SDI0001 (Fila de Mensagens): Criptografia Química

1 Descrição

A aplicação Criptografia Química apoia-se no significado da própria palavra criptografia (em grego: kryptós, "escondido", e gráphein, "escrita") e da base da química, a tabela periódica, para montar um código próprio para tradução de textos escondidos. O objetivo da aplicação é dispor de um servidor de tradução criptográfica, que recebe de um cliente uma mensagem criptografada e retorna-a, descriptografada. O cliente, desenvolvido na linguagem Java e midleware Rabbit MQ, tem por função: o recebimento de um arquivo texto, a leitura (por STDIN), e o envio da mensagem ao servidor. O servidor, por sua vez, identifica o elemento químico, traduz para o símbolo ASCII corres-

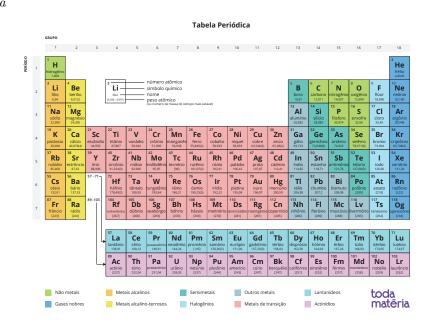


Figura 1: Tabela Periódica

pondente, gera a mensagem descriptografada e retorna-a ao cliente. A mensagem só considera os símbolos codificados em ASCII entre 032 (espaço) e 126 (~acento til). O algoritmo de criptografia faz a associação direta do primeiro código decimal da fração especificada da tabela ASCII ao primeiro elemento na tabela periódica e assim sucessivamente. Por exemplo, o símbolo 'espaço' é codificado pelo número 32 da tabela ASCII (primeiro símbolo da fração especificada da tabela ASCII) e terá, para o nosso algoritmo, o correspondente químico 'H' (Hidrogênio) na tabela periódica (número 1). Este mecanismo de substituição é modificado nas demais ocorrências do símbolo. Suponho o recebimento da seguinte palavra criptografada: SePoDy2.0Mo. O elemento químico Selênio (Se - 34 na tabela periódica) será traduzido para 'A' (065 na Tabela ASCII); O Polônio (Po - 84) será traduzido como 's'; O Disprósio (Dy - 66) será traduzido como 'a'; e O Molibdênio (Mo - 42) será traduzido como 's', pois tem dois parâmetros associados, o primeiro ao número de ocorrência e, o segundo, ao arredondamento. No caso do 2.0Mo, temos 2 * 42, 2 devido a segunda ocorrência do 's' na palavra, e o 42 do elemento Lantânio. O parâmetro .0, indica que não há a necessidade de soma de 1 no elemento, devido ao arredondamento. Portanto, para associar 2.0Mo ao 's', tem-se o seguinte cálculo: (2*42) + 31 + 0 = 115. Finalmente, a palavra final será: **Asas**. Outros exemplos com parâmetros, 3.0Ni (28): 3*28+31+0=115, portanto a tradução da terceira ocorrência da letra 's' (115). O 3.1Ni, já representaria a terceira ocorrência da letra 't' (3*28+31+1=116) na palavra.

2 Arquivo de Entrada

A única forma de entrada de dados é por meio do arquivo de entrada, respeitando o formato pré-definido. Um exemplo de arquivo de entrada é o descrito na sequência. Não existe nenhum espaço (032) no arquivo.

3 Arquivo de Saída

O resultado final da aplicação é determinista. A análise do algoritmo se dará pela verificação do arquivo de saída (Figura 3).

Nro_clientes: 2 Processos: Send Send Maquinas: ens2 ens4

Clientes ## SePoDy2.0Mo XeFrDy

Figura 2: Exemplo de arquivo de entrada

O método de comparação é o textual caractere a caractere, portanto é essencial que o formato especificado seja respeitado. Observe que o arquivo de saída não possui NENHUM caractere invisível, *i.e.*, espaço.

4 Tecnologia

O referido projeto deve ser desenvolvido na linguagem de programação Java e se apoiar no middleware de comunicação Rabbit MQ. No Git da disciplina é possível baixar um código exemplo de Rabbit MQ (https://github.com/mauriciopillon/sdi0001/tree/master/06-RabbitMQJava). As ferramentas de desenvolvimento estão instaladas na Plataforma de Desenvolvimento da disciplina (Laboratório de Processamento Paralelo e Distribuído (LabP2D)), e a aplicação deve, obrigatoriamente, ser validada na Plataforma de Submissão (BOCA) - (http://144.22.207.54/boca/src/). O executável do servidor deverá ser lançado na máquina ens1, via script, e o cliente na máquina

Cliente (1) #### SePoDy2.0Mo: Asas ### Cliente (2) #### XeFrDy: Uva

Figura 3: Exemplo de arquivo de saída.

especificada no arquivo ambiente.in. Apoiem-se nos scripts do problema teste, já disponível no BOCA, adequando o que for necessário.