Data Structures and Algorithms.

Unit 3 - Analysis of Algorithms.

Problems:

Part A - Empirical Analysis

1. Write a function **randomList**(n,a,b) that returns a list of n random integers between a and b. The function can consider that the default values for a and b are 0 and 25, respectively. This functions will be used later.

2. Write a function, called **sumList**(l), which takes a list of integers as parameter and returns the sum of all its elements. You must empirically study the time complexity of this function. To do this, you should include instructions to measure the running time. Then, you should run it for different size of lists (such as 10,100,1000,10000,10000,etc). Finally, plot its time complexity on list of different sizes. What does the running time depend on?

Notes:

- You should use the randomList function to generate lists of different sizes.
- When you plot the results, you should choose the logarithmic scale for the axes.

3. Write a function, named **search**, which takes a Python list of integers and a number as parameters, and returns true if the element exists and false otherwise.

4. Now suppose that the input list is always sorted in ascending order. Write a new version of the search function, named **binary_search**, which takes a sorted list and a number as parameters, and checks if the number exists into the list. In this function, you should take advantage of the list is sorted.

5. Study (empirically) and compare the time complexity of the two previous functions (assume that the lists are sorted). To simplify, you can assume that the number to search is always 5. You can generate the lists with the function randomList and sort them using the Python function of lists, sort (for example, l.sort()). What function is more efficient?

Part B - Theoretical Analysis

6. Calculate the running time function T(n) for the sumList function.

7. Write a function, called ***sumPair0***, which takes a Python list of integers, *data*, as a parameter and returns the number of pairs (i,j) such as *i!=j and data[i]+data[j]=0.* Calculate the running time function T(n).

8. Write a function, called ***sumTriple0***, that takes a Python list of integers, data, as a parameter and returns the number of triples (i,j,k) such as *i!=j and i!=k and j!=k and* data[i]+data[j]]+data[k]=0. Calculate the running time function T(n).