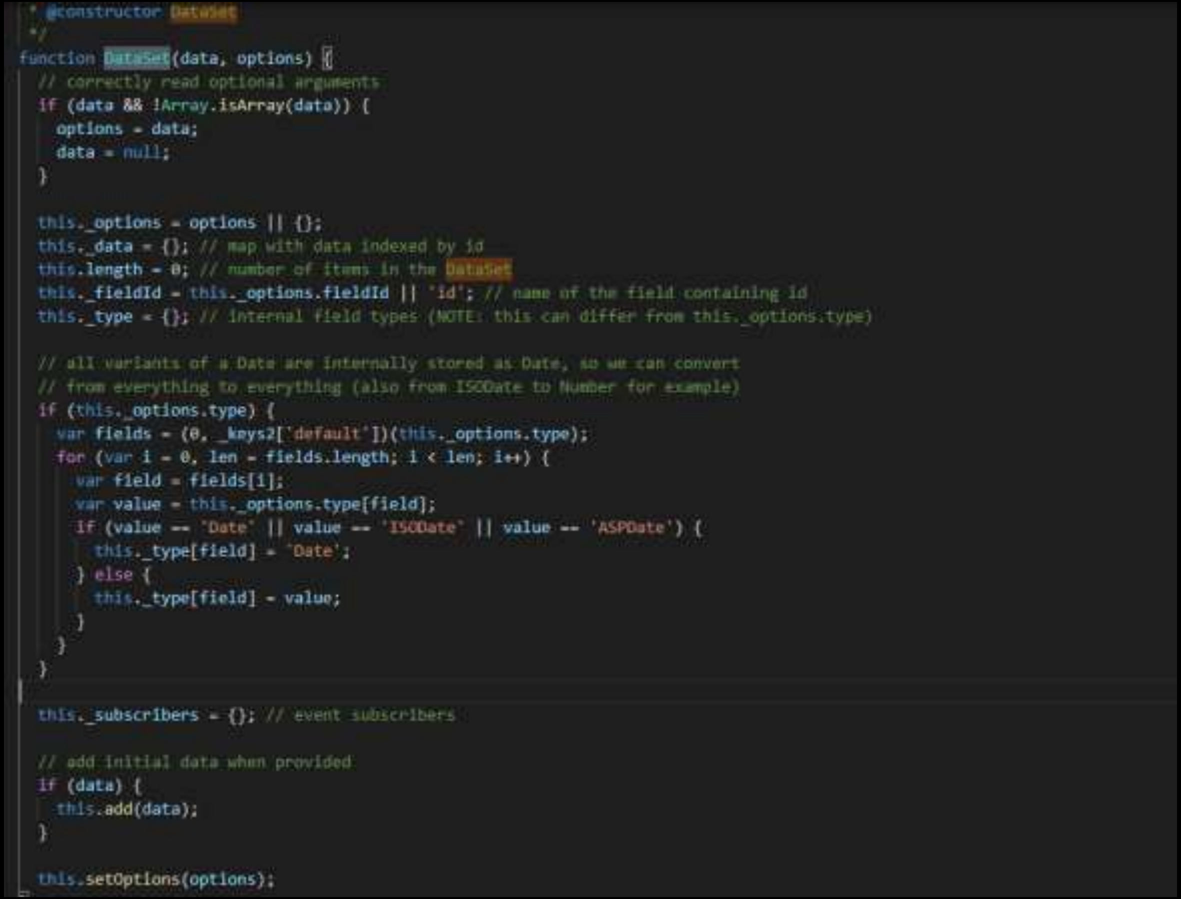


```
puenteUtil = function (u, visited, disc, low, parent) {
    visited[u] = true;
    disc[u] = low[u] = ++this.tiempos;
    var i = (function (a) { var i = 0;
    return { next: function () { return i < a.length ? a[i++] : null; },
    hasNext: function () { return i < a.length; } }; })(this.adj[u]);
    while ((i.hasNext())) {
        {
            var v = i.next();
            if (!visited[v]) {
                parent[v] = u;
                this.puenteUtil(v, visited, disc, low, parent);
                low[u] = Math.min(low[u], low[v]);
                if (low[v] > disc[u]){
                    PUEARIS.push((u+1).toString()+","+(v+1).toString());

                }

            }
            else if (v !== parent[u])
                low[u] = Math.min(low[u], disc[v]);
        }
    }
    ;
```

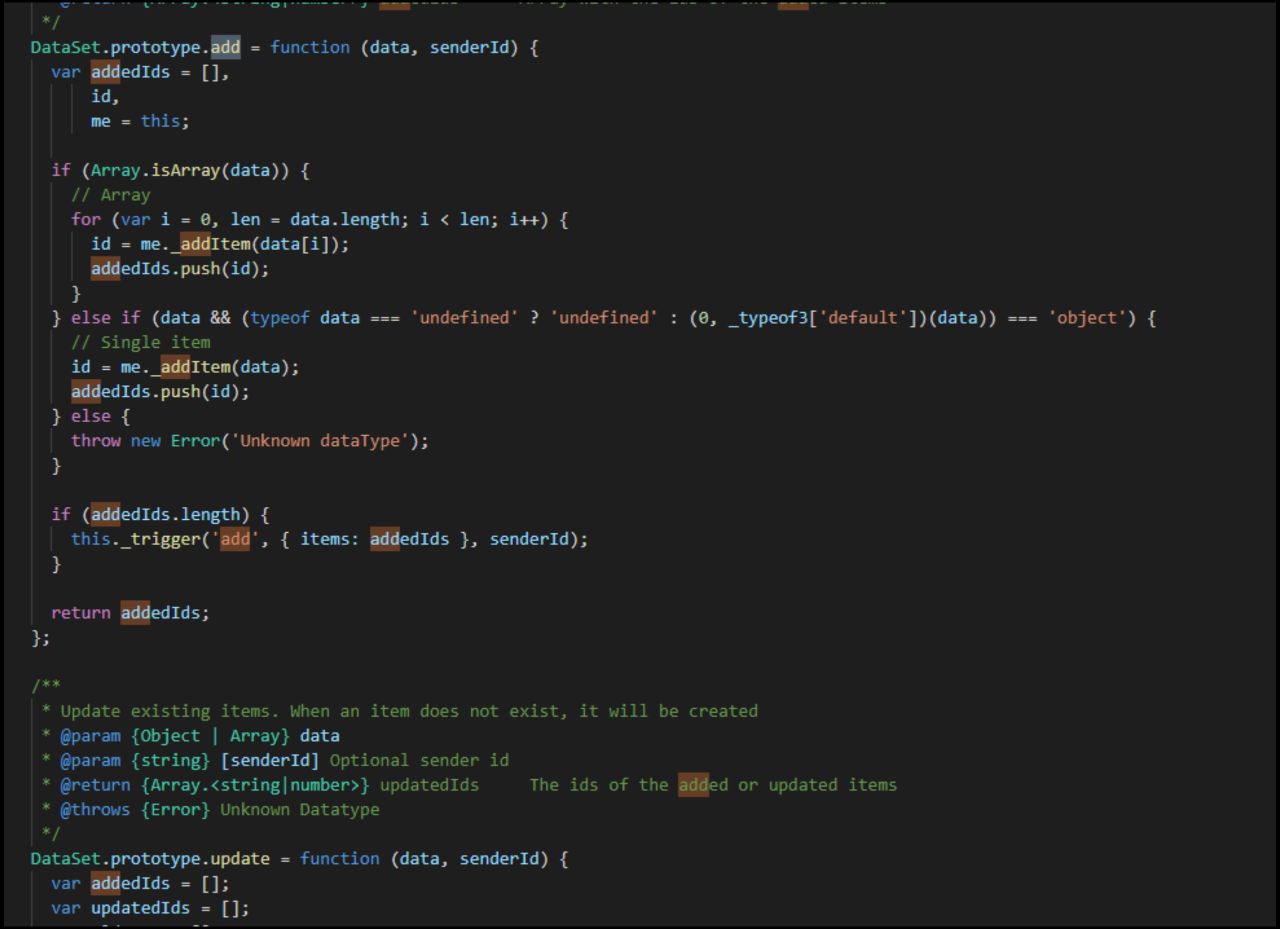
Librería usada:

VIS :

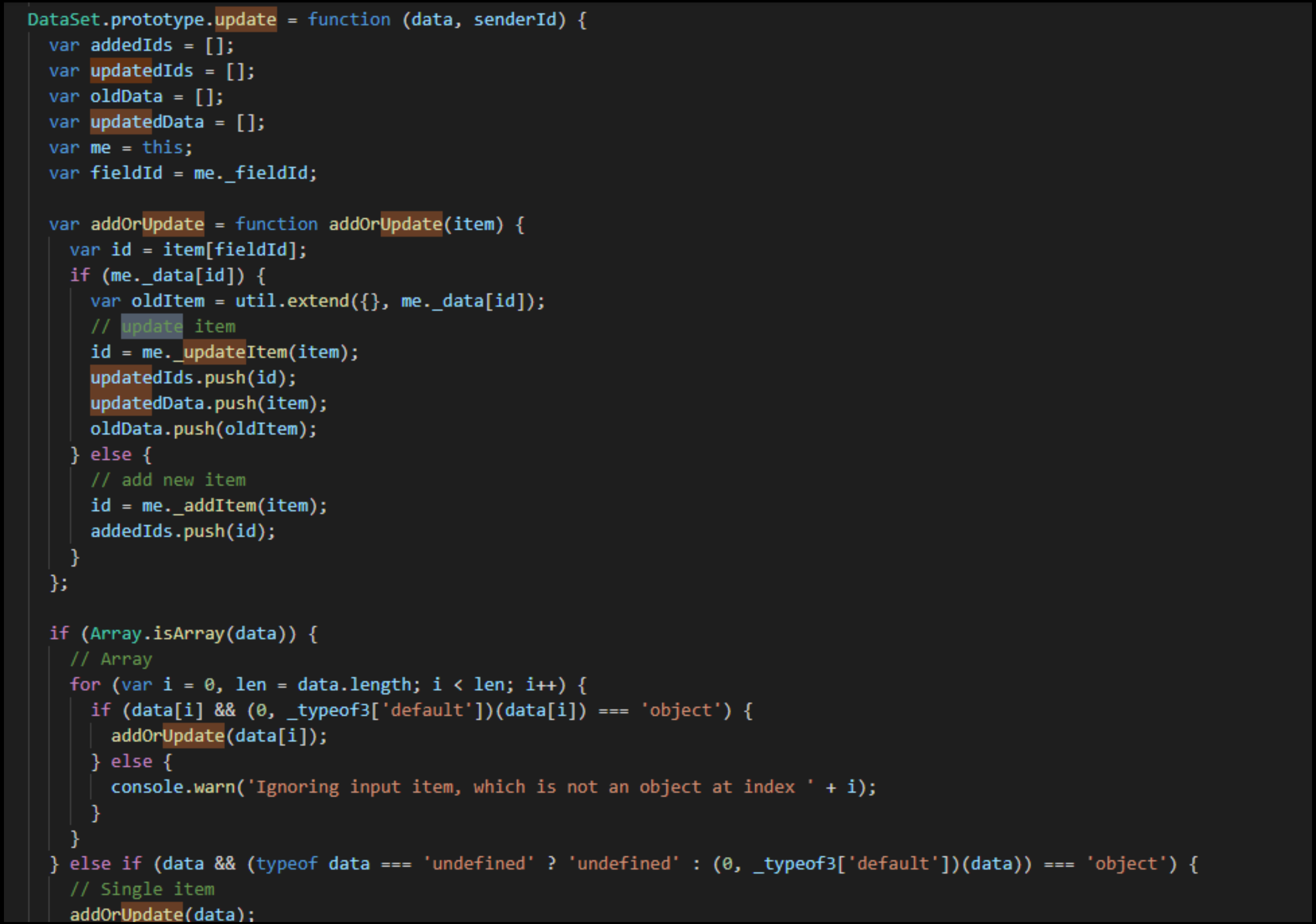
- VIS.JS



```
 * @constructor DataSet
 */
function DataSet(data, options) {
  // correctly read optional arguments
  if (data && !Array.isArray(data)) {
    options = data;
    data = null;
  }

  this._options = options || {};
  this._data = {}; // map with data indexed by id
  this.length = 0; // number of items in the DataSet
  this._fieldId = this._options.fieldId || 'id'; // name of the field containing id
  this._type = {}; // internal field types (NOTE: this can differ from this._options.type)

  // all variants of a Date are internally stored as Date, so we can convert
  // from everything to everything (also from ISODate to Number for example)
  if (this._options.type) {
    var fields = (0, _keys2['default'])(this._options.type);
    for (var i = 0, len = fields.length; i < len; i++) {
      var field = fields[i];
      var value = this._options.type[field];
      if (value == 'Date' || value == 'ISODate' || value == 'ASPDate') {
        this._type[field] = 'Date';
      } else {
        this._type[field] = value;
      }
    }
  }

  this._subscribers = {}; // event subscribers

  // add initial data when provided
  if (data) {
    this.add(data);
  }

  this.setOptions(options);
```

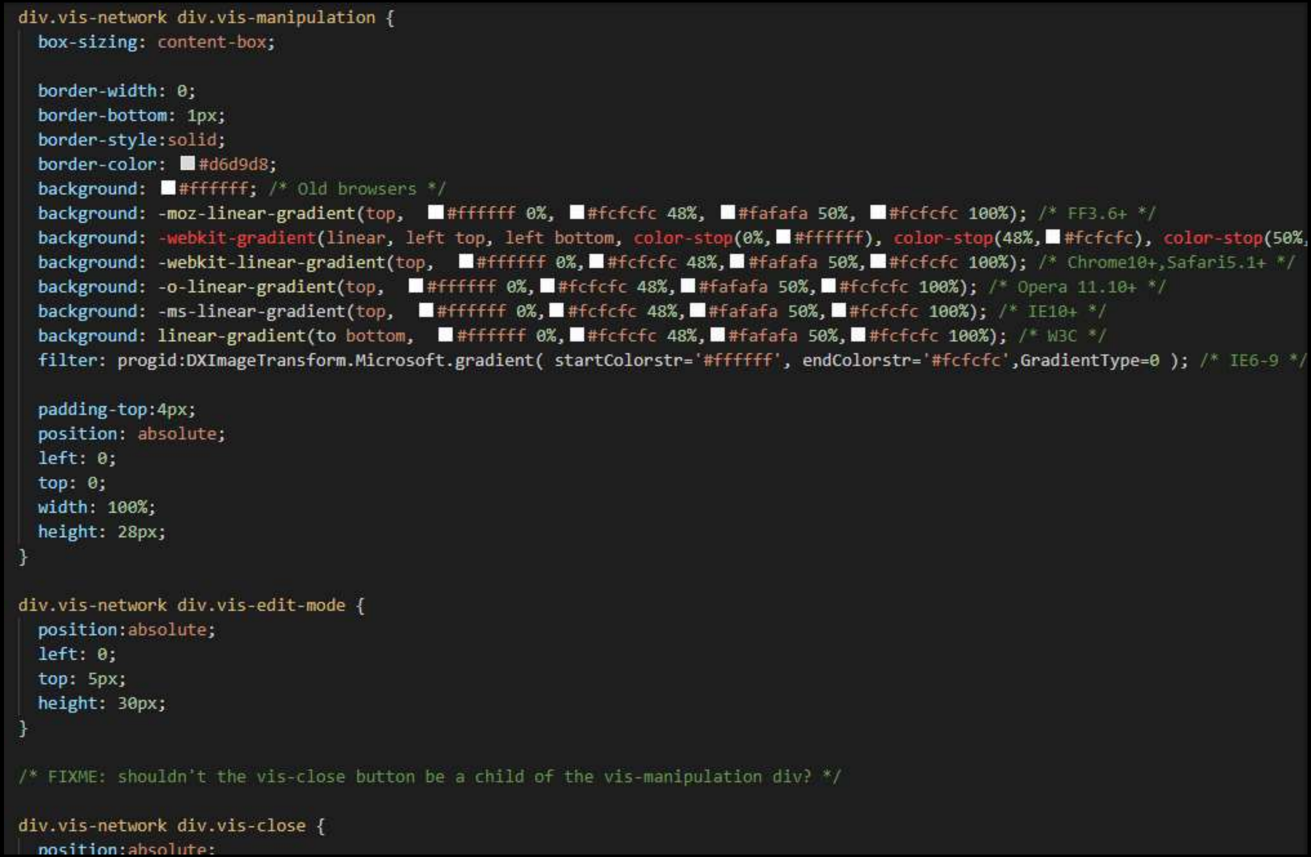


```
 */
DataSet.prototype.add = function (data, senderId) {
  var addedIds = [],
      id,
      me = this;

  if (Array.isArray(data)) {
    // Array
    for (var i = 0, len = data.length; i < len; i++) {
      id = me._addItem(data[i]);
      addedIds.push(id);
    }
  } else if (data && (typeof data === 'undefined' ? 'undefined' : (0, _typeof3['default'])(data)) === 'object') {
    // Single item
    id = me._addItem(data);
    addedIds.push(id);
  } else {
    throw new Error('Unknown dataType');
  }

  if (addedIds.length) {
    this._trigger('add', { items: addedIds }, senderId);
  }

  return addedIds;
};

/**
 * Update existing items. When an item does not exist, it will be created
 * @param {Object | Array} data
 * @param {string} [senderId] Optional sender id
 * @return {Array.<string|number>} updatedIds     The ids of the added or updated items
 * @throws {Error} Unknown Datatype
 */
DataSet.prototype.update = function (data, senderId) {
  var addedIds = [];
  var updatedIds = [];
```



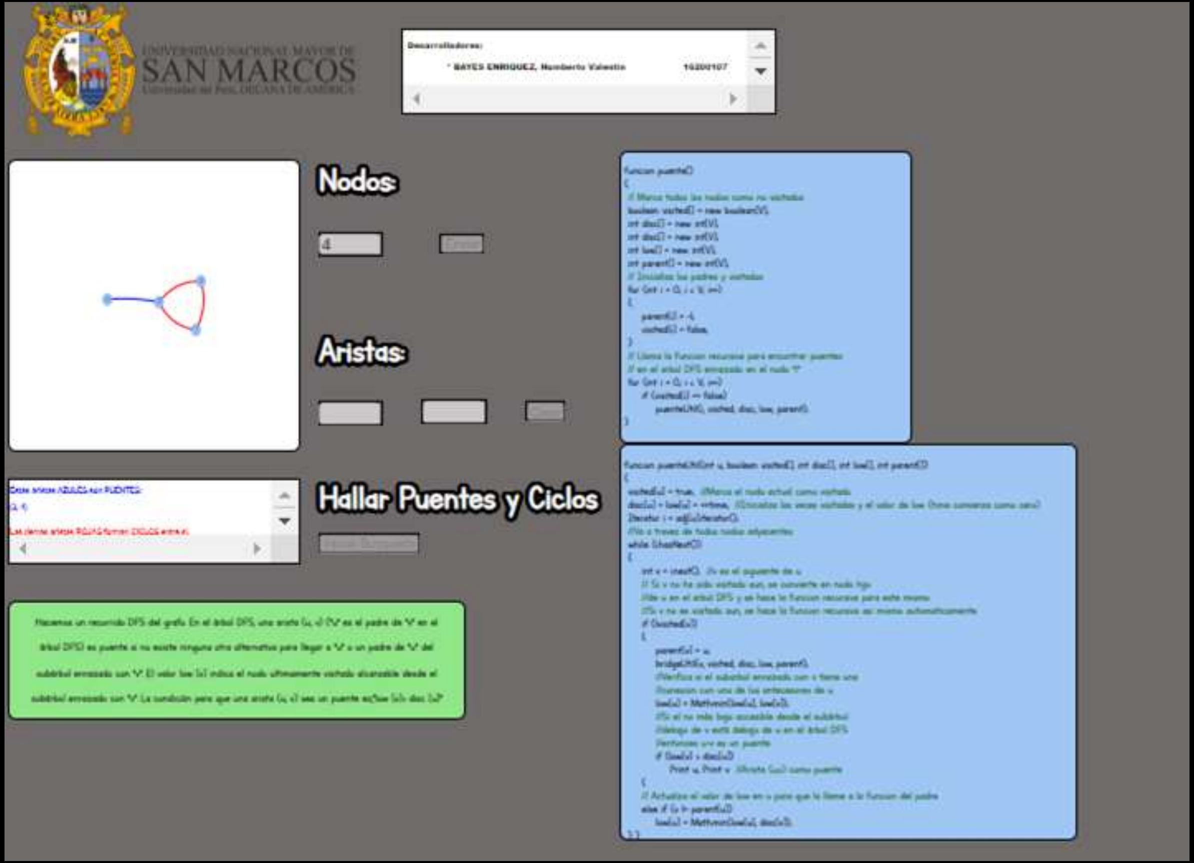
```
DataSet.prototype.update = function (data, senderId) {
  var addedIds = [];
  var updatedIds = [];
  var oldData = [];
  var updatedData = [];
  var me = this;
  var fieldId = me._fieldId;

  var addOrUpdate = function addOrUpdate(item) {
    var id = item[fieldId];
    if (me._data[id]) {
      var oldItem = util.extend({}, me._data[id]);
      // update item
      id = me._updateItem(item);
      updatedIds.push(id);
      updatedData.push(item);
      oldData.push(oldItem);
    } else {
      // add new item
      id = me._addItem(item);
      addedIds.push(id);
    }
  };

  if (Array.isArray(data)) {
    // Array
    for (var i = 0, len = data.length; i < len; i++) {
      if (data[i] && (0, _typeof3['default'])(data[i]) === 'object') {
        addOrUpdate(data[i]);
      } else {
        console.warn('Ignoring input item, which is not an object at index ' + i);
      }
    }
  } else if (data && (typeof data === 'undefined' ? 'undefined' : (0, _typeof3['default'])(data)) === 'object') {
    // Single item
    addOrUpdate(data);
```

- VIS.CSS



```css
div.vis-network div.vis-manipulation {
  box-sizing: content-box;

  border-width: 0;
  border-bottom: 1px;
  border-style:solid;
  border-color: #d6d9d8;
  background: #ffffff; /* Old browsers */
  background: -moz-linear-gradient(top,  #ffffff 0%, #fcfcfc 48%, #fafafa 50%, #fcfcfc 100%); /* FF3.6+ */
  background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#ffffff), color-stop(48%,#fcfcfc), color-stop(50%,
  background: -webkit-linear-gradient(top,  #ffffff 0%,#fcfcfc 48%,#fafafa 50%,#fcfcfc 100%); /* Chrome10+,Safari5.1+ */
  background: -o-linear-gradient(top,  #ffffff 0%,#fcfcfc 48%,#fafafa 50%,#fcfcfc 100%); /* Opera 11.10+ */
  background: -ms-linear-gradient(top,  #ffffff 0%,#fcfcfc 48%,#fafafa 50%,#fcfcfc 100%); /* IE10+ */
  background: linear-gradient(to bottom,  #ffffff 0%,#fcfcfc 48%,#fafafa 50%,#fcfcfc 100%); /* W3C */
  filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#ffffff', endColorstr='#fcfcfc',GradientType=0 ); /* IE6-9 */

  padding-top:4px;
  position: absolute;
  left: 0;
  top: 0;
  width: 100%;
  height: 28px;
}

div.vis-network div.vis-edit-mode {
  position:absolute;
  left: 0;
  top: 5px;
  height: 30px;
}

/* FIXME: shouldn't the vis-close button be a child of the vis-manipulation div? */

div.vis-network div.vis-close {
  position:absolute;
```

DEMOSTRACIÓN DE LA APLICACIÓN


UNIVERSIDAD NACIONAL MAYOR DE
SAN MARCOS
Universidad del Perú, DECANA DE AMÉRICA
Nodos:
4
Crear
Aristas:
Hallar Puentes y Ciclos
Hallar Puentes

Desarrolladores:

* BAYES ENRIQUEZ, Humberto Valentín 16200107

Estas aristas AZULES son PUENTES:

(3, 4)

Las demas aristas ROJAS forman CICLOS entre sí.

Hacemos un recorrido DFS del grafo. En el árbol DFS, una arista (u, v) ('u' es el padre de 'v' en el árbol DFS) es puente si no existe ninguna otra alternativa para llegar a 'u' o un padre de 'u' del subárbol enraizado con 'v'. El valor low [v] indica el nodo últimamente visitado alcanzable desde el subárbol enraizado con 'v'. La condición para que una arista (u, v) sea un puente es, "low [v] > disc [u]".

```
funcion puente()
{
 // Marca todos los nodos como no visitados
 boolean visited[] = new boolean[V];
 int disc[] = new int[V];
 int disc[] = new int[V];
 int low[] = new int[V];
 int parent[] = new int[V];
 // Inicializa los padres y visitados
 for (int i = 0; i < V; i++)
 {
    parent[i] = -1;
    visited[i] = false;
 }
 // Llama la funcion recursiva para encontrar puentes
 // en el arbol DFS enraizado en el nodo "i"
 for (int i = 0; i < V; i++)
    if (visited[i] == false)
        puenteUtil(i, visited, disc, low, parent);
}
```

```
funcion puenteUtil(int u, boolean visited[], int disc[], int low[], int parent[])
{
 visited[u] = true;  //Marca el nodo actual como visitado
 disc[u] = low[u] = ++time;  //Inicializa los veces visitados y el valor de low (time comienza como cero)
 Iterator i = adj[u].iterator();
 //Va a traves de todos nodos adyacentes
 while (i.hasNext())
 {
    int v = i.next();  //v es el siguiente de u
    // Si v no ha sido visitado aun, se convierte en nodo hijo
    //de u en el arbol DFS y se hace la funcion recursiva para este mismo
    //(Si v no es visitado aun, se hace la funcion recursiva asi mismo automaticamente
    if (!visited[v])
    {
        parent[v] = u;
        bridgeUtil(v, visited, disc, low, parent);
        //Verifica si el subarbol enraizado con v tiene una
        //conexion con uno de los antecesores de u
        low[u] = Math.min(low[u], low[v]);
        //Si el no más bajo accesible desde el subarbol
        //debajo de v está debajo de u en el arbol DFS
        //entonces u-v es un puente
        if (low[v] > disc[u])
            Print u, Print v  //Arista (u,v) como puente
    {
    // Actualiza el valor de low en u para que lo llame a la funcion del padre
    else if (v != parent[u])
        low[u] = Math.min(low[u], disc[v]);
}}
```