

*Project 1, Due: March 5, 2023*

The project will be done in teams of two or three members. (Three member teams should also solve Problem 2.) Only one submission is required per team. In addition to the code, please submit a short report that *states clearly the contribution of each member of the team*. Submission details will be specified in detail (along with test cases that should be included with the submission) in the canvas submission page. The project will be graded on (a) correctness on the test cases, (b) meeting the user interface requirements and (c) following standard practices for writing code that can be read by others (specific expectations are listed at the end). Around 5% of the total points will be assigned for (b) and around 10% points for (c).

**PROBLEM 1:**

Given a positive integer  $k > 0$ , and a subset  $S$  of  $\{0, 1, 2, \dots, 9\}$  of digits, the goal is to find the smallest positive integer  $N > 0$  such that  $N \% k = 0$  and  $N$  uses only the digits from the set  $S$ . As an example, if  $k = 26147$  and  $S = \{1, 3\}$ , the value of  $N = 1113313113$ . The obvious approach to this problem of generating the integers that use only the digits in  $S$  in succession and testing if the integer is divisible by  $k$  is extremely inefficient and will take years to solve on the test cases. By using a DFA, we can significantly speed up a solution to this problem.

Described below is the algorithm **FindString** that takes as input a DFA  $M$  and outputs a string  $w$  of shortest length (lexicographically first in case of more than one string) accepted by  $M$ . (If  $L(M)$  is empty, the algorithm outputs **No solution**.) Breadth-First Search (BFS) is used to solve this problem. The algorithm **FindString** takes as input a positive integer  $k$ , and a subset  $S$  of  $\{0, 1, 2, \dots, 9\}$  and outputs the smallest positive integer  $y > 0$  that is an integer multiple of  $k$ , and has only the digits (in decimal) from the set  $S$ . Here is a brief summary of BFS (which was presented in more detail in class. See the pseudo-code below for more details.) Initially, a Queue contains  $n$ , the start state. Also VISITED is set to True for  $n$  and False for all other states. Then, the search is performed until the Queue is empty or state 0 is reached: Delete  $j$  from the Queue and let NEXT be the set of states reachable from  $j$ :  $NEXT = \{ \delta(j, a) \mid \text{for all } a \in S \}$ , and insert for each  $x$  in NEXT such that  $VISITED[x] = \text{false}$  into the queue (and set  $VISITED[x]$  to True.) Also  $PARENT[x]$  is set to  $j$ . When the loop ends, if the QUEUE is empty, the DFA does not generate any string. Otherwise, your algorithm has found the shortest path from  $n$  to 0. By tracing the path (using the PARENT pointers) you can find the shortest string that accepted by the DFA. (Make sure to skip the null string as the shortest string.)

Using FindString algorithm, Problem 2 can be solved as follows. Create a DFA  $M = \langle Q, \Sigma, \delta, 0, F \rangle$  where  $Q = \{0, 1, \dots, k-1\}$ ,  $F = \{0\}$ , and  $\delta(j, a) = (10 * j + a) \% k$  and call FindString with  $M$  and  $k$  as inputs. (This DFA needs a small correction when  $S$  includes 0.)

For this problem, you can assume that  $k$  is in the range 1 to 99999.

Some test cases:

Test case 1:

Inputs:  $k = 92917$ , Digits permitted: 0, 1

Output: 11011011011001001

Test case 2:

Inputs:  $k = 198217$ , Digits permitted: 1

Output: integer containing 10962 ones (Your output will be a string of this many ones.)

Test case 3:

Inputs:  $k = 135$ , Digits permitted: 1 3 7

Output: No solution.

*BFS customized for this project*

As we discussed in class, the generic BFS has to be modified for this project in the following ways. In the usual applications of BFS, the underlying graph has no labels on the edges. But a DFA is a directed graph in which edges are labeled. So to keep track of the shortest path, parent array stores the previous state on this path, and the label array stores the corresponding edge label. Another difference is that this version of BFS terminates as soon as the accepting state is reached, while the standard version of BFS continues the search until all the vertices reachable from the starting vertex are located.

Algorithm FindString:

Input: DFA  $M$  with  $Q = \{0, 1, 2, \dots, k-1\}$ , 0 is the start state,  $S =$  input alphabet,  $\delta$

(In your implementation, the DFA need not be explicitly created; instead, the delta

Output: the shortest string accepted by  $M$ . If there is more than one of the same length

Initialize a queue  $Q$ ;

Initialize a Boolean array visited of size  $k$

Initialize an integer array parent of size  $k$

for all  $j$  in  $\{0, 1, \dots, k-1\}$  do  
    visited[j] = false;

for all  $j$  in  $\{0, 1, \dots, k-1\}$  do  
    parent[j] = -1;

for  $s$  in  $D$  do  
    next = delta(0,  $D[s]$ ,  $k$ );  
    visited[next] = true;  
     $Q.insert(next)$ ;  
    parent[next] = 0;  
    label[next] =  $s$ ;  
end for;

while ( $Q$  is not empty):  
    curr =  $Q.delete()$ ;  
    for each  $s$  in  $D \setminus \{0\}$  do:  
        next = delta(curr,  $s$ ,  $k$ ); // Recall  $\delta(q, r, k) = (10 * q + r) \% k$   
        if (next == 0): // accepting state is reached  
            parent[next] = curr  
            label[next] =  $s$   
            break // out of the while loop  
        else if not visited[next]:

```

        visited[next] = true;
        parent[next] = curr;
        label[next] = s;
        Q.insert(next)
    end (if)
end (for)
end (while)
if (next! = 0):
    output no solution
else
    trace the string using parent pointers and concatenate the corresponding labels a
    output the reverse of the string.
end FindString;

```

## PROBLEM 2:

This is a variation of Problem 2 in which you are given a positive integer  $k > 0$  and a subset  $D \subseteq \{0, 1, \dots, 9\}$  of the ten decimal digits. The goal of your program is to find the smallest integer  $N$  that uses only the digits in  $D$  such that both  $N$  and the reverse of  $N$  are (strictly positive) multiples of  $k$ . A simple example is: suppose  $k = 71$  and  $D = \{1, 2, 4\}$ , the solution is 2414142.