Exercício de Programação 1: Eficiência de pilhas

Prof. Márcio Moretto Ribeiro

2 de Setembro de 2020

1 Introdução

A disciplina de estrutura de dados nos ensina que diferentes formas de representar os dados em um problema pode levar a algoritmos muito mais eficientes. Neste exercício de programação a intenção é testar a eficiência de duas implementações para um mesmo problema.

2 O Problema

Suponha um situação em que uma máquina precisa processar uma sequência de tarefas. Essa máquina é capaz de processar apenas uma tarefa por vez, porém o fluxo de entrada é maior do que o tempo de processamento. A máquina processa primeiro a última requisição que recebeu (como uma pessoa lavando pratos lava primeiro o prato que está no topo da pilha de pratos). Cada tarefa que a máquina precisa processar é identificado com um identificador único representado por um número inteiro menor ou igual a 999999. A máquina gera um arquivo de registros (log) das tarefas que recebe e indica nesse arquivo os momentos em que processou uma tarefa. A tarefa que cabe a vocês é indicar a ordem em que as tarefas foram processados.

3 A Entrada

O arquivo entradas.zip contém um conjunto de arquivos. Cada linha possui do arquivo uma única instrução representada da seguinte forma:

- Uma linha com um identificador indica que uma tarefa foi inserida na pilha.
- Uma linha vazia indica que a máquina processou a tarefa que estava no topo da pilha.

Os arquivos de entrada possuem tamanhos distintos, mas eles nunca ultrapassam 100 mil instruções.

4 A Saída

Para cada arquivo de entrada, o programa deve produzir um arquivo de saída contentendo os identificadores das tarefas na ordem em que eles foram processados.

Exemplo 1 *Arquivos de entrada e saída.*

Entrada	*Saída*
306741 783580 54939 797499 235178 675900 365768 561760 986962 242452	783580 235178 797499 561760 365768 675900 54939

5 A Tarefa

Um programador ingênuo resolveu o problema implementando a seguinte classe:

```
public class PilhaIngenua {
  protected final int MAX = 1000000;
  protected Integer[] pilha;

  PilhaIngenua(){ pilha = new Integer[MAX]; }

  void add(int newElement) {
    int i;
    for(i = 0; pilha[i] != null; i++);
    pilha[i] = newElement;
  }

  int remove() {
    int i;
    for(i = 0; pilha[i] != null; i++);
    int tmp = pilha[i - 1];
    pilha[i - 1] = null;
    return tmp;
  }
}
```

A tarefa de vocês é escrever dois programas que solucionem o problema:

1. O primeiro deve usar a classe PilhaIngenua.
2. O segundo deve implementar uma lista simplesmente ligada como vimos em aula.

Ambos devem produzir as mesmas saídas.

Vocês devem usar o método `System.currentTimeMillis()` para avaliar o tempo de execução de cada um dos dois programas para cada arquivo.

6 A Entrega

Vocês devem entregar um arquivo PDF com um relatório de não que 3 páginas descrevendo o que fizeram. O arquivo precisa obrigatoriamente possuir:

- Os nomes dos membros do grupo (até 3 membros) com os respectivos NUSPs.
- Um link para o programa (preferencialmente no github).
- Um link para os arquivos de saída.
- Uma descrição de como rodar o programa.
- Um gráfico de linha em que o eixo x é o tamanho número de linhas do arquivo de entrada e o eixo y o tempo de processamento de cada um dos programas.

7 Extra

O problema da implementação ingênua não é o fato de usar um arranjo, mas a forma como o arranjo é usado. O relatório pode incluir a implementação da classe `PilhaArranjo` que resolve o mesmo problema em tempo constante usando arranjos. A comparação do desempenho de tempo com essa terceira implementação também é bem vinda.