

Tecnológico Nacional de México
Instituto Tecnológico de Iztapalapa

Alonso Ayala Angel de Jesus

161080-176

Sistemas Computacionales

ISC-8AV

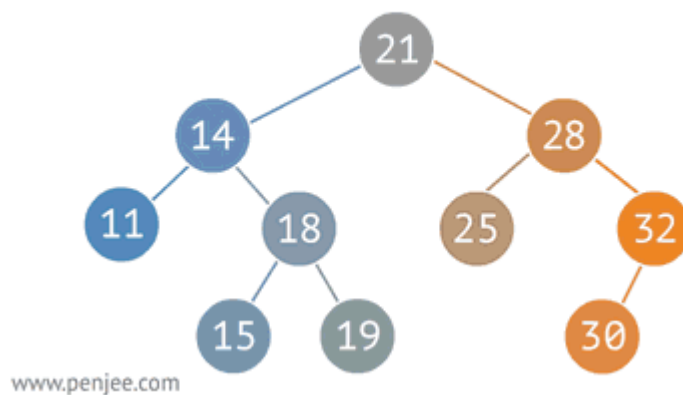
Abiel Tomas Parra Hernández

Inteligencia Artificial

Árbol binario de búsqueda (BST)

Un árbol binario de búsqueda también llamado BST (acrónimo del inglés Binary Search Tree) es un tipo particular de árbol binario que presenta una estructura de datos en forma de árbol usada en informática.

Un árbol binario de búsqueda (ABB) es un árbol binario con la propiedad de que todos los elementos almacenados en el subárbol izquierdo de cualquier nodo x son menores que el elemento almacenado en x , y todos los elementos almacenados en el subárbol derecho de x son mayores que el elemento almacenado en x .



Código:

```
import os

# Se definen los movimientos izquierda, derecha para los nodos
class node():
    def __init__(self, dato):
        self.left = None
        self.right = None
        self.dato = dato

class arbol():
    def __init__(self):
        self.root = None

    # Se crean las funciones para cada una de la opciones
    # Funcion para insertar nodos al arbol
    def insert(self, a, dato):
        if a == None:
            a = node(dato)
        else:
```

```
d = a.dato
if dato < d:
    a.left = self.insert(a.left, dato)
else:
    a.right = self.insert(a.right, dato)
return a

# Funcion para mostrar los nodos de menor a mayor
def inorder(self, a):
    if a == None:
        return None
    else:
        self.inorder(a.left)
        print(a.dato)
        self.inorder(a.right)

# Funcion para mostrar los nodos empezando por el nodo root hacia los
# Demas nodos padres
def preorder(self, a):
    if a == None:
        return None
    else:
        print(a.dato)
        self.preorder(a.left)
        self.preorder(a.right)

# Funcion para mostrar los nodos empezando por los nodos hijos hasta
# Llegar el nodo root
def postorder(self, a):
    if a == None:
        return None
    else:
        self.postorder(a.left)
        self.postorder(a.right)
        print(a.dato)

# Funcion que indicara si el nodo buscado se encuentra en el árbol o
# no
def buscar(self, dato, a):
    if a == None:
        return None
    else:
        if dato == a.dato:
            return a.dato
        else:
            if dato < a.dato:
                return self.buscar(dato, a.left)
            else:
                return self.buscar(dato, a.right)
```

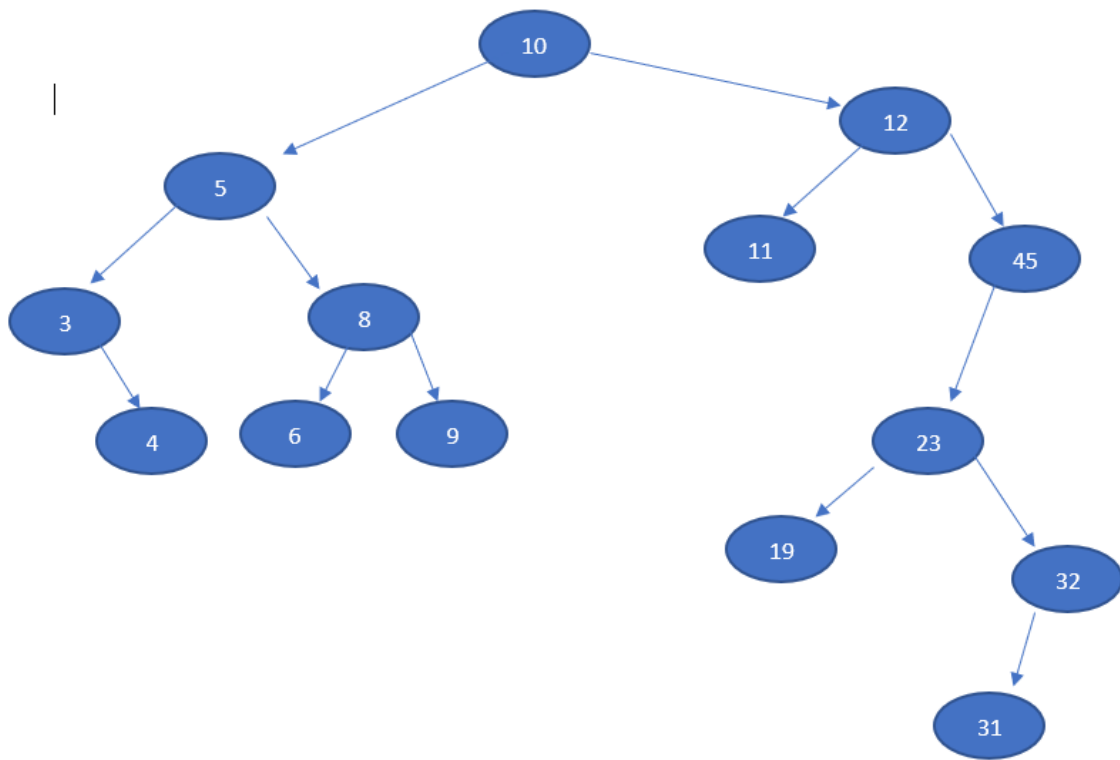
```
tree = arbol()
# Se crea el menu que tendrá todas las posibles opciones
while True:
    os.system("cls")
    print("Arbol ABB")
    opc = input("\n1.-Insertar nodo \n2.-Inorden \n3.-Preorden \n4.-
Postorden \n5.-Buscar \n6.-Salir \n\nElige una opcion -> ")

    if opc == '1':
        nodo = input("\nIngresa el nodo -> ")
        if nodo.isdigit():
            nodo = int(nodo)
            tree.root = tree.insert(tree.root, nodo)
        else:
            print("\nIngresa solo digitos...")
    elif opc == '2':
        if tree.root == None:
            print("Vacio")
        else:
            tree.inorder(tree.root)
    elif opc == '3':
        if tree.root == None:
            print("Vacio")
        else:
            tree.preorder(tree.root)
    elif opc == '4':
        if tree.root == None:
            print("Vacio")
        else:
            tree.postorder(tree.root)
    elif opc == '5':
        nodo = input("\nIngresa el nodo a buscar -> ")
        if nodo.isdigit():
            nodo = int(nodo)
            if tree.buscar(nodo, tree.root) == None:
                print("\nNodo no encontrado...")
            else:
                print("\nNodo encontrado -
> ",tree.buscar(nodo, tree.root), " si existe...")
        else:
            print("\nIngresa solo digitos...")
    elif opc == '6':
        print("\nElegiste salir...\n")
        os.system("pause")
```

```
        break
    else:
        print("\nElige una opcion correcta...")
        print()
        os.system("pause")
print()
```

Resultados

<https://youtu.be/10491r2owYs>



BST.py - Visual Studio Code

```

77         else:
78             tree.inorder(tree.root)
79         elif opc == '3':
80             if tree.root == None:
81                 print("Vacío")
82             else:
83                 tree.preorder(tree.root)
84         elif opc == '4':
85             if tree.root == None:
86                 print("Vacío")
87             else:
88                 tree.postorder(tree.root)
89         elif opc == '5':
90             nodo = input("\nIngresar el nodo a buscar -> ")
91             if nodo.isdigit():
92                 nodo = int(nodo)
93                 if tree.buscar(nodo, tree.root) == None:
94                     print("\nNodo no encontrado...")
95                 else:
96                     print("\nNodo encontrado -> ", tree.buscar(nodo, tree.root), " si existe...")
97             else:
98                 print("\nIngresar solo dígitos...")
99         elif opc == '6':
100             print("\nEscriba salir...")
101             os.system("pause")
102             break
103         else:
104             print("\nElija una opción correcta...")
105         print()
106         os.system("pause")
107     """
    
```

Elige una opción -> 2

 1

 2

 3

 4

 5

 6

 7

 8

 9

 10

 11

 12

 13

 14

 15

 16

 17

 18

 19

 20

 21

 22

 23

 24

 25

 26

 27

 28

 29

 30

 31

 32

 33

 34

 35

 36

 37

 38

 39

 40

 41

 42

 43

 44

 45

 46

 47

 48

 49

 50

 51

 52

 53

 54

 55

 56

 57

 58

 59

 60

 61

 62

 63

 64

 65

 66

 67

 68

 69

 70

 71

 72

 73

 74

 75

 76

 77

 78

 79

 80

 81

 82

 83

 84

 85

 86

 87

 88

 89

 90

 91

 92

 93

 94

 95

 96

 97

 98

 99

 100

 101

 102

 103

 104

 105

 106

 107

 108

 109

 110

 111

 112

 113

 114

 115

 116

 117

 118

 119

 120

 121

 122

 123

 124

 125

 126

 127

 128

 129

 130

 131

 132

 133

 134

 135

 136

 137

 138

 139

 140

 141

 142

 143

 144

 145

 146

 147

 148

 149

 150

 151

 152

 153

 154

 155

 156

 157

 158

 159

 160

 161

 162

 163

 164

 165

 166

 167

 168

 169

 170

 171

 172

 173

 174

 175

 176

 177

 178

 179

 180

 181

 182

 183

 184

 185

 186

 187

 188

 189

 190

 191

 192

 193

 194

 195

 196

 197

 198

 199

 200

 201

 202

 203

 204

 205

 206

 207

 208

 209

 210

 211

 212

 213

 214

 215

 216

 217

 218

 219

 220

 221

 222

 223

 224

 225

 226

 227

 228

 229

 230

 231

 232

 233

 234

 235

 236

 237

 238

 239

 240

 241

 242

 243

 244

 245

 246

 247

 248

 249

 250

 251

 252

 253

 254

 255

 256

 257

 258

 259

 260

 261

 262

 263

 264

 265

 266

 267

 268

 269

 270

 271

 272

 273

 274

 275

 276

 277

 278

 279

 280

 281

 282

 283

 284

 285

 286

 287

 288

 289

 290

 291

 292

 293

 294

 295

 296

 297

 298

 299

 300

 301

 302

 303

 304

 305

 306

 307

 308

 309

 310

 311

 312

 313

 314

 315

 316

 317

 318

 319

 320

 321

 322

 323

 324

 325

 326

 327

 328

 329

 330

 331

 332

 333

 334

 335

 336

 337

 338

 339

 340

 341

 342

 343

 344

 345

 346

 347

 348

 349

 350

 351

 352

 353

 354

 355

 356

 357

 358

 359

 360

 361

 362

 363

 364

 365

 366

 367

 368

 369

 370

 371

 372

 373

 374

 375

 376

 377

 378

 379

 380

 381

 382

 383

 384

 385

 386

 387

 388

 389

 390

 391

 392

 393

 394

 395

 396

 397

 398

 399

 400

 401

 402

 403

 404

 405

 406

 407

 408

 409

 410

 411

 412

 413

 414

 415

 416

 417

 418

 419

 420

 421

 422

 423

 424

 425

 426

 427

 428

 429

 430

 431

 432

 433

 434

 435

 436

 437

 438

 439

 440

 441

 442

 443

 444

 445

 446

 447

 448

 449

 450

 451

 452

 453

 454

 455

 456

 457

 458

 459

 460

 461

 462

 463

 464

 465

 466

 467

 468

 469

 470

 471

 472

 473

 474

 475

 476

 477

 478

 479

 480

 481

 482

 483

 484

 485

 486

 487

 488

 489

 490

 491

 492

 493

 494

 495

 496

 497

 498

 499

 500

 501

 502

 503

 504

 505

 506

 507

 508

 509

 510

 511

 512

 513

 514

 515

 516

 517

 518

 519

 520

 521

 522

 523

 524

 525

 526

 527

 528

 529

 530

 531

 532

 533

 534

 535

 536

 537

 538

 539

 540

 541

 542

 543

 544

 545

 546

 547

 548

 549

 550

 551

 552

 553

 554

 555

 556

 557

 558

 559

 560

 561

 562

 563

 564

 565

 566

 567

 568

 569

 570

 571

 572

 573

 574

 575

 576

 577

 578

 579

 580

 581

 582

 583

 584

 585

 586

 587

 588

 589

 590

 591

 592

 593

 594

 595

 596

 597

 598

 599

 600

 601

 602

 603

 604

 605

 606

 607

 608

 609

 610

 611

 612

 613

 614

 615

 616

 617

 618

 619

 620

 621

 622

 623

 624

 625

 626

 627

 628

 629

 630

 631

 632

 633

 634

 635

 636

 637

 638

 639

 640

 641

 642

 643

 644

 645

 646

 647

 648

 649

 650

 651

 652

 653

 654

 655

 656

 657

 658

 659

 660

 661

 662

 663

 664

 665

 666

 667

 668

 669

 670

 671

 672

 673

 674

 675

 676

 677

 678

 679

 680

 681

 682

 683

 684

 685

 686

 687

 688

 689

 690

 691

 692

 693

 694

 695

 696

 697

 698

 699

 700

 701

 702

 703

 704

 705

 706

 707

 708

 709

 710

 711

 712

 713

 714

 715

 716

 717

 718

 719

 720

 721

 722

 723

 724

 725

 726

 727

 728

 729

 730

 731

 732

 733

 734

 735

 736

 737

 738

 739

 740

 741

 742

 743

 744

 745

 746

 747

 748

 749

 750

 751

 752

 753

 754

 755

 756

 757

 758

 759

 760

 761

 762

 763

 764

 765

 766

 767

 768

 769

 770

 771

 772

 773

 774

 775

 776

 777

 778

 779

 780

 781

 782

 783

 784

 785

 786

 787

 788

 789

 790

 791

 792

 793

 794

 795

 796

 797

 798

 799

 800

 801

 802

 803

 804

 805

 806

 807

 808

 809

 810

 811

 812

 813

 814

 815

 816

 817

 818

 819

 820

 821

 822

 823

 824

 825

 826

 827

 828

 829

 830

 831

 832

 833

 834

 835

 836

 837

 838

 839

 840

 841

 842

 843

 844

 845

 846

 847

 848

 849

 850

 851

 852

 853

 854

 855

 856

 857

 858

 859

 860

 861

 862

 863

 864

 865

 866

 867

 868

 869

 870

 871

 872

 873

 874

 875

 876

 877

 878

 879

 880

 881

 882

 883

 884

 885

 886

 887

 888

 889

 890

 891

 892

 893

 894

 895

 896

 897

 898

 899

 900

 901

 902

 903

 904

 905

 906

 907

 908

 909

 910

 911

 912

 913

 914

 915

 916

 917

 918

 919

 920

 921

 922

 923

 924

 925

 926

 927

 928

 929

 930

 931

 932

 933

 934

 935

 936

 937

 938

 939

 940

 941

 942

 943

 944

 945

 946

 947

 948

 949

 950

 951

 952

 953

 954

 955

 956

 957

 958

 959

 960

 961

 962

 963

 964

 965

 966

 967

 968

 969

 970

 971

 972

 973

 974

 975

 976

 977

 978

 979

 980

 981

 982

 983

 984

 985

 986

 987

 988

 989

 990

 991

 992

 993

 994

 995

 996

 997

 998

 999

 1000

Elige una opción -> 3

 1

 2

 3

 4

 5

 6

 7

 8

 9

 10

 11

 12

 13

 14

 15

 16

 17

 18

 19

 20

 21

 22

 23

 24

 25

 26

 27

 28

 29

 30

 31

 32

 33

 34

 35

 36

 37

 38

 39

 40

 41

 42

 43

 44

 45

 46

 47

 48

 49

 50

 51

 52

 53

 54

 55

 56

 57

 58

 59

 60

 61

 62

 63

 64

 65

 66

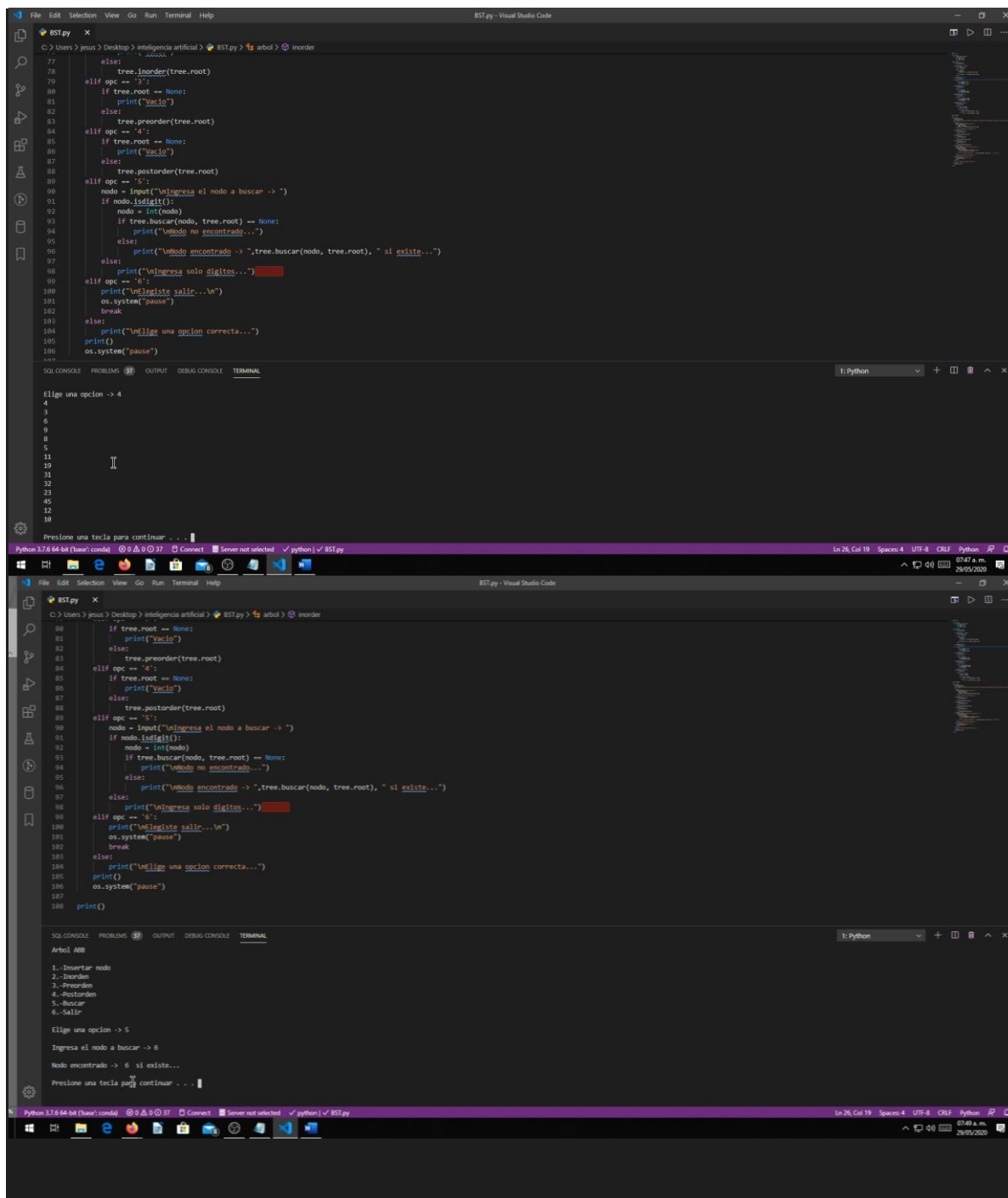
 67

 68

 69

 70

 71



The image shows two screenshots of a Visual Studio Code editor with a Python file named `BST.py`. The code implements a Binary Search Tree (BST) with methods for insertion, traversal, and searching.

Top Screenshot: The code is at line 100, where the user is prompted to choose an option. The terminal shows the user has entered '4'.

```
77:         else:
78:             tree.inorder(tree.root)
79:         elif opc == '3':
80:             if tree.root == None:
81:                 print("Vacío")
82:             else:
83:                 tree.preorder(tree.root)
84:         elif opc == '4':
85:             if tree.root == None:
86:                 print("Vacío")
87:             else:
88:                 tree.postorder(tree.root)
89:         elif opc == '5':
90:             nodo = input("Ingresa el nodo a buscar -> ")
91:             if nodo.isdigit():
92:                 nodo = int(nodo)
93:                 if tree.buscar(nodo, tree.root) == None:
94:                     print("Nodo no encontrado...")
95:                 else:
96:                     print("Nodo encontrado -> ", tree.buscar(nodo, tree.root), " si existe...")
97:             else:
98:                 print("Ingresa solo dígitos...")
99:         elif opc == '6':
100:            print("Ingresa salir...")
101:            os.system("pause")
102:            break
103:         else:
104:            print("Elige una opción correcta...")
105:            print()
106:            os.system("pause")
107:    except:
108:        pass
```

Bottom Screenshot: The code is at line 100, where the user is prompted to choose an option. The terminal shows the user has entered '5', then '6', and the program has found the node.

```
109:         if tree.root == None:
110:             print("Vacío")
111:         else:
112:             tree.preorder(tree.root)
113:         elif opc == '4':
114:             if tree.root == None:
115:                 print("Vacío")
116:             else:
117:                 tree.postorder(tree.root)
118:         elif opc == '5':
119:             nodo = input("Ingresa el nodo a buscar -> ")
120:             if nodo.isdigit():
121:                 nodo = int(nodo)
122:                 if tree.buscar(nodo, tree.root) == None:
123:                     print("Nodo no encontrado...")
124:                 else:
125:                     print("Nodo encontrado -> ", tree.buscar(nodo, tree.root), " si existe...")
126:             else:
127:                 print("Ingresa solo dígitos...")
128:         elif opc == '6':
129:            print("Ingresa salir...")
130:            os.system("pause")
131:            break
132:         else:
133:            print("Elige una opción correcta...")
134:            print()
135:            os.system("pause")
136:    except:
137:        pass
138:    print()
```

The terminal output for the bottom screenshot is as follows:

```
Arbol AB
1.-Insertar nodo
2.-Borrar
3.-Preorden
4.-Postorden
5.-Buscar
6.-Salir

Elige una opción -> 5

Ingresa el nodo a buscar -> 6
Nodo encontrado -> 6 si existe...

Presione una tecla para continuar . . .
```

Referencias

https://es.wikipedia.org/wiki/%C3%81rbol_binario_de_b%C3%BAsqueda

<https://riptutorial.com/es/algorithm/example/20877/arbol-de-busqueda-binario---insercion--python->

<https://gist.github.com/codigosdeprogra>