ACH2043 - Introdução à Teoria da Computação
Exercício de Programação: Emulação de Autômato Finito Não Determinístico

Objetivo:

Desenvolver um programa emulador de autômatos finitos não-determinísticos (AFNs).
Seu programa deve receber um único arquivo-texto contendo as especificações de m AFNs $M_1, M_2, \ldots, M_m$ e, para cada autômato M_i, um conjunto de cadeias de teste $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n$.
O programa deverá executar os autômatos $M_1, M_2, \ldots, M_m$ sobre suas respectivas cadeias de teste e devolver um arquivo-texto de valores 0/1, indicando quais cadeias são aceitas (1) ou rejeitadas (0) pelo seu autômato correspondente.
Em outras palavras, seu programa deverá decidir se cada cadeia de teste $\mathbf{w}_i$ pertence ou não à linguagem reconhecida pelo respectivo autômato.
O programa deverá ser desenvolvido em C, Java, R ou Python, e deverá ser executável via linha de comando do DOS ou Linux.

Chamada do programa e especificações dos arquivos

A chamada do programa será:

Implementação em C: afn.exe

Implementação em Java: java afn

Implementação em Python: python afn.py

Implementação em R: Rscript afn.r

O programa receberá **um arquivo de entrada** denominado `entrada.txt` e retornará **um arquivo de saída** denominado `saida.txt`. Esses dois nomes deverão ser *hard-coded*, e deverá ser assumido que ambos serão lidos e escritos no mesmo diretório do script ou módulo executável do programa.

Arquivo `entrada.txt`:

Este arquivo conterá as especificações dos AFNs e suas respectivas cadeias de testes.

O arquivo será organizado em blocos de linhas, da seguinte maneira:
m (número de autômatos de teste)
Bloco de linhas com a especificação do autômato 1
Bloco de linhas com as cadeias de teste do autômato 1
Bloco de linhas com a especificação do autômato 2
Bloco de linhas com as cadeias de teste do autômato 2

...
Bloco de linhas com a especificação do autômato m
Bloco de linhas com as cadeias de teste do autômato m

Cada bloco de linhas com a especificação de um autômato será organizado da seguinte forma:

- A primeira linha será um cabeçalho contendo seguintes campos separados por espaços:
  $q \quad s \quad t \quad q0 \quad a$
  onde:
  - q é um inteiro positivo indicando a quantidade de estados do autômato
    Obs: Os estados serão representados por números inteiros de 0 a $(q-1)$
  - s é um inteiro positivo indicando a quantidade de símbolos do alfabeto extendido Σ_ϵ (*incluindo a cadeia vazia*)
    Obs: o algarismo 0 é reservado para representar a cadeia vazia; todos os símbolos do alfabeto deverão ser representados por números 1 a $(s-1)$
    P.ex se o alfabeto de um AFN é $\Sigma = \{1, 2\}$, então teremos $s = 3$ e o alfabeto estendido Σ_ε será $\{0, 1, 2\}$ (0 representando ε).
  - t é um inteiro positivo indicando a quantidade de transições do AFN
  - $q0 \in \{0, 1, \ldots, q-1\}$ é o índice do estado inicial
  - a é um inteiro ≥ 0 indicando o número de estados de aceitação do AFN.
    Obs: Note que, embora não usual, a poderia ser igual a 0, indicando que o AFN não teria nenhum estado de aceitação e, portanto, rejeitaria quaisquer cadeias de entrada. Todavia, nos exemplos que trataremos na correção, não consideraremos este caso.

  s é o número de símbolos do alfabeto
- A segunda linha conterá os índices dos estados de aceitação (todos entre 0 e $(q-1)$), separados por espaços.
- As demais linhas do bloco (linhas 3 a $t+2$) conterão as especificações das transições, na forma:

  <índice estado corrente> <índice símbolo> <índice novo estado>.

  Exemplos:

  `0 1 2` → transição do estado q_0 para o estado q_2 com o símbolo 1

  `0 0 2` → transição do estado q_0 para o estado q_2 com a cadeia vazia (ε)

A especificação das cadeias será formada por um bloco de linha no seguinte formato:

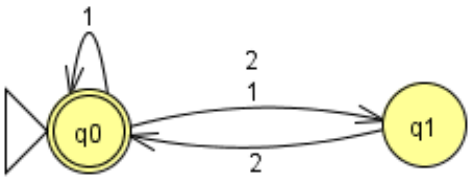
- A primeira linha conterá o número de cadeias de teste (inteiro positivo).
- Cada uma das demais linhas do bloco conterá uma cadeia, com seus símbolos separados por espaços. A cadeia vazia será representada por 0.

  Obs: o símbolo 0 só deverá aparecer se for a cadeia vazia. Ele não aparecerá em cadeias não vazias

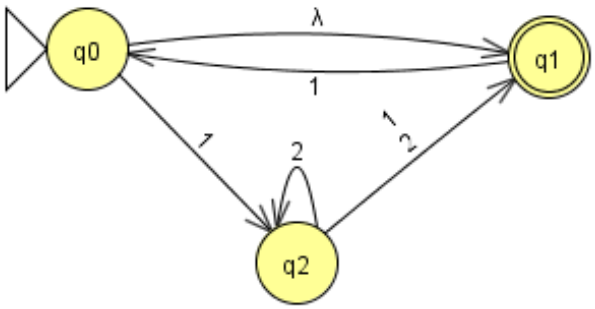
  P.ex. o arquivo de entrada não deverá conter cadeias como "`0 2 1`" ou "`1 2 0 1`"

O exemplo abaixo ilustra um arquivo de testes referente aos dois automatos da figura abaixo:

Autômato 1



Autômato 2




```
2          numero de automatos de teste
2 3 4 0 1  AFN 1: 2 estados, 3 simbolos, 4 transicoes, est.inicial=0, 1 est. aceit
0          F = {0}
0 1 0      especificações das transicoes
0 1 1
0 2 1
1 2 0
6          numero de cadeias de teste do automato 1
1          1a cadeia de teste
1 1        2a cadeia
1 1 1
1 2 2 1 1 1 2 2 1
2 2 2 1
2 1 2 2
3 3 6 0 1  AFN 2: 3 estados, 3 simbolos, 6 transicoes, est.inicial=0, 1 est. aceit
1          F = {1}
0 0 1      exemplo de transicao com cadeia vazia
1 1 0
0 1 2
2 2 2
```

```
2 1 1
2 2 1
8          numero de cadeias de teste do automato 1
0          cadeia de teste 1 (cadeia vazia)
1          cadeia de teste 2
1 2 2 2 2 2 1
1 1 1 2 1 1 1
1 2 1 1
1 2 2 1 2 2
2
1 1 2 2 1 2
```

<ArqSaida>:
O arquivo de saida contera seqüências de 0s e 1s separados por espaço, indicando a aceitação (1) ou rejeição (0) das cadeias pelos respectivos autômatos. Cada linha conterá os resultados dos testes das cadeias de um autômato.

Para arquivo de entrada ilustrado acima, o arquivo de saída resultante seria composto pelas seguintes linhas

```
1 1 1 1 0 0      Aceitacoes/rejeicoes das cadeias de teste do automato 1
1 1 1 1 1 0 0 0  Aceitacoes/rejeicoes das cadeias de teste do automato 2
```

Entrega do trabalho:

Condições da entrega:

- O trabalho poderá ser feito em grupos de ATÉ dois alunos, devidamente identificados na primeira linha do código-fonte.
- Deverá ser entregue um diretório compactado (formato .zip) contendo o arquivo fonte e o executável (ou classe Java compilada). O diretório deve ser nomeado na forma d_*numerousp1*_*numerousp*
  Exemplo: d_1234567_7654321.zip
- O módulo principal deve ter o nome glc.c, glc.java, glc.py ou glc.r. Nas 1as linhas do código fonte deverá constar os nomes e números USP dos membros do grupo.
- O código-fonte deverá ser compilável ou executável via comando gcc, javac, py ou Rscript. Se desenvolver seu programa em IDEs como Eclipse ou Netbeans, certifique-se de que seu programa seja compilável sob as condições aqui expostas.
- Inclua também, no diretório de seu ep, um arquivo chamado LEIAME.TXT contendo:
  - A linguagem utilizada
  - A versão utilizada do compilador ou ambiente
  - IDE utilizada (se for o caso) e sua versão

- Sistema operacional em que desenvolveu e compilou seus programas
- Indicação de pacotes não nativos que precisam ser instalados - A linha de comando exata a ser digitada no prompt do sistema operacional para compilar seu programa

- O trabalho deverá ser submetido via E-DISCIPLINAS.
  Não é necessário que os dois alunos do grupo enviem o código-fonte, basta uma entrega por grupo.

- Dúvidas a respeito das especificações ou a respeito da implementação do trabalho serão sanadas até uma semana antes do prazo indicado no E-DISCIPLINAS. Dúvidas encaminhadas após este prazo correrão o risco de ficarem sem resposta.

- Além da correção do programa, será considerada a qualidade da documentação do código fonte.

- Se houver evidência de plágio entre trabalhos de grupos distintos de qualquer uma das turmas, os mesmos serão desconsiderados.