Tree-Search 1.0
Manual Técnico

indice

Tecnologías:....3
Bibliotecas:....3
    vis.js....3
Módulos....3
Arbol de archivos:....3
Codigo:....4
Repositorio git-hub....10

Tecnologías:

Al ser esta una pagina web exclusivamente front-end las tecnologías utilizadas fueron:

- Html
- Javascript
- Css

Bibliotecas:

vis.js

Esta completa biblioteca te permite crear grafos dinámicos entre otras muchas cosas. Disponible en: http://visjs.org/

Módulos

La aplicación esta desarrollada bajo el modelo vista controlador por lo que se subdivide en tres módulos:

- ui.js : Encargado de la interfaz gráfica del usuario.
- search.js: Contiene los algoritmos de búsqueda y el modelado del grafo.
- sim.js: Es el que lleva a cabo la simulación del recorrido.

ui.js (obtiene el gráfico) → search.js(Procesa y devuelve el recorrido) → sim.js(ejecuta el ciclo del recorrido) → ui.js (re-dibuja el gráfico).

Arbol de archivos:

/tree-search

- index.html

/js

- ui.js
- search.js
- sim.js

/css

- style.css

Codigo:

index.html

```
<!DOCTYPE HTML>
<!--
Author: David Ruiz Garcia
github-repository: https://github.com/david195/tree-search
git-page: https://david195.github.io/tree-search/
-->

<html>
<head>
  <title>Tree-Search</title>
  <meta name="author" content="David Ruiz">
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="css/style.css">
  <script src="js/ui.js"></script>
  <script src="js/search.js"></script>
  <script src="js/sim.js"></script>
  <script src="js/lib/vis.min.js"></script>
</head>

<body onload="load()">
<div id = 'layout'>
  <p>Métodos ciegos de búsqueda</p>
  <hr>
<div id="edit">
        <button    onclick="network.addNodeMode()">nuevo    nodo</button> <input    type="text"
requiered="true" id='nnode'><br>
      <button   onclick="network.addEdgeMode()">nueva   arista</button> <input   id='edge_num'
type="number" min="0" value="0"><br>
  <button onclick="network.deleteSelected()">eliminar seleccion</button><br><br>
         <label     for="file">Leer     archivo:     </label><input     id="file"     type="file"
onchange="load_file(this.files)"><br><br>
        <button    onclick="save_file()">guardar</button> <input    type="text"    id="fname"
requiered="true"> .json<br>
<br><br>
  <button id="binit" onclick="e_set('init')">init</button>
  <button id="bend" onclick="e_set('end')">end</button><br><br>
  <select id="opt">
    <option value="profundidad">Preferente por profundidad</option>
    <option value="proflim">Por profundidad limitada</option>
    <option value="amplitud">Preferente por amplitud</option>
    <option value="profI">Profundidad iterativa</option>
    <option value="costo">Costo uniforme</option>
  </select> <input id='prof' type="number" min="0" value="0">
  <br><br>
  <button onclick="init()">Busqueda</button>
  <div id="data"></div>
</div>
<div id='graf'>Modo Árbol:      
  <label class="switch">
    <input type="checkbox" id="sGa" onchange="checkGa()">
    <div class="slider round"></div>
  </label>
<div id="tree"></div> Simulación de recorrido                
<button id='pause' onclick="pause()">pausa</button>
```

```
<div id="sim"></div>
</div>
</div>
</body>
</html>
```

```
*ui.js*/
var network;
var e_init,e_end;
var on_sim = false;
var data;
var nn = 0;
var opt;
var gerarquia = false;

function load(){
  data ={
    nodes:[],
    edges:[]
  };
  data.nodes = new vis.DataSet(data.nodes);
  data.edges = new vis.DataSet(data.edges);
  network = tree(data,document.getElementById('tree'));
}

function load_file(arg) {
  var file = arg[0];
  var reader = new FileReader();
  reader.onload = function(){
    data = JSON.parse(reader.result);
    data.nodes = new vis.DataSet(data.nodes);
    data.edges = new vis.DataSet(data.edges);
    network = tree(data,document.getElementById('tree'));
  };
  reader.readAsText(file);
}

function tree(data,container){
  if(e_init!=null || e_end!=null && e_init!=e_end){
    for (var id in data.nodes._data) {
      if (data.nodes._data.hasOwnProperty(id)) {
        if(data.nodes._data[id].id==e_init){
          data.nodes._data[id].group = "init";
        }
        else if(data.nodes._data[id].id==e_end && !on_sim){
          data.nodes._data[id].group = "end";
        }
        else {
          if(!on_sim)
            data.nodes._data[id].group = null;
        }
      }
    }
  }
  var dat = {
    nodes: data.nodes,
    edges: data.edges
  };
  var options = {
    layout:{
      randomSeed: .5,
      hierarchical: gerarquia
    },
    groups:{init:{color:{background:'red'}},
            end:{color:{background:'green'}}
```

```
    },
    manipulation: {
      enabled: false,
      addNode: function(nodeData,callback) {
        nodeData.label = document.getElementById('nnode').value;
        nodeData.id = nn;
        nn++;
        data.nodes.add(nodeData);
        callback(nodeData);
      },
      addEdge: function(edgeData,callback) {
        //if(opt=="costo"){
          edgeData.label = document.getElementById('edge_num').value;
        //}
        data.edges.add(edgeData);
        callback(edgeData);
      }
    }
  };
  nn = data.nodes.length+1;
  var nw = new vis.Network(container, dat, options);
  return nw;
}

function e_set(t){
  var node = network.getSelectedNodes()[0];
  if(t=="init"){
    e_init = node;
  }
  else{
    e_end = node;
  }
  network = tree(data,document.getElementById('tree'));
}

function init(){
  opt = document.getElementById('opt').options[document.getElementById('opt').selectedIndex].value;
  search(e_init,e_end,opt,data);
}

function descargarArchivo(contenidoEnBlob, nombreArchivo) {
    var reader = new FileReader();
    reader.onload = function (event) {
        var save = document.createElement('a');
        save.href = event.target.result;
        save.target = '_blank';
        save.download = nombreArchivo || 'archivo.dat';
        var clicEvent = new MouseEvent('click', {
            'view': window,
                'bubbles': true,
                'cancelable': true
        });
        save.dispatchEvent(clicEvent);
        (window.URL || window.webkitURL).revokeObjectURL(save.href);
    };
    reader.readAsDataURL(contenidoEnBlob);
};

function save_file(){
  var a = {
    'nodes':[data.nodes._data],
    'edges':[data.edges._data]
  }
  var edges=[];
  for (var id in data.edges._data) {
```

```
    var e = {};
    e.to = data.edges._data[id].to;
    e.from = data.edges._data[id].from;
    e.label = data.edges._data[id].label;
    edges.push(e);
  }
  var nodes=[];
  for (var id in data.nodes._data) {
    var n = {};
    n.id = data.nodes._data[id].id;
    n.label = data.nodes._data[id].label;
    nodes.push(n);
  }
  var d = {
    nodes:nodes,
    edges:edges
  };
  var j = JSON.stringify(d);
  var b = new Blob([j], {
        type: 'application/json'
    });
  descargarArchivo(b,document.getElementById('fname').value+".json");
}

function checkGa(){
  gerarquia = document.getElementById('sGa').checked;
  network = tree(data,document.getElementById('tree'));
}

function pause(){
  if(pausa)
    document.getElementById('pause').innerHTML = 'pausa';
  else
    document.getElementById('pause').innerHTML = 'reanudar';
  pausa = !pausa;
}
```

```
/*search.js*/
var lnodes,rnodes,limp,lastn;

function search(ei,ef,tsearch,data){
  if(ei == null || ef == null){
    alert("Elige un nodo de inicion y un nodo meta");
    return;
  }
  if(tsearch == 'profI'){
    profI();
    return;
  }
  lnodes = [];
  rnodes = [];
  var np=0;
  lnodes.push(ei);
  if(tsearch == 'costo'){
    costo(ef);
    return;
  }
  do{
    n = eval(tsearch+"()");
    rnodes.push(n);
    for (var id in data.edges._data) {
      if(n<=ef){
        if(data.edges._data[id].from == n)
          lnodes.push(data.edges._data[id].to);
```

```
      }
      else {
        if(data.edges._data[id].to == n)
          lnodes.push(data.edges._data[id].from);
      }
    }
    np++;
    if(tsearch=='proflim' && np>=limp){
      alert("Fallo, nodo meta no encontrado");
      break;
    }
  }while (n!=ef);
  rnodes = rnodes.unique();
                                                  var                          tsh                          =
document.getElementById('opt').options[document.getElementById('opt').selectedIndex].innerHTML;
  var txt="Búsqueda "+tsh+"<br><br>Ruta solucion: <br>";
  for(var i=0;i<rnodes.length;i++){
    txt+=data.nodes._data[rnodes[i]].label+" -> ";
  }
  txt+="<br><br>Numero de pasos: "+np+"<br>";//<br>Nodos visitados: <ul>";
  txt = "</ul><p>"+txt+"<br></p>";
  document.getElementById('data').innerHTML = txt;
  console.log(rnodes);
  console.log(lnodes);
  console.log(np);
  sim(rnodes,data,1000);
}

function profundidad(){
  return lnodes.pop();
}

function amplitud(){
  return lnodes.shift();
}

function proflim(){
  limp = parseInt(document.getElementById('prof').value);
  return profundidad();
}

function profI(){
  li = document.getElementById('prof').value;
  document.getElementById('prof').value=0;
  search(e_init,e_end,'proflim',data);
}

function costo(ef){
  var ns = [];
  var vr = [];
  var v = 0;
  var n = lnodes.pop();
  ns.push(n);
  do{
    for (var i in data.edges._data) {
      if(data.edges._data[i].label==null){
        alert("Todas las aristas deben tener un costo");
        return;
      }
      var p1,p2;
      if(data.edges._data[i].from == n){
        p1 = data.edges._data.to;
        p2 = parseInt(data.edges._data[i].label)+v;
      }
      if(data.edges._data[i].to == n){
```

```
        p1 = data.edges._data[i].from;
        p2 = parseInt(data.edges._data[i].label)+v;
      }
      if(p1!=null && p2!=null){
        console.log(p1);
        console.log(p2);
        lnodes.push(p1);
        vr.push(p2);
      }
    }
    var ind = vr.indexOf(vr.min());
    v = vr[ind];
    n = lnodes[ind];
    if(ns.indexOf(n)==-1)
      ns.push(n);
    vr.splice(ind,1);
    lnodes.splice(ind,1);
  }while(n!=ef);
                                                  var                          tsh                          =
document.getElementById('opt').options[document.getElementById('opt').selectedIndex].innerHTML;
  var txt="Búsqueda "+tsh+"<br><br>Ruta solucion: <br><br>Costo total de ruta: "+v+"<br><br>";
  for(var i=0;i<ns.length;i++){
    txt+=data.nodes._data[ns[i]].label+" -> ";
  }
  txt+="<br><br>Numero de pasos: "+ns.length+"<br>";//<br>Nodos visitados: <ul>";
  txt = "</ul><p>"+txt+"<br></p>";
  document.getElementById('data').innerHTML = txt;
  console.log(ns);
  console.log(v);
  sim(ns,data,1000);
}

Array.prototype.unique = function() {
    var unique = [];
    for (var i = 0; i < this.length; i++) {
        if (unique.indexOf(this[i]) == -1) {
            unique.push(this[i]);
        }
    }
    return unique;
};

Array.prototype.min = function() {
  return Math.min.apply(null, this);
};
```

```
/*sim.js*/
var lp,nd,n=0;
var pausa = false;

function sim(nv,data,t){
  clearInterval(lp);
  document.getElementById('sim').innerHTML='';
  nd = document.createElement('div');
  document.getElementById('sim').appendChild(nd);
  lp = setInterval(function(){
    loop(nv,data);
  },t);
}

function sim_off(){
  if(lp!=null)
    clearInterval(lp);
}
```

```
function loop(nv,data){
  on_sim = true;
  if(n<nv.length){
    for (var id in data.nodes._data) {
      if(data.nodes._data[id].id==nv[n])
        data.nodes._data[id]['group'] = 'init';
    }
  }
  else {
    n=0;
    for (var id in data.nodes._data) {
      data.nodes._data[id].group=null;
    }
    if(opt=='profI' && limp<=li){
      document.getElementById('prof').value = limp+1;
      clearInterval(lp);
      alert("iteracion: "+limp);
      search(e_init,e_end,'proflim',data);
    }
  }
  if(!pausa)
    n++;
  var netw = tree(data,nd);
}
```

Repositorio git-hub

https://github.com/david195/tree-search