Final Project:

Problem 12 - DFA Minimization

Justin Blechel & Zach Gassner

Department of Computer Science, Sonoma State University

CS 454: Theory of Computation

Dr. Ravikumar

May 20, 2023

Problem Statement:

Implement an algorithm to minimize a DFA. The standard algorithm that runs in $O(m\ n^2)$ time can be improved. One such faster algorithm is provided in an earlier version of our textbook.

Implementation Analysis:

Our approach was to use the algorithm that partitions states to different sets of states that are equivalent. The main idea is that states are first partitioned into two sets: accepting states and non-accepting states. Within each partition, each states' transitions are compared with each other, and if a transition leads to different partitions then that set is further partitioned. This process is continued until there can be no further partitions.

We started by making a DFA class in C++ to store the states, the alphabet, the accepting states, as well as the transitions. We then added a method to the DFA class that removes unreachable states from the DFA if applicable. Then we added a method that implements the above DFA minimization algorithm. The minimize method calls a boolean helper function, isEquivalent, that returns true if two states in the same partition are equivalent or not. Other than that we have a couple methods to output the minimized DFA to a text file and to standard output.

The minimize function has a time complexity of $O(states^2)$ and $\Omega(states)$. The time complexity is dominated by the nested loop that achieves best case if the set of states are partitioned into just the two initial partitions: accepting states and non-accepting states. The best case time complexity would mean that there are only two states in the minimized DFA. As the sets get partitioned further, this results in more than two states and brings the time complexity up to its worst case of $O(states^2)$. Overall this is an improvement over the standard algorithm that has $O(m\ n^2)$.

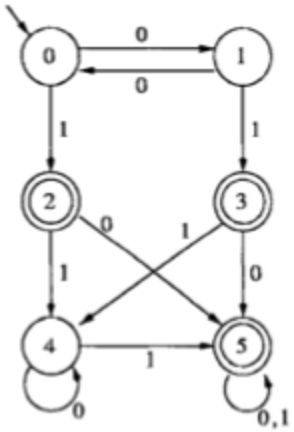
How to run our program:

Our program assumes the DFA comes in a text file in which the first line contains the number of states, the second line contains the number of input symbols, the third line contains the set of accepting states, and all remaining lines contain the transitions. This program assumes the states are numbered from 0 to number of states - 1, and 0 is always the starting start. This program also assumes the input symbols are 0 through number of input symbols - 1.

To run our program, use the make command to get the executable file DFA.x. Run the executable file with command line arguments of the name(s) of the file(s) that the DFA's are in. You can minimize one or many DFA's at once. For example, to run our program with only one DFA to be minimized, type './DFA.x dfa.txt'. If you want to minimize three DFA's for example, run './DFA.x dfa1.txt dfa2.txt dfa3.txt'. The output will create a new text file that outputs the DFA in the same format as the input DFA file. If the name of the text file that is being minimized is 'dfa3.txt', then the minimized DFA will be stored in 'dfa3-minimized.txt'.

Walkthrough:

As an example, we are going to illustrate how the DFA from problem 3 from the second midterm is minimized. The original DFA is shown below:



This DFA can be represented in a text file named ‘input2.txt’, which is shown below:

6

2

2 3 5

1 2

0 3

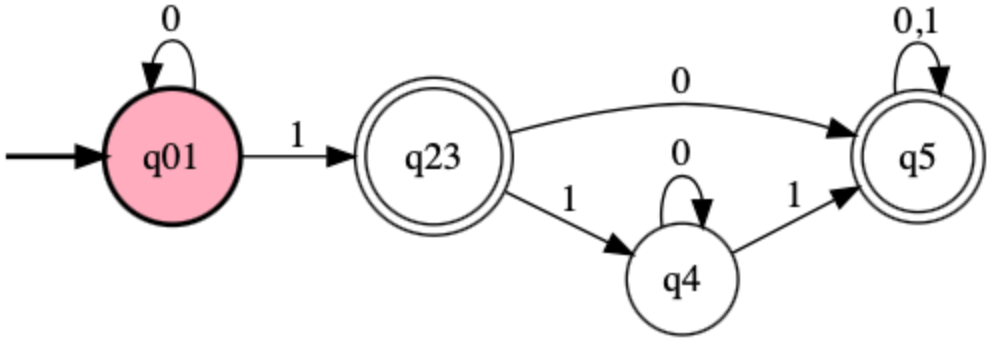
5 4

5 4

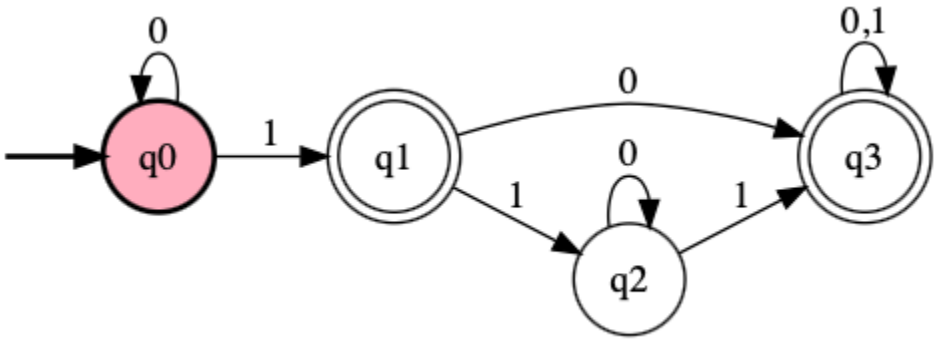
4 5

5 5

Minimizing this DFA yields the following:



It would not be possible to store this DFA following the above formatting guidelines since the first line in the text file would be 4, meaning the states are 0, 1, 2, and 3. So, after renaming the states we get the following DFA:



This DFA would be represented in ‘input2-minimzed.txt’ as:

4

2

1 3

0 1

3 2

2 3

3 3