ED1 2024/2: Lista de Exercícios do Cap. 9
Estrutura de Dados I
Professor(es): Abrantes Araújo Silva Filho Entrega: 2024-11-12 23:59:59

Exercícios referentes ao Capítulo 9: Eficiência e TADs

1. Exercícios de programação:

   (a) Para entender como a implementação interna de um tipo abstrato de dado (TAD) pode ter grande influência na complexidade computacional de seus programas, você aprendeu a implementar um **editor de *buffer*** simples cujo código fonte está no arquivo "`meu_editor.c`". Esse editor foi criado tendo como base um tipo abstrato de dado chamado "`bufferTAD`" cuja interface está definida no arquivo "`buffer.h`". Além disso a implementação do "`bufferTAD`" foi feita através de três métodos diferentes:

   - Com arrays (arquivo "`arraybuff.c`");
   - Com pilhas (arquivo "`stackbuff.c`"); e
   - Com uma lista simplesmente encadeada (arquivo "`lsebuff.c`").

   Você também aprendeu que poderíamos alterar completamente a implementação da interface `buffer.h` (usando arrays, pilhas ou listas simplesmente encadeadas) sem ter que alterar 1 única linha de código na interface em si, nem no programa cliente `meu_editor.c`.

   Ao comparar a eficiência computacional das três diferentes implementações para a interface `buffer.h` vimos o seguinte:

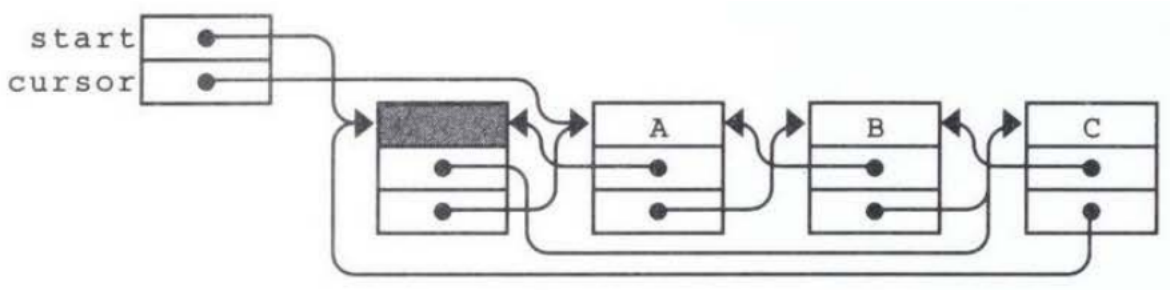
   Tabela 1: Complexidade das operações da interface `buffer.h` quando implementada com Array, Stack e Lista (simplesmente encadeada)

Função	**Array**	**Stack**	**Lista**
mover_cursor_para_frente	$O(1)$	$O(1)$	$O(1)$
mover_cursor_para_tras	$O(1)$	$O(1)$	$O(N)$
mover_cursor_para_inicio	$O(1)$	$O(N)$	$O(1)$
mover_cursor_para_final	$O(1)$	$O(N)$	$O(N)$
inserir_caractere	$O(N)$	$O(1)$	$O(1)$
apagar_caractere	$O(N)$	$O(1)$	$O(1)$

   Você aprendeu que alterar o modo como uma interface é implementada pode afetar drasticamente a eficiência de seu programa. Infelizmente, para todas as implementações que testamos acima, não conseguimos que nosso editor de *buffer* realizasse todas as operações em tempo constante, $O(1)$. Mas existe uma maneira! Temos que criar uma outra implementação da interface `buffer.h`, dessa vez utilizando uma **lista duplamente encadeada** (mais especificamente uma lista circular duplamente encadeada)!

   Estude como a interface `buffer.h` funciona e como a implementação com uma lista simplesmente encadeada funciona (`lsebuff.c`). Note que estamos utilizando a estratégia de ter uma célula "boba" (*dummy cell*) no início da lista para simplificar as operações. Depois que você entender como a implementaçao funciona, sua tarefa é criar uma nova implementação, agora utilizando uma **lista circular duplamente encadeada**. A figura a seguir ilustra como sua implementação deve funcionar.

Figura 1: Implementação da interface `buffer.h` com uma lista circular duplamente encadeada



Na figura acima, os ponteiros em *start* e *cursor* estão dentro do *buffer*, dentro de uma variável do tipo `bufferTAD`, conforme você estudou na implementação com uma lista simplesmente encadeada. A diferença agora é que cada célula da lista terá dois ponteiros: o ponteiro `proximo` aponta para a próxima célula na lista, e o ponteiro `anterior` aponta para a célula anterior na lista. Note que a lista é uma lista circular, pois o ponteiro `proximo` da última célula aponta para a *dummy cell*, e o ponteiro `anterior` da *dummy cell* aponta para a última célula da lista (a implementação da lista de forma circular permitirá que todas as operações de nosso editor de *buffer* sejam feitas em tempo constante).

Você deve criar a implementação em um arquivo com o nome `lcdebuff.c`. Atenção: você **não deve alterar a interface** e também **não deve alterar o programa cliente**! Tudo o que você deve fazer é programar a nova implementação da interface.

Teste seu programa com essa nova interface e tenha certeza de que o editor de *buffer* está funcionando corretamente. Dica: para saber como gerar o binário da implementação e como compilar o editor usando essa implementação, estude o arquivo `Makefile` que está nos códigos do capítulo.

Quanto terminar, faça a compactação do arquivo `lcdebuff.c` em um arquivo com formato ZIP e envie no Portal do Aluno até a data limite.