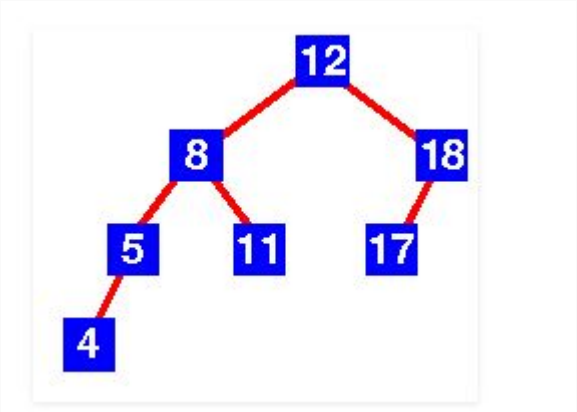
Binary Search Tree

AVL Tree

¿Qué es un AVL Tree?

Un AVL Tree es un BST autobalanceado, en el cual la diferencia entre las alturas de los subárboles izquierdo y derecho no puede ser mayor que 1 para todos los nodos.



Funciones clave

- Búsqueda de elementos
- Insertar elemento
- Eliminar elemento
- Rotaciones

Software can be chaotic, but we make it work

Expert

Trying Stuff Until it Works

O RLY?

The Practical Developer
@ThePracticalDev

Clases:
Vertex
Árbol

Funciones:
Init
Agregar

```
lass vertex:
    def __init__(self,v):
        self.id = v
        self.padre = None
        self.hizq = None
        self.hder = None
        self.altura = -1
        self.profundidad = -1
        self.FE = 0

lass arbol:
    def __init__(self):
        self.raiz = None

    def agregar(self,act, ver):
        if ver.id > act.id:          #Vertice mayor que actual
            if act.hder != None:     #Si hay hDer aplicar recursividad
                self.agregar(act.hder,ver)
            else:                    #Si no hay hDer se inserta como su
                act.hder = ver
                ver.padre = act
        else:                        #Vertice menor que actual
            if act.hizq != None:     #Si hay hIzq aplicar recursividad
                self.agregar(act.hizq,ver)
            else:                    #Si no hay hIzq se inserta como su
                act.hizq = ver
                ver.padre = act
```

Funciones:

- Inserción de vértice
- Eliminación de vértice
- Creación del árbol

```
37      def insertarVertice(self,v):
38          ver = vertex(v)
39          if self.raiz == None:
40              self.raiz = ver
41          else:
42              self.agregar(self.raiz, ver)
43
44      def eliminarVertice(selv,v):
45          pass
46
47      def crearArbol(self,listVer):
48          for i in range(len(listVer)):
49              self.insertarVertice(listVer[i])
```

Funciones:

- Generación de Json

```
51      def generarJSON(self,JSON=[],act = False,): #recorrido infix
52          #global JSON
53          if act is False: act = self.raiz
54          if act != None:
55              self.generarJSON(JSON,act.hder)
56
57              if act.padre == None:
58                  padre = "" #Instead of NONE
59              else:
60                  padre = act.padre.id
61              if act.hizq == None:
62                  hIzq = "None"
63              else:
64                  hIzq = act.hizq.id
65              if act.hder == None:
66                  hDer = "None"
67              else:
68                  hDer = act.hder.id
69
70              JSON.append({ 'name': str(act.id),'parent': str(padre)})
71
72              self.generarJSON(JSON,act.hizq)
73              return JSON
```

Funcion:
Alturas

```
97      def alturas(self,act = False,prof = 0): #Recorrido postorden,calcula
98      if act is False: act = self.raiz        #la altura de las hojas a la raiz, y la profundidad
99          if act != None:
100             act.profundidad = prof;
101             a1 = self.alturas(act.hizq,prof+1)
102             a2 = self.alturas(act.hder,prof+1)
103             #print(max([a1,a2]))
104             act.altura = max([a1,a2]) + 1
105             return act.altura
106         else:
107             return -1
108
```

Funciones:

- calcFe - Cálculo del factor de equilibrio

```
109     def calcFE(self,act = False): #Factor de equilibrio
110         if act is False: act = self.raiz
111         if act != None:
112             self.calcFE(act.hizq)
113             if act.hizq != None and act.hder != None:
114                 act.FE = act.hder.altura - act.hizq.altura
115             elif (act.hizq == None and act.hder != None):
116                 act.FE = act.hder.altura + 1
117             elif (act.hizq != None and act.hder == None):
118                 act.FE = -act.hizq.altura  - 1
119             else:
120                 act.FE = 0
121             self.calcFE(act.hder)
```

Funciones:

- Desvalanceados

```
123     def imprimirDesv(self,arrDesv,act = False):
124         if act is False:
125             act = self.raiz
126             self.alturas()
127             self.calcFE()
128         if act != None:
129             self.imprimirDesv(arrDesv,act.hizq)
130             if act.FE > 1:
131                 #print(act.id)
132                 arrDesv.append([act.id,"der"])
133             elif act.FE < -1:
134                 #print(act.id)
135                 arrDesv.append([act.id,"izq"])
136             self.imprimirDesv(arrDesv,act.hder)
137         return arrDesv
```

Funciones: Generar JSON

```
139     def obtenerOBJDesv(self,arrDesv,act = False): #Devuelve los vertices
140         if len(arrDesv) < 1:                       # desviados como OBJ
141             if act is False:
142                 act = self.raiz
143                 self.alturas()
144                 self.calcFE()
145             if act != None:
146                 self.obtenerOBJDesv(arrDesv,act.hizq)
147                 if len(arrDesv) < 1:
148                     if act.FE > 1:
149                         #print(act.id)
150                         arrDesv.append([act,"der"])
151                     elif act.FE < -1:
152                         #print(act.id)
153                         arrDesv.append([act,"izq"])
154                     self.obtenerOBJDesv(arrDesv,act.hder)
155             return arrDesv
```

Funciones:

- LR - Rotación a la izquierda

```
def LR(self,n,act = False): #Rotacion a la izquierda
    if act is False: act = self.raiz
    #print("act: ", act.id)
    if (act.hder != None) and act.id == n :
        #Saber si actual es hIzq o IDer de su padre
        if act.padre != None:
            if act.padre.hizq == act:
                ladoRef = "izq"
            else:
                ladoRef = "der"

        nr = act.hder   #nr = Nodo Referencia
        act.hder = nr.hizq
        nr.hizq = act
        nr.padre = act.padre
        act.padre = nr
        if act.hder != None:
            act.hder.padre = act

        if nr.padre != None:
            if ladoRef == "izq":
                nr.padre.hizq = nr
            elif ladoRef == "der":
                nr.padre.hder = nr

        if self.raiz == act:
            self.raiz = nr
    else:
        if act.hizq != None and act.hder != None:
            self.LR(n,act.hizq)
            self.LR(n,act.hder)
        elif act.hizq != None and act.hder == None:
            self.LR(n,act.hizq)
        elif act.hizq == None and act.hder != None:
            self.LR(n,act.hder)
```

Funciones:

- RR - Rotación a la derecha

```
def RR(self,n,act = False): #Rotacion a la derecha (n = int)
    if act is False: act = self.raiz
    #print("act: ", act.id)
    if (act.hizq != None) and act.id == n :
        #Saber si actual es hIzq o IDer de su padre
        if act.padre != None:
            if act.padre.hizq == act:
                ladoRef = "izq"
            else:
                ladoRef = "der"

        nr = act.hizq   #nr = Nodo Referencia
        act.hizq = nr.hder
        nr.hder = act
        nr.padre = act.padre
        act.padre = nr
        if act.hizq != None:
            act.hizq.padre = act

        if nr.padre != None:
            if ladoRef == "izq":
                nr.padre.hizq = nr
            elif ladoRef == "der":
                nr.padre.hder = nr

        if self.raiz == act:
            self.raiz = nr
    else:
        if act.hizq != None and act.hder != None:
            self.RR(n,act.hizq)
            self.RR(n,act.hder)
        elif act.hizq != None and act.hder == None:
            self.RR(n,act.hizq)
        elif act.hizq == None and act.hder != None:
            self.RR(n,act.hder)
```

Funciones:

- RDR - Doble rotación derecha
- LDR - Doble rotación izquierda

```
def RDR(self,o): #Rotación doble a la derecha (o = obj)
    self.LR(o.hizq.id)
    self.RR(o.id)

def LDR(self,o): #Rotación doble a la izquierda (o = obj)
    self.RR(o.hder.id)
    self.LR(o.id)
```

Funcion: Autobalanceo

```
239     def autobalanceo(self):
240
241         arrObjDesv = self.obtenerOBJDesv([])
242
243         while(len(arrObjDesv)>0):
244             ladoDesv = arrObjDesv[0][1]
245             objDesv = arrObjDesv[0][0]
246
247             if ladoDesv == "izq" and objDesv.hizq.FE == 1:
248                 self.RDR(objDesv)
249             elif ladoDesv == "der" and objDesv.hder.FE == -1:
250                 self.LDR(objDesv)
251             elif ladoDesv == "izq":
252                 self.RR(objDesv.id)
253             elif ladoDesv == "der":
254                 self.LR(objDesv.id)
255             else:
256                 #print("NO SE ENTRO A NINGUNA OPCION")
257                 pass
258
259             arrObjDesv = self.obtenerOBJDesv([])
260
```

Funciones:

- **buscarVertice - Búsqueda de elemento**

```
def buscarVertice(self,v, act = False): #Retorna un arreglo,
    if act is False:    #Encontrado:[1, objAct]; No encontrado:[0]
        act = self.raiz

    if act != None:
        if act.id == v:
            searchedItem.append(act.id)
            searchedItem.append(act.profundidad)
            return [1,act]
        else:
            if act.id < v:
                retorno = self.buscarVertice(v, act.hder)
            elif act.id > v:
                retorno = self.buscarVertice(v,act.hizq)
    else:
        searchedItem.append(-1)
        searchedItem.append(-1)
        return [0]

    return retorno
```

Funciones:

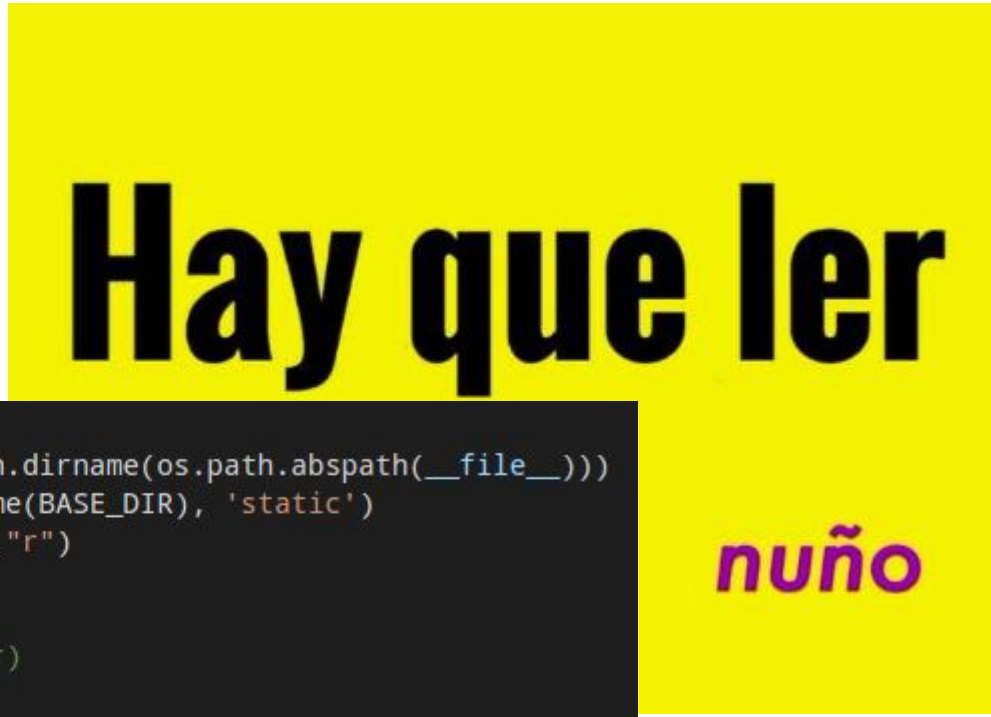
- obtenerMin - Obtención del elemento mínimo
- obtenerMax - Obtención del elemento máximo

```
def obtenerMin(self):
    act = self.raiz
    if self.raiz != None:
        while act.hizq:
            #print(act.id)
            act = act.hizq
        print("Elemento menor:", act.id)
        maxmin[1] = int(act.id)
        #return act.id

def obtenerMax(self):
    act = self.raiz
    if self.raiz != None:
        while act.hder:
            #print(act.id)
            act = act.hder
        print("Elemento mayor:", act.id)
        maxmin[0] = int(act.id)
        #return act.id
```

Funciones:

- leerArchivo - Lectura de archivo

```
def leerArchivo(self):
    BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
    rute = os.path.join(os.path.dirname(BASE_DIR), 'static')
    a = open( rute + '/leerArbol.txt',"r")
    arr = a.readlines()
    a.close()
    #print("Arreglo de prueba: " + arr)
    for line in arr:
        line = line.split(',')
        line = list(map(int, line))

    return line
```



Funciones:

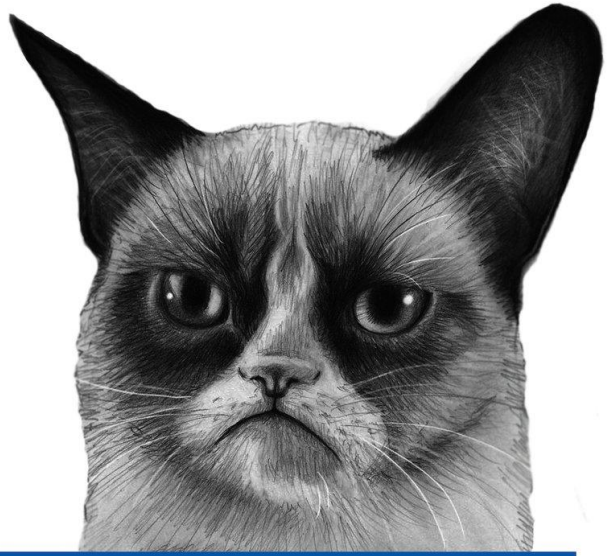
- guardarArchivo - Almacenamiento de archivo

```
def guardarArchivo(self,arr):
    BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
    rute = os.path.join(os.path.dirname(BASE_DIR), 'static')
    a = open( rute + '/leerArbol.txt',"w")
    #print("Arreglo " , arr)

    for item in range(len(arr)-1):
        a.write(str(arr[item])+",")
    a.write(str(arr[len(arr)-1]))

    a.close()
    #print("Arreglo de prueba: " + arr)
```

Django + D3.js

¿Cómo funciona la parte gráfica?

You're a 10x hacker and it must be someone else's fault.



Blaming the User

Pocket Reference

O RLY?

@ThePracticalDev

API Endpoints

```
from django.conf.urls import url
from .views import graph, data, get_tree, readFrom, saveTo, MaxMin,mySearch

urlpatterns = [
    url(r'^$', graph),
    url(r'^api/data', data, name='data'),
    url(r'^send/$',get_tree ),
    url(r'^api/read',readFrom),
    url(r'^api/save',saveTo),
    url(r'^api/MaxMin',MaxMin),
    url(r'^api/search',mySearch),

]

```

Initialization

```
from django.http import JsonResponse
from django.shortcuts import render

from django.http import HttpResponse
from django.views.decorators.csrf import csrf_exempt

from .BSTAuto import *

JSONlocal = [] #global
a = arbol()
volatil = [23, 54, 89, 39, 13, 36, 75, 14, 27,10,9,8,76,77,78,90]
a.crearArbol(volatil)
a.autobalanceo()
JSONlocal = a.generarJSON()
print('EJECUTANDO DESDE ARRIBA_____')
#This populates a global VAR named JSON.

def graph(request):
    return render(request, 'graph/graph.html')



def data(request, data={'void':'void'}):
    print('en data request')

    return JsonResponse([JSONlocal,volatil], safe=False)

```

Get Tree

```
18  @csrf_exempt
19  def get_tree(request):
20      # if this is a POST request we need to process the form data
21      sanitized = []
22      if request.method == 'POST':
23          print('METHOD POST')
24          arr = request.POST.getlist('arr[]')
25          for item in arr:
26              sanitized.append(int(item))
27
28          a = arbol()
29          a.crearArbol(sanitized)
30          a.autobalanceo()
31          JSONlocal = []
32          JSONlocal = a.generarJSON([]) #actualizamos el JSON
33          return JsonResponse( ([JSONlocal, sanitized]) , safe=False)
34      else:
35          return JsonResponse( (sanitized) , safe=False)
```

jQuery and Form

```
5   // jQuery helpers for FORM
6   $(document).ready(function () {
7       $('#SubmitData').click(function () {
8           var input = $('#getFromInput').val();
9           if (input === "") {
10              alert('Introduce numeros separados por coma');
11          } else {
12              var arr = input.split(",");
13              postArr(arr);
14          }
15
16      });
```

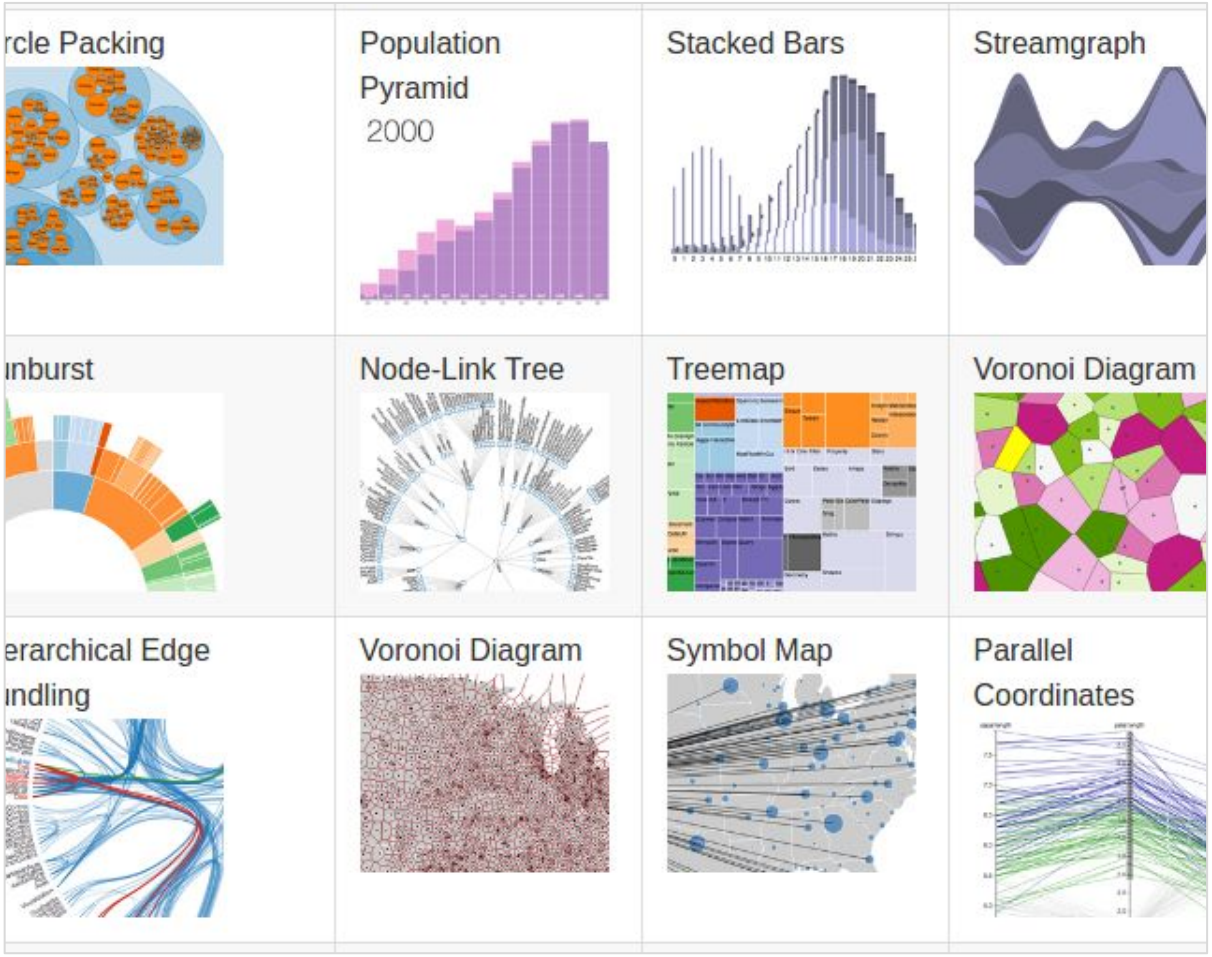
Post Array

```
153  function postArr(myArr) {
154      $.ajax({
155          type: "POST",
156          url: /send/,
157          data: { arr: myArr },
158          success: function (data) {
159              askForJSON(data);
160          },
161          dataType: 'json'
162      });
163  }
```

Ask for JSON

```
258  function askForJSON(data = []) {
259      var api;
260      if (data.length < 1) {
261          api = "/api/data";
262          d3.json(api, function (error, rawData) {
263              if (error) throw error;
264              todo(rawData[0]);
265              actualArray = rawData[1]; //save actual array
266          });
267      } else {
268          todo(data[0]);
269          actualArray = data[1]; //save actual array
270      }
271
272  } //askforJSON
```

D3.js



D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG, and CSS.

D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

```
root = d3.stratify()
    .id(function (d) {
        return d.name;
    })
    .parentId(function (d) {
        return d.parent;
    })
    (rawData);
```

```
console.log('Init roo after Promise')
root.x0 = height / 2;
root.y0 = 0;

function collapse(d) {
    if (d.children) {
        d._children = d.children;
        d._children.forEach(collapse);
        d.children = null;
    }
}

//root.children.forEach(collapse);
update(root);
```

```
311  function update(source) {
312
313      // Compute the new tree layout.
314      var nodes = tree.nodes(root).reverse(),
315          links = tree.links(nodes);
316
317      // Normalize for fixed-depth.
318      nodes.forEach(function (d) {
319          d.y = d.depth * 120;
320      });
321
322      // Update the nodes…
323      var node = svg.selectAll("g.node")
324          .data(nodes, function (d) {
325              return d.id || (d.id = ++i);
326          });
327
328      // Enter any new nodes at the parent's previous position.
329      var nodeEnter = node.enter().append("g")
330          .attr("class", "node")
331          .attr("transform", function (d) {
332              return "translate(" + source.y0 + "," + source.x0 + ")";
333          })
334          .on("click", click);
```

```
    // Transition nodes to their new position.
    var nodeUpdate = node.transition()
        .duration(duration)
        .attr("transform", function (d) {
            return "translate(" + d.y + "," + d.x + ")";
        });

    nodeUpdate.select("circle")
        .attr("r", 13.5)
        .style("fill", function (d) {
            return d._children ? "#1976D2" : "#BBDEFB";
        });

    nodeUpdate.select("text")
        .style("fill-opacity", 1);

    // Transition exiting nodes to the parent's new position.
    var nodeExit = node.exit().transition()
        .duration(duration)
        .attr("transform", function (d) {
            return "translate(" + source.y + "," + source.x + ")";
        })
        .remove();

    nodeExit.select("circle")
        .attr("r", 1e-6);

    nodeExit.select("text")
        .style("fill-opacity", 1e-6);

    // Update the links…
    var link = svg.selectAll("path.link")
        .data(links, function (d) {
            return d.target.id;
        });
```

FAQ

- ¿Qué es Async?
- ¿Qué es Javascript?
- ¿Qué es jQuery?
- ¿Qué es una Promesa en JS?
- ¿Qué es c9.io
- ¿Qué es Backend y Frontend?

Fin