Atividade Avaliativa II

Estruturas de Dados Básicas I Instituto Metrópole Digital

2020.6

Introdução

Nesta atividade você deve implementar três estruturas de dados: lista duplamente encadeada, pilha e fila. As implementações devem respeitar a complexidade máxima das operações.

1 Descrição

Utilizamos estruturas de dados para resolver diversos problemas e a escolha de uma estrutura pode impactar diretamente na eficiência da solução. Estruturas de dados como listas, filas e pilhas foram criadas para solucionar problemas que possuem características específicas:

- **Lista Encadeada ou Lista Duplamente Encadeada:** utilizada quando precisamos inserir mais objetos do que consultar;
- **Pilha:** utilizada quando acessar o elemento inserido mais recentemente primeiro;
- **Fila:** utilizada em problemas em que a ordem de inserção é importante no processamento dos elementos;

Todas as estruturas possuem características distintas de funcionamento e com isso complexidades diferentes para operações semelhantes. Por exemplo, a fila só pode inserir um elemento no final, mas uma lista encadeada pode inserir em qualquer posição. Apesar de limitada, a lista concede uma complexidade constante para esta operação, já a lista encadeada possui complexidade linear. Não faz sentido utilizar uma lista encadeada para simular uma fila, já que as operações custarão tanto quanto uma lista encadeada, porém possuirão a limitação de uma fila. Por este motivo, nesta atividade vocês devem implementar as estruturas respeitando a complexidade das operações, que serão descritas nas seções a seguir.

2 Lista Duplamente Encadeada

A lista duplamente encadeada funciona de forma parecida com a Lista Encadeada, porém cada nó, além do ponteiro pro próximo elemento, também possui um ponteiro pro elemento anterior. Este ponteiro serve para otimizar algumas operações que necessitam saber da posição do elemento anterior e do próximo (inserção do meio da lista, por exemplo). Um nó de uma lista duplamente encadeada geralmente tem a seguinte interface:

```
template<typename T>
class Node
{
    T value;
    Node* prev;
    Node* next;
};
```

As seguintes operações devem ser implementadas:

- `size()` retorna a quantidade de elementos na lista. Complexidade: $O(1)$.
- `insertAt(value, position)` insere o elemento `value` na posição `position`. Complexidade: $O(n)$.
- `removeAt(position)` remove o elemento na posição `position`. Complexidade: $O(n)$.
- `insertAtFront(value)` insere o valor `value` na primeira posição da lista. Complexidade: $O(1)$.
- `insertAtBack(value)` insere o valor `value` na última posição da lista. Complexidade: $O(1)$.
- `removeAtFront()` remove o primeiro valor da lista. Complexidade: $O(1)$.
- `removeAtBack()` remove o último valor da lista. Complexidade: $O(1)$.
- `print()` Imprime todos os elementos da lista. Complexidade: $O(N)$.

3 Pilha

A pilha deve possuir as seguintes operações:

- `size()` retorna a quantidade de elementos na pilha. Complexidade: $O(1)$.
- `push(value)` insere o valor `value` no topo da pilha. Complexidade: $O(1)$.
- `top()` retorna o elemento do topo da pilha. Complexidade: $O(1)$.
- `pop()` remove o elemento do topo da pilha. Complexidade: $O(1)$.

4 Filas

A fila deve possuir as seguintes operações:

- `size()` retorna a quantidade de elementos na pilha. Complexidade: $O(1)$.
- `pushBack(value)` insere o valor `value` no final da fila. Complexidade: $O(1)$.
- `popFront()` remove o primeiro elemento da fila. Complexidade: $O(1)$.
- `front()` retorna o primeiro elemento da fila. Complexidade: $O(1)$.
- `back()` retorna o último elemento da fila. Complexidade: $O(1)$.

5 Testando as Estruturas

Para testar as estruturas, vocês devem criar um programa para realizar as operações nelas. O programa deve aceitar apenas um parâmetro de linha de comando: o nome do arquivo com as instruções.

5.1 Arquivo com instruções

Inicialmente o programa deve criar uma estrutura de cada tipo, em seguida o arquivo deve ser lido e as instruções devem ser realizadas na estrutura específica. Uma linha do arquivo possui o seguinte formato:

```
<estrutura> <operação> <parâmetros>
```

Abaixo segue um exemplo de arquivo de testes:

```
stack push 10
list insertAtBack 1
list insertAtBack 3
list insertAt 2 1
list print
stack top
stack pop
stack size
queue pushBack 10
queue pushBack 20
queue popFront
queue front
queue back
```

O programa deverá ler o arquivo linha a linha e informar as operações que estão realizando. Um exemplo possível do programa anterior seria:

```
./prog instrucoes.txt

Inserindo elemento na pilha: 10
Inserindo elemento no final da lista: 1
Inserindo elemento no final da lista: 3
Inserindo elemento na posição um da lista: 3
Imprimindo lista:
1
2
3
Elemento do topo da pilha: 10
Removendo elemento da pilha
Tamanho da pilha: 0
Inserindo elemento da fila: 10
Inserindo elemento da fila: 20
Removendo elemento da fila
Primeiro elemento da fila: 20
Último elemento da fila: 20
```

Atenção: seu programa deve aceitar todas as operações das três estruturas.

5.2 Dicas de implementação

- Como os métodos possuem quantidades diferentes de parâmetros, você terá que ler cada linha de forma diferente. Uma dica para fazer isto, é usar o `std::getline()` para ler o arquivo linha a linha. De posse do conteúdo da linha, você pode utilizar um `std::stringstream` para ler os valores necessários.

6 Avaliação

O trabalho possui uma pontuação máxima de 100 pontos e deve ser feito **individualmente**. Esses pontos estão distribuídos da seguinte forma:

- Implementação da fila — até **20 pontos**
- Implementação da pilha — até **20 pontos**
- Implementação da lista duplamente encadeada — até **30 pontos**
- Implementação do programa de testes — até **30 pontos**

7 Colaboração e Plágio

Ajudar o colega de classe é legal, mas copiar seu trabalho, não. Por isso, cópias de trabalhos **não** serão toleradas, resultando em nota **zero** para **todos** os envolvidos.

Entrega

- Os arquivos devem ser entregues em um arquivo **zip**.
- **NÃO** serão aceitos envios por e-mail. O arquivo deve ser enviado **apenas** pelo *SIGAA* através da opção *Tarefas* até a data divulgada no sistema.