**RUN – TIME ANALYSIS**

(Worst-Case Big-O Notation)

Student: **Saidov Jumanazar**

Date: **15.05.2021**

Solution 1. Task 0 - The script correctly prints out the information of first record of texts and last record of calls.

def getTextRecord(index): # O(1)  
 msg = f"First record of texts, {texts[index][0]} texts {texts[index][1]} at time {texts[index][-1]}"  
 return msg  
  
  
def getCallRecord(index): # O(1)  
 msg = f"Last record of calls, {calls[index][0]} calls {calls[index][1]} at time {calls[index][2]}, lasting {calls[index][-1]} seconds"  
 return msg  
  
  
print(getTextRecord(0))  
print(getCallRecord(-1))

O(1+1 ) = O(2) => O(1)

Solution 2. Task 1 - The script correctly prints number of distinct telephone numbers in the dataset.

def allDifferentRecords(): # O(n + 2)  
 result = []  
 for row in calls:  
 if row[0] not in result:  
 result.append(row[0])  
 if row[1] not in result:  
 result.append(row[1])   
 return len(result)  
  
  
num = allDifferentRecords()  
print(f"There are {num} different telephone numbers in the records.")

O(n +2) => O(n)

Solution 3. Task 2 - The script correctly prints the telephone number that spent the longest time on the phone and the total time in seconds they spend on phone call.

def getLongestCall(): # O(2n + 3)  
 differentCalls = {}  
 for index in range(1, len(calls)): #O(n + 2)  
 callerNumber = calls[index][0]  
 receivingNumber = calls[index][1]  
 callDuration = int(calls[0][-1])  
  
 if callerNumber in differentCalls:  
 differentCalls[callerNumber] += callDuration  
 else:  
 differentCalls[callerNumber] = callDuration  
 if receivingNumber in differentCalls:  
 differentCalls[receivingNumber] += callDuration  
 else:  
 differentCalls[receivingNumber] = callDuration  
 maxCall = 0  
 num = 0  
  
 for item in differentCalls.items(): #O(n +1)  
 if item[1] > maxCall:  
 maxCall = item[1]  
 num = item[0]  
 print(f"{num} spent the longest time, {maxCall} seconds, on the phone during September 2016.")  
  
  
getLongestCall()

O(2n + 3) => O(2n) => O(n)

Solution 4. Task 3 - The script correctly prints the telephone codes called by fixed-line numbers in Bangalore and the percentage of calls from fixed lines in Bangalore that are to fixed lines in Bangalore.

def printAllPrefixes(): # O(2n +1)  
 print("The numbers called by people in Bangalore have codes:")  
 callsFromBangalore = []  
 calls\_bangalore2bangalore = []  
 bangaloreReceivedCalls\_prefixes = []  
 for call in calls: # O(n + 1)  
 if call[0].startswith("(080)"):  
 callsFromBangalore.append(call)  
 if (call[1].startswith('7') or call[1].startswith('8') or call[1].startswith('9')) and " " in call[1]:  
 prefix = call[1][:4]  
 if prefix not in bangaloreReceivedCalls\_prefixes:  
 bangaloreReceivedCalls\_prefixes.append(prefix)  
 elif call[1].startswith("("):  
 st = str(call[1]).find('(') + 1  
 en = call[1].find(')')  
 prefix = call[1][st:en]  
 if prefix not in bangaloreReceivedCalls\_prefixes:  
 bangaloreReceivedCalls\_prefixes.append(prefix)  
 elif call[1].startswith("140"):  
 prefix = call[1][:3]  
 bangaloreReceivedCalls\_prefixes.append(prefix)  
 if call[1].startswith("(080)"):  
 calls\_bangalore2bangalore.append(call[1])  
 bangaloreReceivedCalls\_prefixes.sort()  
 for code in bangaloreReceivedCalls\_prefixes: # O(n)  
 print(code)  
 received\_len = len(calls\_bangalore2bangalore)  
 allCalls\_len = len(callsFromBangalore)  
 percentage = (received\_len \* 100) / allCalls\_len  
  
 print(  
 f"{percentage:.2f} percent of calls from fixed lines in Bangalore are calls to other fixed lines in Bangalore.")  
  
  
printAllPrefixes()

O(2n+1) => O(2n) => O(n)

Solution 5. Task 4 - The script correctly prints the list of numbers that could be telemarketers.

def printTelemarketerNumbers(): # O(2n + 1)  
 print("These numbers could be telemarketers: ")  
 numbers = []  
 for call in calls: # O(n +1)  
 if call[0].startswith('140') and call[0] not in numbers:  
 numbers.append(call[0])  
 numbers.sort()  
 for num in numbers: #O(n)  
 print(num)  
  
  
printTelemarketerNumbers()

O(2n + 1) => O(2n) => O(n)