20CS 4033 & 6033 AI – I

Fall 2023

Instructor: Anca Ralescu

**Homework Assignment #1**

**Assigned on August 28, 2023**

**Due on sept 8, 2023**

**11:59PM on Canvas**

**50 points**

This assignment is a kind of warm-up exercise, which will be useful later in the course.

It can be programmed in python or Matlab. **In either case, students must implement all the algorithms, that is they are not allowed to use any code lifted from online or from some library.**

At the top of the submitted program, in a comment section, the names of ALL students in the group must be written.

PART 1:

In this assignment students are asked to implement a hybrid sort algorithm which is a hybrid of a *recursive algorithm* such as **mergeSort** **(1**) or **quickSort** **(2)** and a iterative algorithm such a **bubbleSort (3)**. Let us call **hybridSort** the function for this new algorithm.

**hybridSort** takes the following arguments:

1. L: the list to be sorted
2. BIG: the Id or the Name of a sort algorithm to be used for large lists (e.g., **mergeSort**, or **quicksort**) or the number (1, or 2)
3. SMALL: the name or the Id of a sort algorithm for small lists (e.g., **bubbleSort**)
4. T: a threshold on the number of elements in the list

**hybridSort** outputs the sorted list.

What must be turned in is a well-documented program file in which you have defined each of the algorithms:

1. **Bubblesort**
2. **quickSort**
3. **mergeSort**
4. **hybridSort**
5. **several runs of each of these algorithms for lists different lengths**
6. **At the end of the program file, in a comment section, examples of runs, and the results obtained along with an analysis comparing the behavior of these algorithms (for example, number of pairwise element comparisons, or execution time).**

**Note:** It is very important is to notice that for large lists, that is, with more than T elements, **hybridSort** behaves like **mergeSort** or **quicksort** but DOES NOT CALL/INVOKE these.

Example: Suppose that the list to be sorted is

L = [5 1 9 3 67 90 2]

T = 5

and we call **hybridSort(L, 1, 3, 5**), that is if the list size is greater than 5 then behave like **mergeSort**, otherwise, call **bubbleSort**.

In this example, the list size is 7. This means that according to the first step of **mergeSort,** we need to divide it into two (approximately) equal halves.

Suppose the result is

L1 = [5 1 9 3]

L2 = [67 90 2]

(it could also be L1= [5 1 9] and L2 = [3 67 90 2])

Now we call **hybridSort** on each of these sublists. Since each of these lists has less than T=5 elements, **hybridSort** CALLS **bubblesort**

which returns

[1 3 5 9] and [2 67 90]

Now **hybridSort** returns to its **mergeSort** behavior and merges these lists into the result

[1 2 3 5 9 67 90]

PART II: View sorting as a task to be performed by an agent. What kind of agent architecture would be appropriate for each of the sorting algorithms considered here? In each case discuss the PEAS descriptions.