Relatório EP4 - MAC0323

Anahí Coimbra Maciel | N°USP: 11809127

Julho de 2023

Esse relatório é sobre o EP4 da disciplina Algoritmos e Estruturas de Dados II, lecionada pelo professor Carlos Ferreira no primeiro semestre de 2023. O programa foi escrito em C++. O programa respeita todas as restrições apresentadas na proposta.

1 Como compilar e executar o EP

Para compilar o EP simplesmente rode no diretório do EP o comando:

```
g++ EP4.cpp -o EP4
```

Em seguida, para executá-lo, utilize:

```
./EP4
```

2 Estrutura e funções implementadas

Para esse EP foi implementada a estrutura *Graph*, que corresponde a um grafo dirigido que representa um autômato finito não determinístico (NFA). Essa estrutura foi implementada a partir de uma matriz de adjacência:

```
vector<vector<bool>> adj;
```

Além disso, há um vetor letters, que guarda em cada posição o valor contido no vértice do grafo correspondente:

```
vector <char> letters;
```

As funções (além do construtor) implementadas na classe *Graph* foram:

```
private:

void dfs (int vertex, bool* visited);

void addEdge(int u, int w);

public:

void buildGraph(string exp);

bool testWord(string word);
```

Respectivamente, as funções:

1. Faz a busca em largura a partir do vértice dado;
2. Adiciona um arco partindo do vértice u para o vértice w;
3. Faz o NFA a partir da expressão regular.
4. Testa a palavra dada, verificando se há um caminho no grafo que leve à aceitação da palavra.

Além disso, algumas funções auxiliares foram implementadas:

```
bool isCharacter(char c);
bool isLetter(char c);
bool isNumber(char c);
bool bothNumbersOrLetters (char a, char b);
```

Respectivamente:

1. Retorna se o char c não é caractere especial;
2. Retorna se o char c é letra;
3. Retorna se o char c é número;
4. Retorna se os char a e b são ambos letras ou números.

3 Notas sobre o funcionamento do EP

O EP recebe uma entrada pela linha de comando, o nome do arquivo de entrada com a expressão regular, o número de palavras a serem testadas e as palavras a serem testadas. O programa constrói o NFA correspondente à expressão regular e testa a validez de cada palavra, imprimindo "S"se a palavra é aceita e "N"se não.

3.1 Notas sobre a solução encontrada

O EP se baseou na implementação apresentada pelo professor em sala, com algumas adições para contemplar os símbolos adicionais citados na proposta:

1. Para implementar o uso da barra invertida, foi adicionada uma condição na função testWord: caso o caractere analisado da palavra seja igual ao próximo vértice do grafo, ele é aceito;
2. Para implementar o coringa, foi adicionado uma condição na função testWord: no caso do coringa, qualquer caractere é aceito;
3. Para implementar conjuntos, intervalos e complementos, foram adicionadas condições na função testWord: caso o caractere analisado da palavra corresponda às condições do conjunto, intervalo ou complemento, ele é aceito;
4. Para implementar o um ou mais, na função buildGraph adiciona-se um arco entre o + e o próximo vértice, e um arco para o (correspondente.

Além disso, deve-se salientar que devido a particularidades da implementação, o programa só funciona corretamente se o uso de parênteses for correto e da seguinte forma:

1. No caso de alternativas: (x|y)
2. No caso de fecho ou um ou mais: (x)* ou (x)+

Por causa disso, a terceira entrada apresentada na proposta teve de ser alterada, resultando em:

```
(((A)*CG | (A)*TA )| (AAG)*T)*
```

Por fim, no caso dos intervalos não se considerou diferenças entre letras maiúsculas e minúsculas.

4 Testes

Foram feitos testes com as quatro entradas da proposta do EP e mais duas entradas feitas por mim. Com essas seis entradas, foi possível testar todas as funções do EP. Abaixo estão as entradas e saídas dos testes:

```
Input:
(([a-z])*|([0-9])*)*@(([a-z])+\.)+br
2
cef1999@ime.usp.br
thilio@bbb.com
Output:
S
N
Input:
(.)*A(.)*
4
AAAAAAAAA
BCA
AAAAABBBBBB
BBB
Output:
S
S
S
N
Input:
(((A)*CG | (A)*TA )| (AAG)*T)*
4
AACGTAAATA
CAAGA
ACGTA
AAAGT
Output:
S
```

```
N
S
N
Input:
[^AEIOU][AEIOU][^AEIOU][AEIOU]
5
GATO
FINO
OLHO
BELO
RUSSO
Output:
S
S
N
S
N
Input:
\(([0-9])+\) 9[0-9][0-9][0-9][0-9]\-[0-9][0-9][0-9][0-9]
6
(86)95152-1314
(11)91234-5678
(11)1234-5678
(-)99999-1234
()
(00)9ABCD-EFGH
Output:
S
S
N
N
N
N
Input:
([A-Z])+([0-9])*
5
abcd12334
a
ab34a
56
ABCD
Output:
S
S
N
N
S
```

Percebe-se que as todas as saídas correspondem aos valores esperados.