Για την προσθηκη των μεθοδων δημιουργησα ενα αρχειο **btree_extend.py**, που λειτουργει σαν επεκταση του **btree.py**. Οι προσθηκες στα **Node** και **Btree** γινεται ως εξης:

```
92      def remove(self, value):
93          Extend.remove(self, value)
94
95
96
97  class Btree:
98
99      def merge(self, node_left, node_right, oldParValue = None):
100         Extend.merge(self, node_left, node_right, oldParValue)
101
102     def delete(self, value):
103         return Extend.delete(self, value)
104         
105     def setColumn(self, nodeValue, columnIndex, columnValue):
106         Extend.setColumn(self, nodeValue, columnIndex, columnValue)
107         
108     def removeColumn(self, nodeValue, columnName):
109         removeColumn(self, nodeValue, column)
```

Προσθεσα επισης το **indices**, για τα MultiIndex

```
14              self.indices = []
15          for i in range(0, len(values)):
16              self.indices.append(Extend.MultiIndex)
```

Το delete εντος του κομβου γινεται αφαιροντας την τιμη και τα indices, και αν δεν ειναι εσωτερικος κομβος, τη θεση στη βαση.

```
36  # Node
37
38  def remove(self, value):
39      for index, existing_val in enumerate(self.values):
40          if value == existing_val:
41              self.values.pop(index)
42              self.indices.pop(index)
43              
44              if self.is_leaf:
45                  self.ptrs.pop(index)
46              break
```

Για την διαγραφη εφτιαξα πολλαπλες συναρτησεις για να βοηθησουν, αλλα η βασικη που καλειται δια διαγραφες ειναι η **delete(self, value)**.

```
 73  def merge(self, node_left, node_right, oldParValue = None):
117
118  def delete(self, value):
290
291  def shiftValueToNode(self, value, oldNode, newNode):
295
296  def removeEmpty(self, value):
311          #self.nodes.pop(i)
312
313  def mergeEmptyParentsSingleNode(self, node):
328
329  def mergeEmptyParents(self, value):
343
344  def replaceParentNodeValue(self, oldVal, newVal):
361
362  def deleteParentNodeValue(self, oldVal):
369
370      #if node.parent == None:
371      #    return
372
373      #if node.parent and len(self.nodes[node.parent].values) == 0:
374      #    sibs = getNodeSiblings(self, node, oldVal)
375      #    if sibs != None:
376      #        print("Merge within merge")
377      #        self.merge(node, sibs[0])
378      #    else:
379      #        print("No merge inside merge")
380
381  def getNodeSiblings(self, node, value):
410
411  def removeNodeFromParent(self, node):
```

Η delete χρησιμοποιει τα 3 cases και τις περιπτωσεις, τροποποιημενα με βαση το link που σταλθηκε μετα απο την πρωτη προσπαθεια (https://www.programiz.com/dsa/deletion-from-a-b-plus-tree)

Για τα **MultiIndex** δημιουργησα τις συναρτησεις **setColumn** και **removeColumn**.

```
51  def setColumn(self, nodeValue, columnName, columnValue, columnType):
52      index = self._search(nodeValue)
53      
54      if index != None:
55          node = self.nodes[index]
56          index = node.values.index(nodeValue)
57          
58          if (index != None):
59              node.indices[index].setMultiIndex(node.indices[index], columnName, columnValue, columnType)
60              print(node.indices[index].indices[-1].name + ", " + str(node.indices[index].indices[-1].value))
61  
62  def removeColumn(self, nodeValue, columnName):
63      index = self._search(nodeValue)
64      
65      if index != None:
66          node = self.nodes[index]
67          index = node.values.index(nodeValue)
68          
69          if (index != None):
70              node.indices[index].removeMultiIndex(node.indices[index], columnName)
71              print(node.indices[index].indices[-1].name + ", " + str(node.indices[index].indices[-1].value))
```

Για την εκτελεση, προσθεσα ενα loop εντος του **btree_test.py** για να κανει 5 deletions.

```
46  bt.plot("Pre")
47  i = 0
48  for q in range(0, 5):
49      n = randrange(NUM)
50      
51      if bt.delete(n) == True:
52          i = i + 1
53          bt.plot("/Step " + str
54          print("Step " + str(i))
55          sleep(0.2)
56          print("--------")
```

Προσθεσα επισης μια δευτερη εκδοση του plot (που ξεχασα να βαλω στο **btree_extend.py**) που προσθετει ενα prefix στο ονομα του plot, ωστε το loop να μπορει να δειξει το καθε βημα ξεχωριστα.

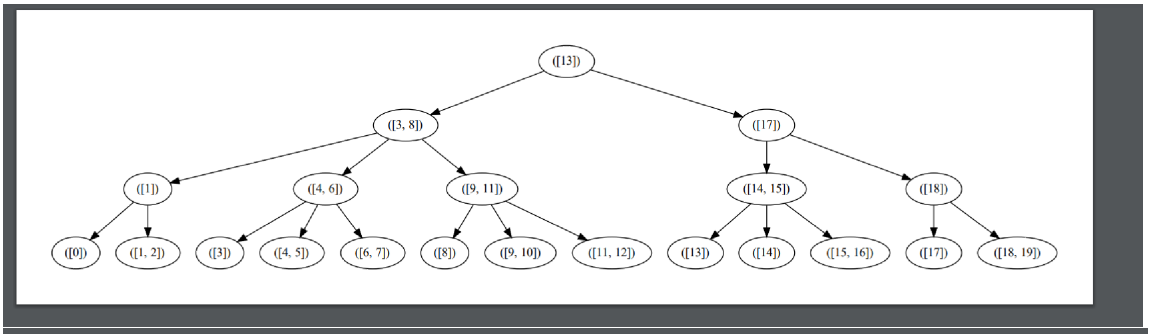
```
256     def plot(self):
293     
294     def plot(self, suffix = ""):
```

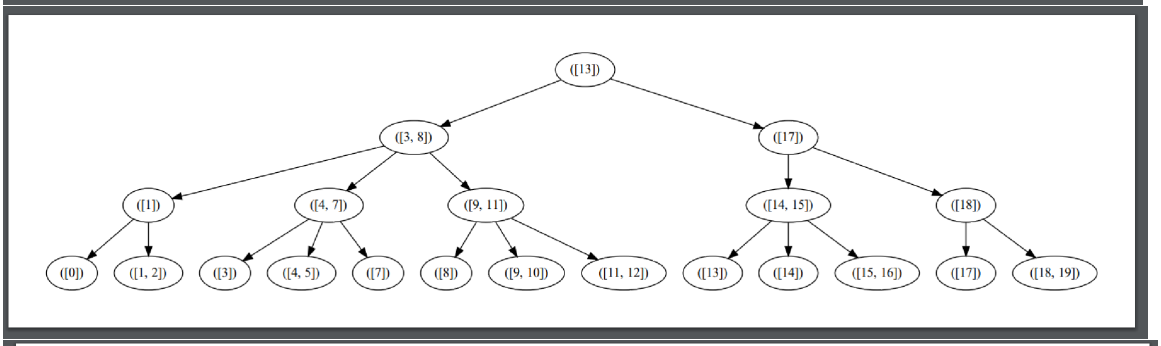
Εκτελεση

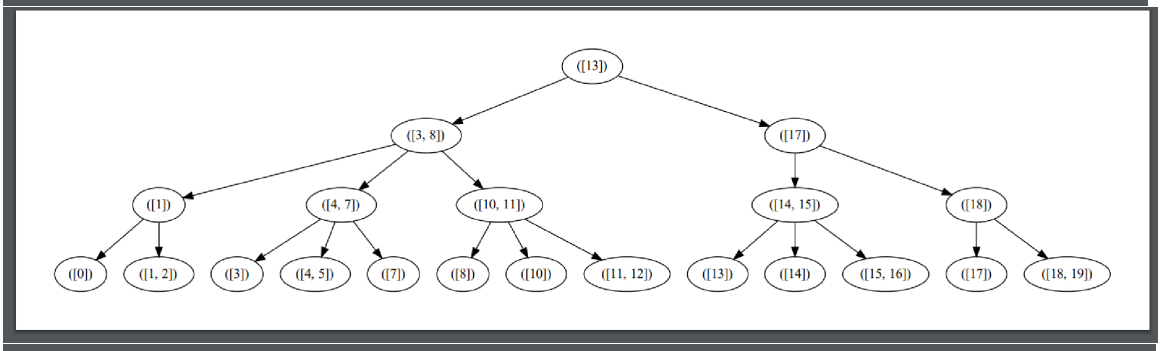
```
\Fork\miniDB> python .\btree_test.py 20 3
```

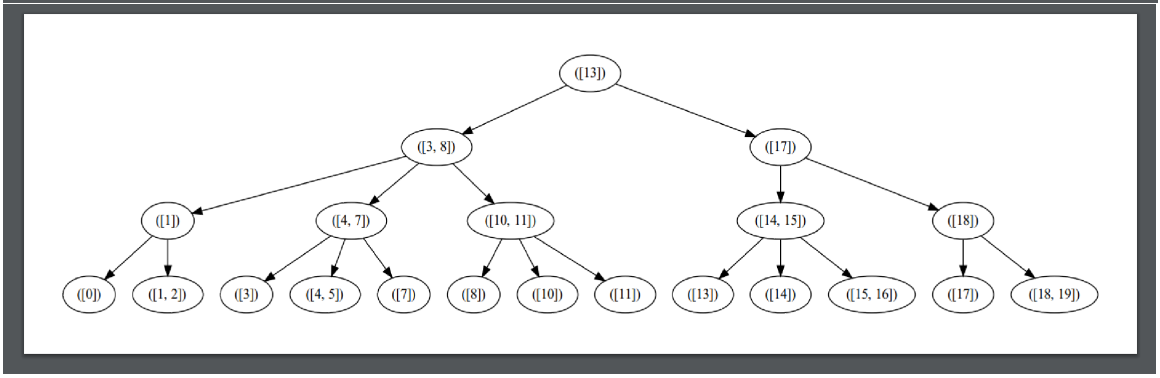
(Το load προσθετει την ιδια σειρα αριθμων οπως η τελευταια εκτελεση)

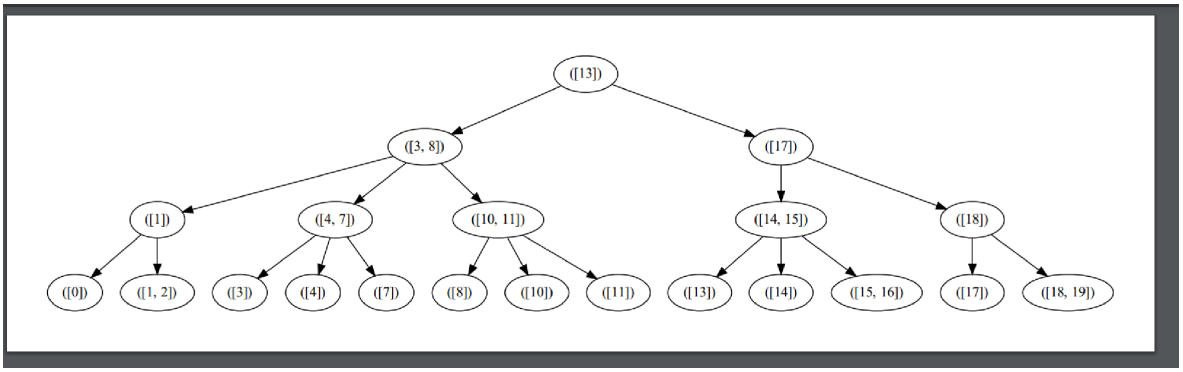
```
python .\btree_test.py 20 3 load
```










([13])
([3, 8])
([17])
([1])
([4, 7])
([10, 11])
([14, 15])
([18])
([0])
([1, 2])
([3])
([4])
([7])
([8])
([10])
([11])
([13])
([14])
([15, 16])
([17])
([18, 19])