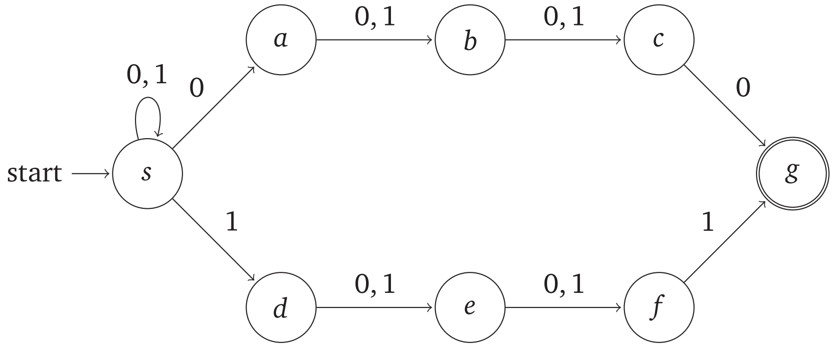
CS/ECE 374 B Fall 2019 Jiashuo Tong (jtong8)

**Homework 2 Problem 1**

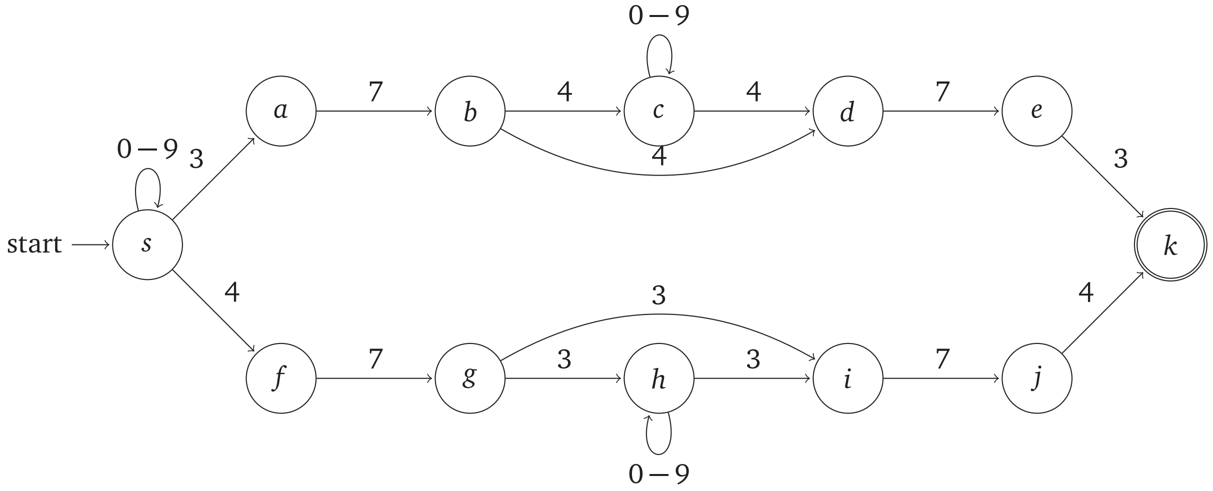
**Solution:**

(a) NFA that accepts the language that has two of the same characters at a distance 3 from each other



- $s$: strings that are arbitrary
- $a$: strings contain 0
- $b$: strings contain 00 or 01
- $c$: strings contain 000, 001, 010 or 011
- $d$: strings contain 1
- $e$: strings contain 10 or 11
- $f$: strings contain 100, 101, 110 or 111
- $g$: strings that contain substrings with 2 same characters at a distance 3

(b) NFA that accepts the language that contains either 374 or 473 as its substring



- $s$: strings that are arbitrary
- $a$: strings contain 3

- $b$: strings contain 37
- $c$: strings contain 374
- $d$: strings contain 374 and 4
- $e$: strings contain 374 and 47
- $f$: strings contain 4
- $g$: strings contain 47
- $h$: strings contain 473
- $i$: strings contain 473 and 3
- $j$: strings contain 473 and 37
- $k$: strings contain 473 and 374

■

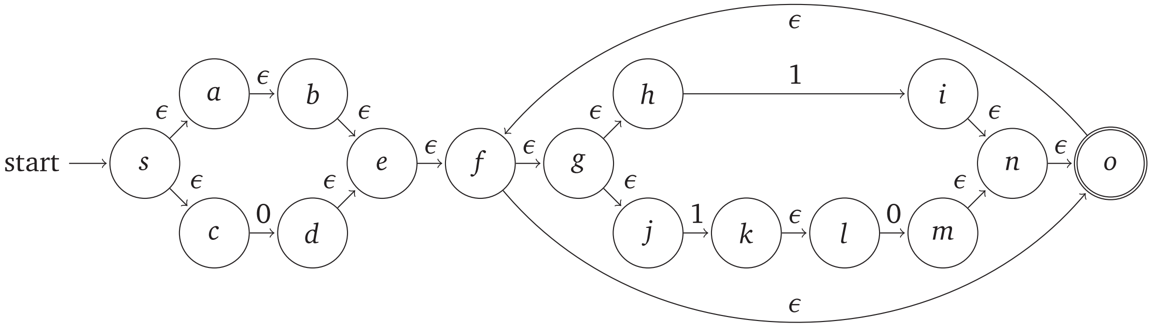
CS/ECE 374 B Fall 2019 Jiashuo Tong (jtong8)

**Homework 2 Problem 2**

**Solution:**

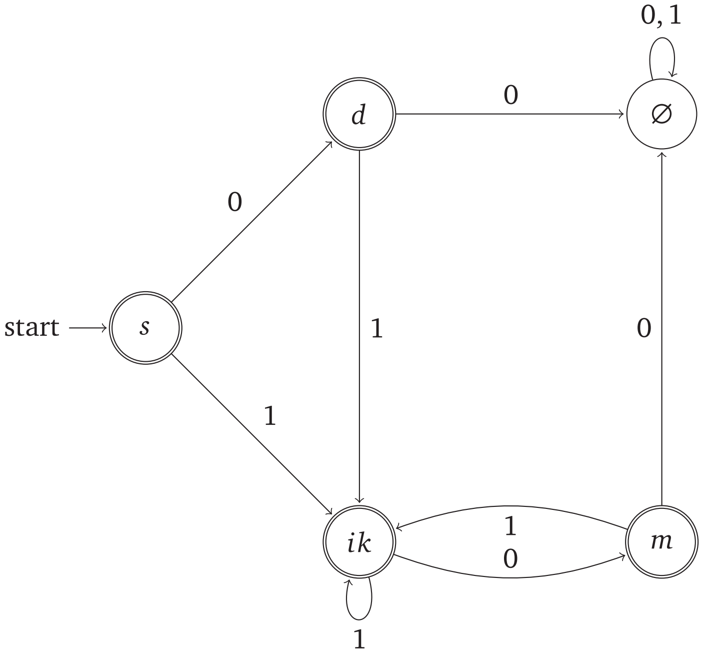
(a) For the regular expression $(\epsilon + 0)(1 + 10)^*$,

1. Construct an NFA corresponding to the regular expression using Thompson's algorithm



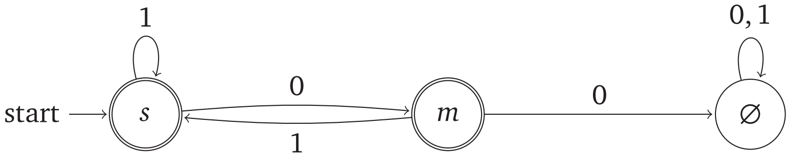
2. Use the incremental subset construction to convert the NFA to a DFA

| $q'$ | $\epsilon - reach(q')$ | $q' \in A'$ | $\delta'(q', 0)$ | $\delta'(q', 1)$ |
|---|---|---|---|---|
| $s$ | $sabcefghjo$ | ✓ | $d$ | $ik$ |
| $d$ | $efghjo$ | ✓ | $\varnothing$ | $ik$ |
| $ik$ | $fghjlno$ | ✓ | $m$ | $ik$ |
| $m$ | $fghjno$ | ✓ | $\varnothing$ | $ik$ |



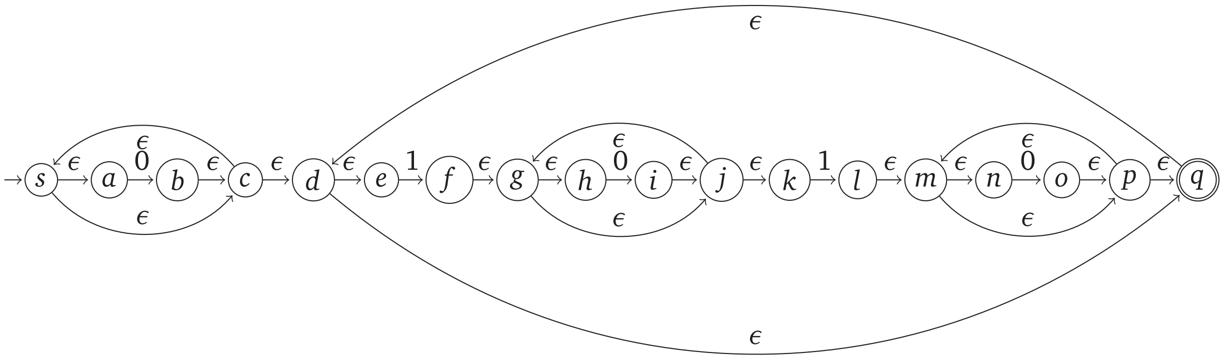
- $s$: strings are empty
- $d$: strings start with 0
- $ik$: strings start with 1
- $\varnothing$: strings start with 2 0's
- $m$: strings end with 10

3. Create another DFA with fewer states to recognize the language
   The DFA can be optimized by combining s with ik and combining d with m



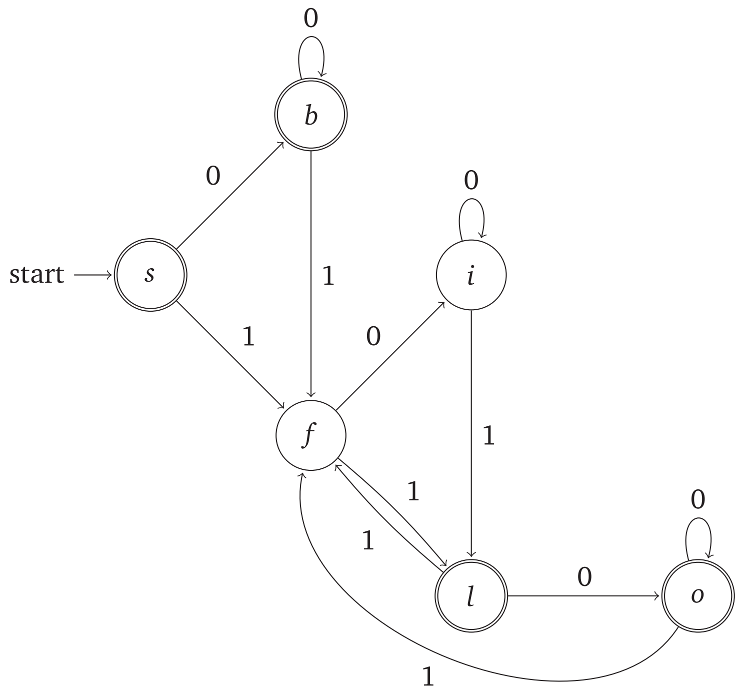
(b) For the regular expression $0^*(10^*10^*)^*$,

1. Construct an NFA corresponding to the regular expression using Thompson's algorithm



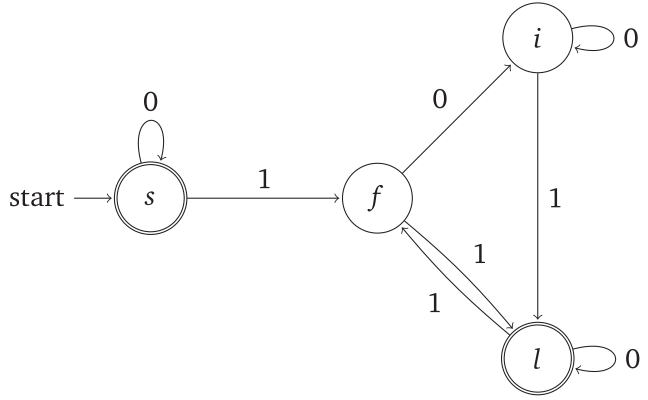
2. Use the incremental subset construction to convert the NFA to a DFA

| $q'$ | $\epsilon - reach(q')$ | $q' \in A'$ | $\delta'(q', 0)$ | $\delta'(q', 1)$ |
|---|---|---|---|---|
| $s$ | $sabcdeq$ | ✓ | $b$ | $f$ |
| $b$ | $sacdeq$ | ✓ | $b$ | $f$ |
| $f$ | $ghjk$ |  | $i$ | $l$ |
| $i$ | $ghjk$ |  | $i$ | $l$ |
| $l$ | $demnpq$ | ✓ | $o$ | $f$ |
| $o$ | $demnpq$ | ✓ | $o$ | $f$ |



- $s$: strings are arbitrary
- $b$: strings start with one or multiple 0's
- $f$: strings contain odd number of 1's
- $i$: strings contain odd number of 1's and end with 0
- $l$: strings contain even number of 1's
- $o$: strings contain even number of 1's and end with 0

3. Create another DFA with fewer states to recognize the language
   We can optimize the DFA by combining b with s and combing o with l



■