College of Engineering and Applied Science
Department of Electrical Engineering and Computer Science

20CS 4033 & 6033 AI – I
Fall 2023
Instructor: Anca Ralescu

Homework Assignment #1- PART II
Assigned on August 28, 2023
Due on sept 8, 2023
11:59PM on Canvas

Algorithm	Agent Arch. Type	Performance Measure	Environment	Actuators	Sensors/items to be sensed
BubbleSort	Reflex	Memory usage, time efficiency, accuracy	Unsorted list	Comparisons, element swaps	Element/list reading, index position
MergeSort	Goal-Oriented	Memory usage, time efficiency, accuracy	Unsorted list, temporary sublist	Splits lists into sublists, comparisons, merges	Element/list reading, index position, recursion depth
QuickSort	Goal-Oriented	Memory usage, time efficiency, accuracy	Unsorted list, temporary sublist	Selects pivots, partitions lists, comparisons, element swaps, merges	Element/list reading, index position, recursion depth
HybridSort	Reflex with state	Memory usage, time efficiency, accuracy, adaptability	Unsorted list, sublist, other algorithm env.	Selects algorithm behavior, calls algorithms, adjusts threshold, recursion	Element/list reading, list size, index position, inputs(BIG/SMALL/ID)

Table 1: Algorithm-specific PEAS description

1.1. BubbleSort

BubbleSort is best described as possessing a simple reflex agent architecture due to its straightforward and non-complex design. A reflex architecture is inherently reactive to input with no consideration of future or past states, making them basic and minimal in nature. In the BubbleSort algorithm each element of the input list is compared to its adjacent element with the potential for a swap to occur. One could argue that the adjacent element is a future state, however, it is important to note that BubbleSort's decision-making process is based off immediate comparison and doesn't factor in future or past considerations. Going further, in the PEAS model for BubbleSort we see that the algorithm is reactive in nature. The environment being an unsorted, unpredictable, list and the actuators of swapping and comparing are based off the current perception of the input, supporting that BubbleSort aligns with a simplistic reflex agent architecture.

1.2. MergeSort

Out of the agent architectures, MergeSort loosely exhibits the characteristics of a goal-oriented agent. In goal-oriented architectures, the framework is capable of setting/pursuing goals to achieve a desired

outcome. MergeSort strategically divides its sorting task into smaller subtasks, sorting each one individually through planned recursive calls. With recursion, MergeSort can effectively "decide" how to sort with the best efficiency to reach the goal of progressing towards a sorted list. In the PEAS model of the algorithm, there are a few signs that point towards a reactive nature; for example, the environment of an unsorted list is unpredictable and reflexive. MergeSort shies away from reactive architecture in other areas of the PEAS model, such as the 'sensor' category where MergeSort must sense and recall past recursive sublists in its memory along with the present index of the algorithm to properly coordinate its actuators. The ability to systematically work until a desired outcome is reached is apt to the goal-oriented architecture.

1.3. QuickSort

Like the MergeSort algorithm, QuickSort is another algorithm which exhibits properties of a goal-oriented agent architecture. As described above, goal-oriented architectures are capable of setting and pursuing goals to achieve a desired outcome, focusing on optimizing the pursuit of said goal. QuickSort uses a "divide and conquer" framework to break down its task into small subtasks using recursion (Sensor and Actuator in PEAS model), addressing each separately until the sorting objective is met. Each subtask created is a new 'goal' for QuickSort to pursue. QuickSort additionally shows adaptability in its decision-making process due to the usage of pivot points in its current states. This optimizes the sorting process to minimize the swaps needed to reach each objective the QuickSort algorithm faces. Additionally, in the PEAS model, the environment of QuickSort becomes more and more predictable as it creates temporary sublists (subgoals). These characteristics support QuickSort as a base goal-oriented architecture.

1.4. HybridSort

In HybridSort, all three algorithms mentioned above are apparent in sorting based on the size of the input to be sorted. Simply put, the behavior of HybridSort changes as the number of elements, T, changes. As HybridSort is a mixture of different algorithms, it is fair to say that HybridSort is a mixture of different agent architectures as well; namely, the reflex-agent with state and goal-oriented architectures. Like a reflex-agent with state architecture, HybridSort makes decisions in an unpredictable environment (PEAS: unsorted list) using its current perception to contextually choose which sorting algorithm should be used. For example, BubbleSort is called when there are less than 5 elements to be sorted. The characteristic of using current perception to optimize goal-reaching also points towards HybridSort having goal-oriented features. From the PEAS model, HybridSort 'actuates' both recursion and threshold adjustment to break down its initial task into reachable subtasks (subgoals as mentioned above), which are actively worked towards along with the overall sorting goal. HybridSort effectively combines the attributes of both goal-oriented and reflex-agent with state architectures, bridging the gap between the approaches.