A priority queue for a simplified agenda manager in a rule-based expert system shell

Abhishek Kumar

R#11416761

CS 5381 Analysis of Algorithms, Fall 2017

Table of Contents

1. Objective

To create an application of priority queue for a simplified agenda manager in a rule-based expert system shell by using the concepts of heap. The agenda manager application should take an input file from the user and extract the highest priority rule after each cycle is executed.

2. Background

The two basic elements of a rule-based expert system shell are a knowledge base and an inference engine. The knowledge base contains the domain knowledge in the form of rules. The inference engine uses these rules to produce a solution to a given problem. The inference engine operates in cycles. In each cycle, it first determines which rules can be activated for firing (called activated rules), then selects one rule for execution and finally executes the selected rule.

Each activated rule has a corresponding priority. An agenda contains a list of activated rules, with their priorities, maintained by an agenda manager. In each cycle, an agenda manager updates list of activated rules by adding new activated rules and deleting rules that are no longer activated and the rule that was executed in a previous cycle.

What is a priority queue?

In computer science, a priority queue is an abstract data type which is like a regular queue or stack data structure, but where additionally each element has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority. If two elements have the same priority, they are served according to their order in the queue.

3. Program Description

- ❖ The program gets its input file from the user. Once the user provides the source path, the file is obtained & starts processing each line of the input file.
- ❖ After the whole line is read, a heap will be built using buildHeap() function & heapify().
- ❖ After building the heap, the higher priority rule will be found out & displayed as output through extractMax () method.
- ❖ The displayed element will be removed from the heap using deleteMax().
- ❖ Similarly rules present in further lines of input file will be added into the heap built.
- ❖ Till 30 cycles, the rules will be displayed as per the highest priority or until all the rules are exhausted.

4. Instructions to execute the program

Step 1: Save the input files (test1.txt, test2.txt, test3.txt) in the same location as the source code (if not available).

Step 2. Enter the source of the input file in the given AgendaManager code.

Step 3: Run the program using any IDE (Eclipse).

Step 4: The results for the given choice will be printed in the console.

Test Results

5.1 Input file: test1.txt

Given below is the GitHub link of the input file – test1.txt that is being executed to call the priority queue for a simplified Agenda Manager.

https://github.com/abhishek-kumar-code/AgendaManager/blob/master/src/test1.txt

The results for the input file – test1.txt can be found in the following GitHub link.

https://github.com/abhishek-kumar-code/AgendaManager/blob/master/Results/Output1.txt

5.2 Input file: test2.txt

Given below is the GitHub link of the input file – test2.txt that is being executed to call the priority queue for a simplified Agenda Manager.

https://github.com/abhishek-kumar-code/AgendaManager/blob/master/src/test2.txt

The results for the input file – test2.txt can be found in the following GitHub link.

https://github.com/abhishek-kumar-code/AgendaManager/blob/master/Results/Output2.txt

5.2 Input file: test3.txt

Given below is the GitHub link of the input file – test3.txt that is being executed to call the priority queue for a simplified Agenda Manager.

https://github.com/abhishek-kumar-code/AgendaManager/blob/master/src/test3.txt

The results for the input file – test3.txt can be found in the following GitHub link.

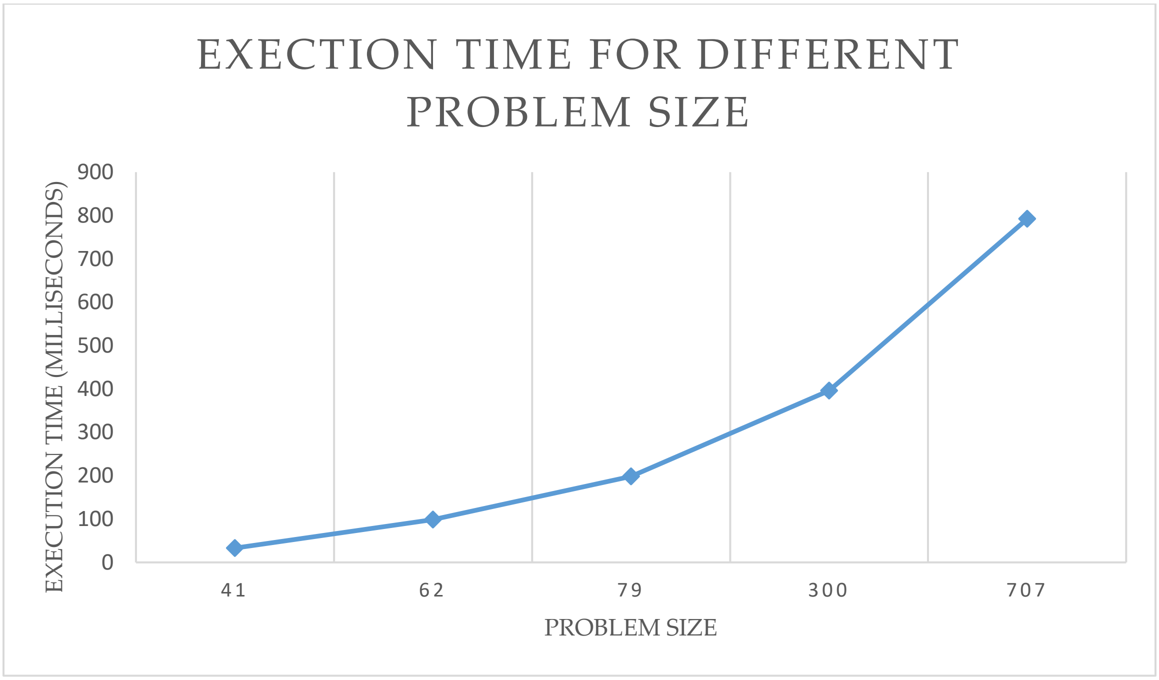
https://github.com/abhishek-kumar-code/AgendaManager/blob/master/Results/Output3.txt

6. Performance Evaluation

6.1 Experimental results and Graphical representation

The execution time of the program for different problem sizes is given below

Problem Size	Time Elapsed (in milliseconds)
33	41
66	62
198	79
396	300
792	707



The worst-case time complexity of the priority queue for a simplified agenda manager in a rule-based expert system shell is **O(nlgn)**.