Middle East Technical University | Department of Computer Engineering

CENG 315

ALGORITHMS

Fall 2016

Take Home

Due date: 27th December 2016, Tuesday, 23:55

1 Problem Definition

In this assignment, you will implement the algorithm developed by Hopcroft and Karp in 1971 that checks the equality of two finite automaton. The algorithm checks for equivalence in (very nearly) linear time without minimization of the automatons that being checked. Algorithm uses numerous set merge operations and in this assignment you will implement this operations with weighted quick union with path compression.

The algorithm is described in the pseudo-code 1. The quick-union functions are also given for reference and details can be found in the book *Introduction to Algorithms* by Cormen et al.

The flow of the algorithm and the states of the data structures for the DFAs given in figure 1 are described in table 1 and table 2 shows the flow for the DFAs in figure 2.

Algorithm 1 DFA equality

On input (M_1, M_2), where M_1 and M_2 are DFAs with start states p_0 and q_0, respectively
for all state $q \in M_1, M_2$ **do**
  $MAKE - SET(q)$
end for
Push the pair p_0, q_0 to the stack.
$UNION(p_0, q_0)$
while The stack is not empty **do**
  Pop pair q_1, q_2 from the stack.
  for all symbol $a \in \Sigma$ **do**
    Let $r_1 = FIND-SET(\delta(q_1, a))$ and $r_2 = FIND-SET(\delta(q_2, a))$, so that r_1 and r_2 are the names of the sets containing the successors to q_1 and q_2 .
    if $r_1 \neq r_2$ **then**
      UNION(r_1 ,r_2) and push the pair r_1, r_2 on the stack.
    end if
  end for
end while
If any set contains both an accepting state and a non-accepting state, then the two automata are not equivalent. Otherwise, the automata are equivalent.

Algorithm 2 $MAKE - SET(x)$

On globally accessible arrays p and $rank$
$p[x] \leftarrow x$
$rank[x] \leftarrow 0$

Algorithm 3 $FIND - SET(x)$

if $x \neq p[x]$ **then**
  $p[x] \leftarrow FIND - SET(p[x])$
end if
return $p[x]$

Algorithm 4 $LINK(x, y)$

if $rank[x] > rank[y]$ **then**
  $p[y] \leftarrow x$
else
  $p[x] \leftarrow y$
end if
if $rank[x] = rank[y]$ **then**
  $rank[y] \leftarrow rank[y] + 1$
end if

Algorithm 5 $UNION(x, y)$

$LINK(FIND - SET(x), FIND - SET(y))$

2 Specifications

2.1 Input Specifications

Your program will get two file names as parameters. The file structure that define the DFA shown in figure 1 is as follows (a=0,b=1,c=2,d=3 for the first file):

The first DFA:

```
4
2
0
2
1 3
2 3
3 3
3 3
```

The second DFA:

```
4
2
0
2
1 3
3 2
3 3
```

```
3 3
```

First line of the file gives the total number of states. Second line is the number of symbols. Third line gives the starting state. Fourth line gives the final states, which can be more than one. Rest of the lines gives the arcs for each of the states in order. Each number in a line gives the arc for one symbol. For example for the first DFA, fifth line is '1 3', which means state 0 goes to 1 for the first symbol and goes to 3 for the second symbol. States are numbered starting from 0.

2.2 Output Specifications

Your program should write the result to the standard output. If two automatons are equal, the output should be `equal`, otherwise it should be `notequal` with line breaks at the end.

2.3 Execution

Your program will be run with the following command.

```
runHK fileName1 fileName2
```

2.4 Submission

Your submission should be a tar file, only containing your code and the make file to compile your code in its root. You can do this with a command such as this: `tar -cvf TH.tar *.cpp *.hpp makefile`

3 Regulations

1. **Programming Language:** You must code your program in `C++`. Your submission will be compiled with `g++` on department lab machines.
2. **Late Submission:** There is no late submission.
3. **Evaluation:** Black box evaluation method is going to be used, therefore you need to be careful about input/output specifications. You should not print any unnecessary characters and/or white spaces. Missing make files will be evaluated as compile errors. Wrong executable names or wrong parameterization will be evaluated as if no output is produced.
4. **Cheating:** In case of cheating, the university regulations will be applied.

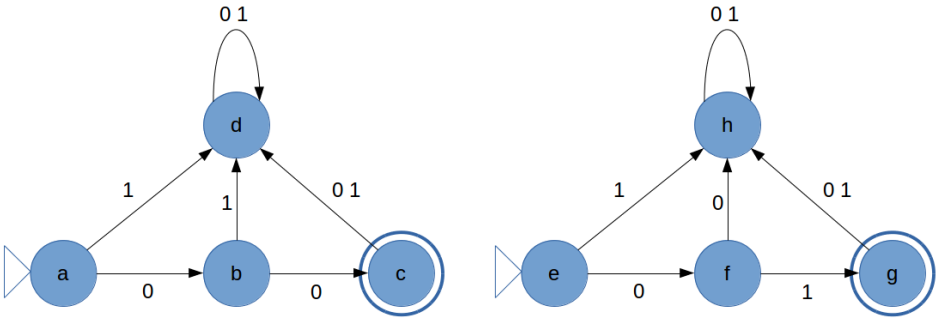

Figure 1: Two DFAs that accepts 00 and 01

Table 1: Flow of the algorithm for the DFAs in figure 1

SETS	**STACK**	
$\{a\}\{b\}\{\underline{c}\}\{d\}\{e\}\{f\}\{\underline{g}\}\{h\}$		Initial state
{a, e}$\{b\}\{\underline{c}\}\{d\}\{f\}\{\underline{g}\}\{h\}$	$\{a, e\}$	put starting states into the same set, push the pair into the stack
$\{a, e\}$**{b, f}**$\{\underline{c}\}\{d\}\{\underline{g}\}\{h\}$	$\{b, f\}$	pop $\{a, e\}$ and follow 0, merge the sets with b and f, push $\{b, f\}$
$\{a, e\}\{b, f\}$**{d, h}**$\{\underline{c}\}\{\underline{g}\}$	$\{d, h\}\{b, f\}$	from $\{a, e\}$ follow 1, merge the sets with d and h, push $\{d, h\}$
$\{a, e\}\{b, f\}\{h, d\}\{\underline{c}\}\{\underline{g}\}$	$\{b, f\}$	pop $\{d, h\}$, both 0 and 1 go to itself, nothing else to do
$\{a, e\}\{b, f\}$**{c, h, d}**$\{\underline{g}\}$	$\{c, h\}$	pop $\{b, f\}$ and follow 0, merge the sets with c and h, push $\{c, h\}$
$\{a, e\}\{b, f\}$**{c, h, d, g}**	$\{c, h\}$ $\{g, d\}$	from $\{b, f\}$ follow 1, merge the sets with g and d, push $\{g, d\}$
$\{a, e\}\{b, f\}\{\underline{c}, h, d, \underline{g}\}$	$\{g, d\}$	pop $\{c, h\}$ and follow 0, d and h are already in the same set
$\{a, e\}\{b, f\}\{\underline{c}, h, d, \underline{g}\}$	$\{g, d\}$	from $\{c, h\}$ follow 1, d and h are already in the same set
$\{a, e\}\{b, f\}\{\underline{c}, h, d, \underline{g}\}$		pop $\{g, d\}$ and follow 0, d and h are already in the same set
$\{a, e\}\{b, f\}\{\underline{c}, h, d, \underline{g}\}$		from $\{g, d\}$ follow 1, d and h are already in the same set
$\{a, e\}\{b, f\}$**{c, h, d, g}**		a set contains both accepting and non-accepting states, DFAs **are not** equal

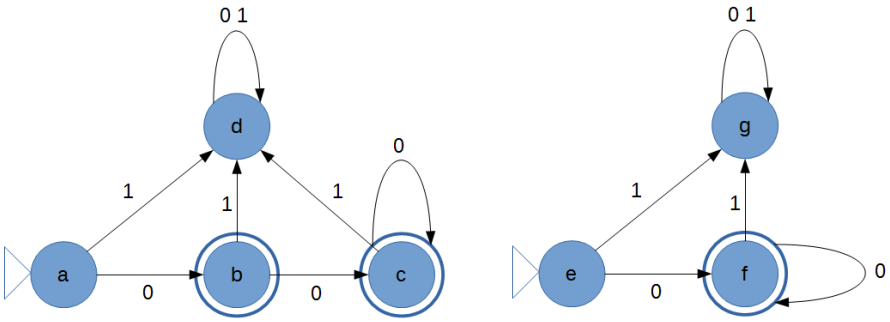

Figure 2: Two DFAs that accepts 0^+

Table 2: Flow of the algorithm for the DFAs in figure 2

SETS	**STACK**	
$\{a\}\{\underline{b}\}\{\underline{c}\}\{d\}\{e\}\{\underline{f}\}\{g\}$		Initial state
{a, e}$\{\underline{b}\}\{\underline{c}\}\{d\}\{\underline{f}\}\{g\}$	$\{a, e\}$	put starting states into the same set, push the pair into the stack
$\{a, e\}$**{b, f}**$\{\underline{c}\}\{d\}\{g\}$	$\{b, f\}$	pop $\{a, e\}$ and follow 0, merge the sets with b and f, push $\{b, f\}$
$\{a, e\}\{\underline{b}, \underline{f}\}$**{d, g}**$\{\underline{c}\}$	$\{d, g\}\{b, f\}$	from $\{a, e\}$ follow 1, merge the sets with d and g, push $\{d, g\}$
$\{a, e\}\{\underline{b}, \underline{f}\}\{d, g\}\{\underline{c}\}$	$\{b, f\}$	pop $\{d, g\}$, both 0 and 1 go to itself, nothing else to do
$\{a, e\}$**{b, f, c}**$\{d, g\}$	$\{c, f\}$	pop $\{b, f\}$ and follow 0, merge the sets with c and f, push $\{c, f\}$
$\{a, e\}\{\underline{b}, \underline{f}, \underline{c}\}\{d, g\}$	$\{c, f\}$	from $\{b, f\}$ follow 1, g and d are already in the same set
$\{a, e\}\{\underline{b}, \underline{f}, \underline{c}\}\{d, g\}$		pop $\{c, f\}$ and follow 0, c and f are already in the same set
$\{a, e\}\{\underline{b}, \underline{f}, \underline{c}\}\{d, g\}$		from $\{c, f\}$ follow 1, d and g are already in the same set
$\{a, e\}\{\underline{b}, \underline{f}, \underline{c}\}\{d, g\}$		no set contains both accepting and non-accepting states, DFAs **are** equal