Laporan

Struktur Data  
Binary Search Tree

short line



Dosen Pengampu:

[Muchammad Chandra Cahyo Utomo, M.Kom.](https://if.itk.ac.id/profile/dosen/detail/muchammad-chandra-cahyo-utomo-mkom) 199205202019031013

Disusun Oleh :

|  |  |
| --- | --- |
| Guntur Wisnu Saputra | 11211042 |
| Muhammad Insan Kamil | 11211058 |
| Muhammad Ricky Zakaria | 11211062 |
| Ramadhan Djibran Sanjaya | 11211070 |
| Rangga Hermawan | 11211071 |
| Rendy Pernanda | 11211074 |
|  |  |

24 Oktober 2022

# *# Source Code*

|  |  |
| --- | --- |
| No. | BinarySearchTree.py |
| 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 | class Node:      def \_\_init\_\_(*self*, *value*):  *self*.\_\_left = None  *self*.\_\_right = None  *self*.\_\_value = *value*      def setLeft(*self*,*left*):  *self*.\_\_left = *left*      def setRight(*self*,*right*):  *self*.\_\_right = *right*      def setValue(*self*, *value*):  *self*.\_\_value = *value*      def getLeft(*self*):  *return* *self*.\_\_left      def getRight(*self*):  *return* *self*.\_\_right      def getValue(*self*):  *return* *self*.\_\_value  def insert(*root*, *value*):  *if* *root* *is* None:  *return* Node(*value*)  *else*:  *if* *root*.getValue() == *value*:  *return* *root*  *elif* *root*.getValue() < *value*:  *root*.setRight(insert(*root*.getRight(), *value*))  *else*:  *root*.setLeft(insert(*root*.getLeft(), *value*))  *return* *root*  def PrintTree(*root*):      def height(*root*):  *return* 1 + max(height(*root*.getLeft()), height(*root*.getRight())) *if* *root* *else* -1      nlevels = height(*root*)      width =  pow(2,nlevels+1)      q=[(*root*,0,width,'c')]      levels=[]  *while*(q):          node,level,x,align= q.pop(0)  *if* node:  *if* len(levels)<=level:                  levels.append([])                levels[level].append([node,level,x,align])              seg= width//(pow(2,level+1))              q.append((node.getLeft(),level+1,x-seg,'l'))              q.append((node.getRight(),level+1,x+seg,'r'))  *for* i,l *in* enumerate(levels):          pre=0          preline=0          linestr=''          pstr=''          seg= width//(pow(2,i+1))  *for* n *in* l:              valstr= str(n[0].getValue())  *if* n[3]=='r':                  linestr+=' '\*(n[2]-preline-1-seg-seg//2)+ '¯'\*(seg +seg//2)+'\\'                  preline = n[2]  *if* n[3]=='l':                 linestr+=' '\*(n[2]-preline-1)+'/' + '¯'\*(seg+seg//2)                 preline = n[2] + seg + seg//2              pstr+=' '\*(n[2]-pre-len(valstr))+valstr              pre = n[2]          print(linestr)          print(pstr)  def inorder(*root*):  *if* *root*:          inorder(*root*.getLeft())          print(*root*.getValue(), *end*=" ")          inorder(*root*.getRight())  r = Node(5)  print()  print("Binary Search Tree")  print("After call method insert(r,3):")  r = insert(r, 3)  PrintTree(r)  print("After call method insert(r,2);")  r = insert(r, 2)  PrintTree(r)  print("After call method insert(r,4):")  r = insert(r, 4)  PrintTree(r)  print("After call method insert(r,7):")  r = insert(r, 7)  PrintTree(r)  print("After call method insert(r,6);")  r = insert(r, 6)  PrintTree(r)  print("After call method insert(r,8);")  r = insert(r, 8)  PrintTree(r)  print("in-order")  inorder(r) |

# *#Hasil Run*

# **BinarySearchTree.py**

