CS Artificial Intelligence – Assignment 2 – Hybrid Sort Prolog – Group 53

Joe Ferguson, Brad Greene

Program Notes:

While completed and operable, bubbleSort fails to sort lists beyond lengths of 20 on seemingly random occasion. This might be because of a few factors: infinite backtracking, deadlock/stalling of prolog execution after any number of seconds, or an unforeseen factor we were not aware of. Several changes were implemented to try to handle this, none worked and because the starting code (with the "fillinheres" completed) was the code assigned, these changes were discarded.

So, in regards to testing bubbleSort, it was commented out on tests with unmanageable sizes for it (sizes 100+). To comment on its performance, it is by far the slowest algorithm both in theory and application.

Because of this, both versions of hybridSort that utilized bubbleSort had to either be commented out, or the thresholds changed to accommodate bubbleSort's failing with sizes beyond 100.

For the rest of the algorithms, there were no problems. Insertion sort and quick sort were exceedingly fast (for prolog execution times).

Another note to be taken into account is that when hybridSort starts to fail, for large lists where the iterative sorts fail), the time it would take had we lowered the threshold size would near the time of the recursive sort themselves. For example, on a list of size 10000, insertionSort fails, and if we continuously lower the threshold to manageable levels for insertionSort, it is called less and the recursive sort is called more, making it closer to the time of the recursive sort.

The results and execution are listed below.

Execution.

Assuming you run swipl on the prolog file and are in the same directory,

To generate and store the random lists: listsGeneration(X). X being the number of lists(in this case 50)

To run the tests, see the times for each run, and store the times: run_prog.

To see the average time of an algorithm for these runs: average_time(algorithm, X). Algorithm being the name of the algorithm you used.

Results.

Results on 50 Lists sizes 10-100 and threshold 40:

bubbleSort: Fails, stalls indefinitely

insertionSort: 0.000625 seconds

mergeSort: 0.0 seconds

quickSort: 0.0 seconds

hybridSort1: 0.0 seconds

hybridSort2: Fails, uses bubbleSort

hybridSort3: 0.0 seconds

hybridSort4: Fails, uses bubbleSort

Results on 50 Lists sizes 100-1000 and threshold 400:

bubbleSort: Fails, stalls indefinitely,

insertionSort: 3.94 seconds

mergeSort: 0.0 seconds

quickSort: 0.0 seconds

hybridSort1: 0.083 seconds

hybridSort2: Fails, uses bubbleSort

hybridSort3: 0.086 seconds

hybridSort4: Fails, uses bubbleSort

Results on 50 Lists sizes 1000-10000 and threshold 4000:

bubbleSort: Untested, stalls

insertionSort: Untested, stalls

mergeSort: 0.000625 seconds

quickSort: 0.0 seconds

hybridSort1: Fails, uses bubbleSort

hybridSort2: Fails, uses insertionSort

hybridSort3: Fails, uses bubbleSort

hybridSort4: Fails, uses insertionSort

Results on 50 Lists sizes 10000-100000 and threshold 4000:

bubbleSort: Untested, stalls

insertionSort: Untested, stalls

mergeSort: 0.39 seconds

quickSort: fails(unknown reason).

hybridSort1: Fails, uses bubbleSort

hybridSort2: Fails, uses insertionSort

hybridSort3: Fails, uses bubbleSort

hybridSort4: Fails, uses insertionSort

Discussion.

All algorithms acted and performed the way they would in theory, with the exception of bubbleSort completely failing randomly, along with quicksort with larger lists. This failure, it must be said, is not to be blamed on the code as it is correct and can be proven. One can even test the algorithms on a list like [1,5,2,9,8,3,2,2,0], and it will be sorted as it should. However, due to some unforeseen reason, maybe a characteristic of prolog, they will stall/deadlock completely.

This is beyond the terms of this assignment however, so we collected the results as usual anyway.