Complexitatea

Daca exista caz favorabil/defavorabil:

* + Descrie **caz favorabil**
  + Calculeaza **best case**
  + Descrie **caz defavorabil**
  + Calculeaza **worst case**
  + Calculeaza **complexitatea medie**
  + Calculeaza **complexitatea generala**

CAUTARE BINARA – recursive - 𝑂(log2𝑛)

def **binaryS**(el, l, left, right):

"""

Search an element in a list

el - element to be searched;

l - a list of ordered elements left,right the sublist in which we search return the position of first occurrence or the insert position

"""

if left>=right-1:

return right

m = (left+right)/2

if el<=l[m]:

return binaryS(el, l, left, m)

else:

return binaryS(el, l, m, right)

def **searchBinaryRe**c(el, l):

*"""*

*Search an element in a list*

*el - element to be searched*

*l - a list of ordered elements return the position of first occurrence or the insert position*

*"""*

if len(l)==0:

return 0

if el>l[len(l)-1]:

return len(l)

return binaryS(el, l, 0, len(l))