Birla Institute of Technology and Science, Pilani

Data Structures and Algorithms (CS F211)

Lab Assignment - 10

Instructions

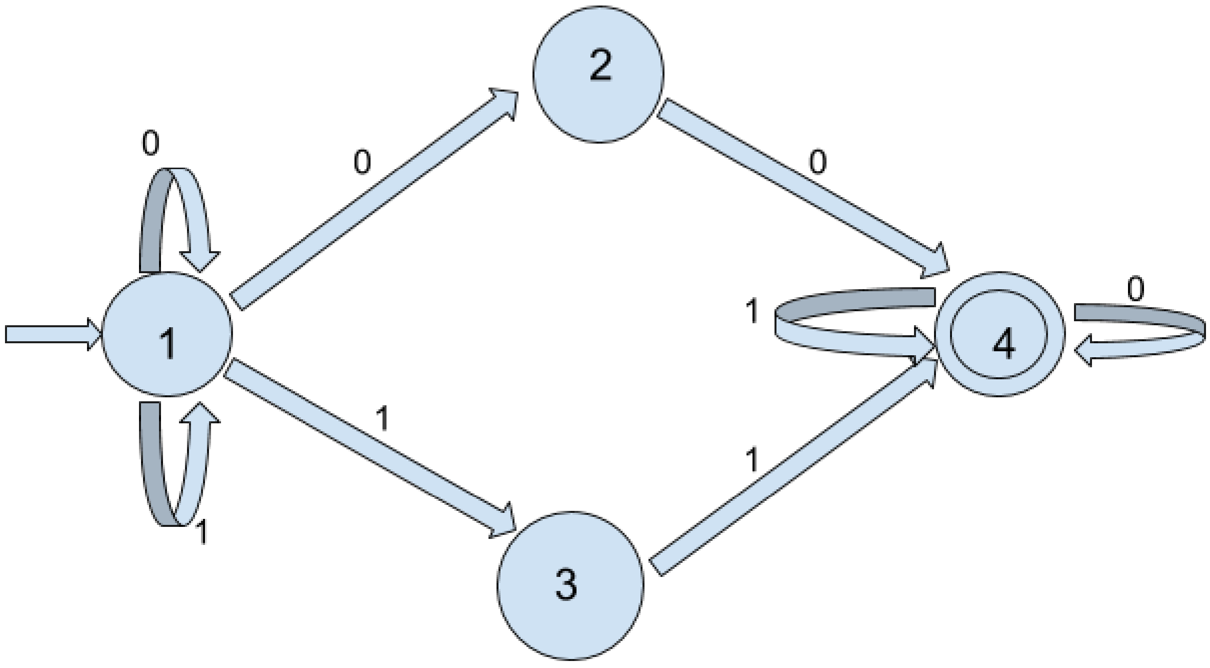
- All input should be read from stdin and output should be printed to stdout.
- Only the last submission by the student on the portal before the end of contest will be considered for evaluation.
- Following verdicts on the portal will **tentatively** fetch these marks:
  – Correct → 4 marks
  – Wrong-answer (correct for more than half test cases) → 3 marks
  – Run-error/Compiler-error/Time-limit-error → 2 marks
- All submitted source codes will be checked manually and by a plagiarism detector, and final marks will then be awarded and posted on Nalanda**.**
- **Important:** The problem should be solved by using the algorithm and data structures mentioned in the lab sheet. Failure to do so will result in a 1-mark penalty.

Simulation of Nondeterministic Finite Automata (NFA)

An NFA is a directed multigraph which can have edges that are self-loops. Each edge and vertex is labeled either 0 or 1. Vertices represent the states of the NFA. Those labeled 0 are non-accepting states, and those labeled 1 are accepting states. Vertices are numbered in sequence starting from 1 and the starting state will always be the vertex numbered 1. The start state can be either accepting or non-accepting. An input binary string is given to the NFA: $w = x_1x_2x_3x_4...x_n$ (w can also be an empty string of length 0). We begin with the vertex 1 which is always the start state. From vertex 1, we follow the edge x_1 ($x_1 \in \{0, 1\}$), to another state V_2. Now from V_2 we follow the edge x_2 to vertex V_3 and so on. Finally from state V_n we follow the edge x_n to state V_{n+1}. The term 'non-deterministic' means that at any state V_i we have more than one choice of following an edge labeled x_i. The NFA consumes the input $w = x_1x_2x_3x_4...x_n$ and it can be represented as a set of states Q' if we consider all possible moves of the NFA.

If *Q’* contains at least one final state, then we say that the NFA has accepted the input *w*. If all states in *Q’* are non-accepting, then we say that the NFA has rejected the input *w*.

Example of NFA



Starting state (vertex 1) is shown with an arrow. Accepting states are vertices with double circles (vertices with label 1). Vertex 4 is the only accepting state. Non-accepting states are shown with single circles (vertices with label 0). The non-accepting states here are vertices 1, 2 and 3.

Consider the input **1010**. In state 1, we have two possibilities, either follow the self loop and stay in state 1 or follow the edge labeled 1 and go to state 3.

{1} 1010 => {1, 3} 010

In state 3, there is no edge labeled 0, so the computation will die out. In state 1, we have two possibilities, either follow the self loop and stay in state 1, or follow the edge labeled 0 to state 2.

{1, 3} 010 => {1, 2} 10

Now there is no edge labeled 1 from state 2. The computation will die out. From V_1 we have two possibilities: either follow the self loop to state 1 or follow the edge labeled 1 to state 3.

{1, 2} 10 => {1, 3} 0

In state 3, there is no edge labeled 0. So the computation will die out. In state 1, we have two possibilities: either follow the self loop to state 1 or the edge labeled 0 to state 2.

{1, 3} 0 => {1, 2}

Now the NFA has consumed the input. It can be either be in states 1 or 2, both of which are non-accepting. So the NFA has rejected the input 1010.

Now consider the input **1100**.

    {1} 1100 => {1, 3} 100
    {1, 3} 100 => {1, 3, 4} 00
    {1, 3, 4} 00 => {1, 2, 4} 0
    {1, 2, 4} 0 => {1, 2, 4}

Now the NFA has consumed the input. It can either be in state 1, 2 or 4. State 4 is an accepting state. So, the NFA accepts the string 1100.

You can easily verify that the given NFA accepts all binary strings with “00” and/or “11” as a substring.

Input Format

You will be given the adjacency list representation of the NFA in the following format.

```
<Total Number of Vertices in the NFA>
1 <Label of vertex 1> <number of outgoing edges from vertex 1>
        <Label of edge 1> <Destination vertex>
        <Label of edge 2> <Destination vertex>
        <Label of edge 3> <Destination vertex>
        And so on….
2 <Label of vertex 2> <number of outgoing edges from vertex 2>
        <Label of edge 1> <Destination vertex>
        <Label of edge 2> <Destination vertex>
        <Label of edge 3> <Destination vertex>
        And so on….
And so on..
```

The given example will be represented as

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

Procedure

Backtracking will be used to solve the problem. Please refer to ‘Backtracking tutorial’ on Nalanda for the same.

Output Format

You have to output the first 10 binary strings which are accepted by the NFA, in lexicographical order (e denotes the empty string): {e, 0, 1, 00, 01, 10, 11, 000, ...}
The sample output for the given test case is as follow.

```
00
11
000
001
011
100
110
111
0000
0001
```