**def** binary\_Search(seq, low, high):

firstSeqPlace = 0

seqLen = len(seq)

lastSeqPlace = seqLen - 1

**if** low >= high:

**raise** ValueError ("The 'low' value can not be bigger then 'high' value")

**while** firstSeqPlace <= lastSeqPlace **and** **True**:

middleValue = seqLen // 2

**if** seq[middleValue] **in** range (low, high+1):

**return** (**True**)

**else**:

**if** seq[middleValue] < low:

**return** (binary\_Search(seq[middleValue+1:], low, high))

**else**:

**return** (binary\_Search(seq[:middleValue], low, high))

seq = [2,3,5,7,9,13]

low = 10

high = 14

biSearch = binary\_Search(seq, low, high)

print (biSearch)

This is a binary search that has an algorithm of divide and conquer. This type of algorithms have got Big O notation of O(log n). That’s because as a size of a problem grows, the number of division also grows, but slowly.