Trabalho Prático 1 - Estrutura de dados

Victória Olívia Aráujo Vilas Boas - 2019431429

Belo Horizonte, 3 de Outubro 2019

1 Introdução

O objetivo do trabalho proposto é ajudar o cientista Rick Sanchez a descobrir a quantidade de frascos que irá utilizar na realização de seus experimentos malucos.

Rick deve ser capaz de armazenar todos os frascos que possui no laboratório em uma estrutura de dados, e deve ser capaz de excluir dessa estrutura os frascos quebrados ao longo do tempo.

Além disso, Rick deve ser capaz de consultar o número de frascos que serão utilizados na realização de um determinado experimento. Para isso, ele deseja informar a quantidade de mililitros que o experimento exigirá, e deseja receber como resposta a quantidade mínima de frascos que precisará organizar para conseguir comportar os mililitros que deseja.

Para tanto, a resolução desse desafio contará com duas estruturas de dados implementadas em código em linguagem c++. Uma delas armazenará os frascos, e a outra o histórico de operações realizadas envolvendo eles. Essas estruturas serão melhor detalhadas nas próximas seções.

2 Implementação

2.1 Estílo de Código

O estilo de código adotado na implementação do problema segue o estilo de código do projeto Unix, em sua maioria.

2.2 Arquitetura

A arquitetura adotada no problema conta com a seguinte estruturação de pastas:

main.cpp
Makefile
header

Dentro da pasta header são armazenadas as estruturas que serão utilizadas para a resolução do problema proposto, de modo a separar a declaração das classes de seus respectivos métodos:

LinkedList.cpp
LinkedList.hpp
Node.hpp
Queue.cpp
Queue.hpp

2.3 Classes e métodos

Para a implementação do problema, foi contruida a classe LinkedList. Essa classe foi utilizada para armazenar os frascos disponíveis no laboratório de Rick. Para que a classe atendesse ao problema, ela precisaria ser capaz de adicionar elementos a uma lista, e remover algum elemento pertencente a lista, que o usuário porventura desejasse. Para tanto, a classe conta com os seguintes métodos:

push: método utilizado para adicionar um valor ao final da lista.

remove_by_value: método utilizado para remover valor passado como parâmetro, de qualquer lugar da lista.

print: mètodo utilizado para imprimir os valores da lista. Esse método não utilizado no resultado final do trabalho, mas foi importante ao longo da implementação para fins de debbug.

Além da classe LinkedList, foi necessária a criação de uma fila - classe Queue - para adicionar todas as operaçoes realizadas entre os valores da lista. Essa classe deveria ser necessáriamente uma fila pois seria necessário remover valores do começo da estrutura.

Assim, para a implementação da fila foram herdados métodos da lista, e foi adicionado o método ***first_out*** como método exclusivamente da fila para a preservação da integridade conceitual das estruturas de dados. Como resultado, o objeto Queue conta com todos os mtodos herdados do objeto LinkedList, com a adiao do método ***first_out***.

2.4 Funções e detalhes de implementação

A entrada do usuário será um ***inteiro*** seguido de um ***char***, em que ele especificará se o valor inteiro será inserido na estrutura de lista, se o valor inteiro será removido da estrutura, ou se o valor inteiro é o valor em mililitros que ele deseja consultar para saber o número de frascos mínimos necessários para formá-lo.

Quando o usuário optar por fazer uma consulta do número de frascos, o programa irá chamar a função ***calculatesFlasks***, passando como parâmetro a quantidade de mililitros. Essa função irá armazenar todos os frascos disponíveis na lista em uma fila, e em seguida fará operações com esses valores da seguinte forma:

1. Retira um valor do começo da fila
2. Cria uma fila para armazenar o numero de operações realizadas com os numeros.
3. Soma e subtrai com cada um dos valores da lista, armazenando o resultado das operações no fim da fila, e o numero de operações realizadas com esse valor na mesma sequencia.
4. Se o valor da operação é igual ao valor que o usuário deseja consultar, retornar o número de operações realizadas.
5. Se o valor da operação for maior que zero, armazena no fim da lista e recomeça o processo repetindo o passo 1, 2, 3, e 4.

3 Intruções de compilação e execução

Para compilar o código adequadamente é necessario seguir os seguintes passos:

1. Entrar na pasta ***src*** com o comando:

```
cd src
make tp1
```

2. Rodar o programa com o comando:

```
./tp1
```

4 Análise de complexidade

1. A complexidade do número de operações de soma e subtração no ***melhor caso***:

O melhor caso acontece quando os mililitros consultados equivalem a primeira posição da lista de frascos. Nesse caso, no inicio da lista, antes de inserir na fila, o algoritmo irá encontrar o valor e irá retorná-lo para o usuário.

A ordem de complexidade é:

$$\mathcal{O}(1) \tag{1}$$

2. A complexidade do número de operações de soma e subtração no ***pior caso***:

No pior caso as operações de soma e subtração irão ocorrer n vezes em uma entrada m (tamanho da lista).

Desse modo, a complexidade será exponencial:

$$\mathcal{O}(m^n) \tag{2}$$

5 Conclusão

Assim, nota-se que o custo desse algoritmo é alto, porém é um algoritmo eficiente para os casos de testes apresentados.