IMD0029 – ESTRUTURAS DE DADOS BÁSICAS I

PROF. EIJI ADACHI M. BARBOSA

Lista de Exercícios – Pilha, Fila e Deque

Obs.: Além das questões nesta lista, vejam também as questões das provas passadas.

Questão 1. Em sala de aula, discutimos que os TADs Pilha, Fila, e Deque podem ser considerados casos específicos do TAD Sequência. Em outras palavras, ao restringirmos a forma que uma Sequência é manipulada, teremos o comportamento esperado de uma Pilha, Fila, ou Deque. Também vimos em sala de aula duas formas distintas de implementar o TAD Sequência: uma baseada em arrays, e outra baseada em listas encadeadas. Nesta questão, implemente os TADs Pilha, Fila, e Deque com base em arrays e em listas encadeadas. Todas as operações listadas abaixo ter implementação com complexidade $\Theta(1)$.

a) `Pilha – Operações Push e Pop`
b) `Fila – Operações Queue e Dequeue`
c) `Deque – PushBack, PushFront, PopBack, PopFront`

Dica: Para implementar as operações da Fila e Deque em tempo constante, lembre-se de como discutimos em sala de aula a implementação de uma Sequência com base em array usando a estratégia de ocupação circular do array.

Questão 2. Com base em arrays, projete uma estrutura de dados chamada DoubleStack (pilha dupla). Uma DoubleStack consiste numa estrutura que mantém duas pilhas num mesmo array: uma pilha que se inicia na parte inferior do array e cresce para a parte superior, e uma outra pilha que se inicia na parte superior e cresce para a parte inferior. Uma DoubleStack deve prover as seguintes operações:

- `Bool PushDown( Elemento ) - Insere um elemento na pilha inferior`
- `Bool PushUp( Elemento ) - Insere um elemento na pilha superior`
- `Elemento PopDown( ) - Remove um elemento da pilha inferior, retornando-o`
- `Elemento PopUp( ) - Remove um elemento da pilha superior, retornando-o`
- `Bool DoublePush( Elemento ) - Insere o elemento nas duas pilhas`
- `(Elemento, Elemento) DoublePop( ) - Remove um elemento da pilha inferior e um elemento da pilha superior, retornando ambos elementos como um par (a,b)`

Seu projeto deverá checar os casos em que a pilha dupla já está cheia e que já está vazia, retornando códigos de erro `PILHA_CHEIA` e `PILHA_VAZIA`.

Questão 3: Suponha que você já possui uma estrutura Pilha implementada corretamente. Reuse esta estrutura Pilha para implementar uma estrutura Fila. Em outras palavras, implemente uma estrutura Fila usando internamente uma ou mais estruturas Pilha. Lembre-se que as operações da Fila devem obedecer a estratégia FIFO.

Questão 4: Uma das desvantagens de se implementar os TADs Pilha e Fila com base em arrays é a falta de flexibilidade quanto a memória usada. Em muitos casos, podemos subutilizar o TAD ao alocarmos um espaço de memória muito maior do que aquele que usamos na prática. Por outro lado, também podemos utilizar o TAD até o ponto em que todos as posições do seu array interno estão preenchidos. Uma possibilidade para contarmos esta situação é implementarmos o TAD com base em arrays seguindo uma estratégia "elástica", ou "redimensionamento dinâmico". Tal estratégia consiste no seguinte: após inserirmos um elemento no TAD, verificamos se o seu array interno ficou "muito cheio"; se sim, realocamos um novo array com o dobro do tamanho e copiamos todos os elementos para este novo array; se não, deixamos o array interno como está. Similarmente, após removermos um elemento do TAD, verificamos se o seu array interno ficou "quase vazio"; se sim, realocamos um novo array com a metade do tamanho, copiando todos os elementos para este novo array; se não, deixamos o array interno como está. As definições de "muito cheio" e "quase vazio" podem ser definidas como: "muito cheio" quer dizer que mais de 75% do array interno já está ocupado, enquanto "quase vazio" quer dizer que menos de 25% do array interno está ocupado. Neste contexto, implemente os TADs Pilha (`Push` e `Pop`) e Fila (`Queue` e `Dequeue`) com base em array e seguindo a estratégia elástica descrita acima.

Dica: Projete funções para redimensionar o array interno dos TADs de modo a serem reutilizáveis tanto no contexto da Pilha quanto da Fila.

Dica: Lembre-se que após realocar os elementos do array interno, as propriedades FIFO e FILO devem permanecer..

Questão 5: Palíndromo é a frase ou palavra que mantém o mesmo sentido quando lida de trás pra frente. São exemplos de palíndromo: "arara", "osso", "reler", "somos", e "amor à roma". Podemos também considerar palíndromos algumas combinações de palavras em que desprezamos pontuações, acentos e espaços em branco, como por exemplo: "ralo do dólar", "até o poeta", "após a sopa", etc. Neste contexto:

a) Use uma Pilha para projetar uma função que recebe uma string como entrada e verifica se ela é ou não um palíndromo.
b) Use um Deque para projetar uma função que recebe uma string como entrada e verifica se ela é ou não um palíndromo.

Obs.: Para implementar as funções acima, considere que os caracteres da string de entrada não terão acentos, mas poderão existir caracteres "em branco".

Questão 6: Uma aplicação bastante comum em analisadores de texto, como compiladores e interpretadores, é verificar se uma dada sequência de caracteres especiais está bem formada. Por exemplo, a sequência de parênteses e colchetes a seguir é bem formada [( ) (( ))] enquanto a sequência [( ]) não é bem formada. A primeira sequência é bem formada pois o abre-fecha de parênteses está correto, enquanto na segunda sequência o colchete foi fechado antes de se fechar o parênteses. Outros exemplos de sequências que não são bem

formadas: "] [", "((((()))", etc. Este mesmo tipo de verificação pode ser feito para analisar o abrir e fechar de tags HTML, por exemplo. Nesta questão, faça uma função que recebe uma string representando uma sequência de parênteses e colchetes e verifica se esta sequência é bem formada ou não.

Dica: Use uma pilha para resolver esta questão. Percorra a string da esquerda para direita, utilizando a pilha como uma espécie de "memória" do que já foi "visto" na string de entrada.

Questão 7:[1] Na notação usual de expressões aritméticas, os operadores são escritos entre os operandos; por isso, a notação é chamada infixa. Na notação pós-fixa, ou notação polonesa inversa, os operadores são escritos depois dos operandos. Eis alguns exemplos de expressões infixas e correspondentes expressões pós-fixas:

Infixa	Pós-fixa
(A+B*C)	ABC*+
(A*(B+C)/D-E)	ABC+*D/E-
(A+B*(C-D*(E-F)-G*H)-I*3)	ABCDEF-*-GH*-*+I3*-
(A+B*C/D*E-F)	ABC*D/E*+F-
(A+B+C*D-E*F*G)	AB+CD*+EF*G*-
(A+(B-(C+(D-(E+F)))))	ABCDEF+-+-+
(A*(B+(C*(D+(E*(F+G))))))	ABCDEFG+*+*+*

Note que os operandos (A, B, C, etc.) aparecem na mesma ordem na expressão infixa e na correspondente expressão pós-fixa. Note também que a notação pós-fixa dispensa parênteses e regras de precedência entre operadores (como a precedência de * sobre + por exemplo), que são indispensáveis na notação infixa. Nosso problema: traduzir para notação pós-fixa a expressão infixa armazenada em uma string `inf`. Para simplificar nossa vida, vamos supor que

- a expressão `inf` é válida e contém apenas letras, parênteses esquerdos, parênteses direitos, e símbolos para as quatro operações aritméticas,
- todas as operações (em particular - e +) têm dois operandos,
- os nomes das variáveis têm apenas uma letra cada,
- a expressão `inf` está embrulhada em um par de parênteses (ou seja, o primeiro caractere é '(' e os dois últimos são ')' e '\0').

Neste contexto, implemente uma função que recebe como entrada uma string `inf` representando uma expressão aritmética em notação infixa e retorna uma outra string representando a expressão equivalente em notação pós-fixa.

Dica: Use uma pilha para resolver esta questão. Percorra a string da esquerda para direita, utilizando a pilha como uma espécie de "memória" do que já foi "visto" na string de entrada.

Questão 8: Agora implemente uma função que recebe como entrada uma string representando uma expressão aritmética em notação pós-fixa e retorna a expressão equivalente em notação infixa. Ou seja, uma função que faz o contrário da função da questão anterior.

[1] Questão extraída e adaptada de [1]

Questão 9: Implemente uma função que recebe como entrada uma string representando uma expressão aritmética em notação pós-fixa e retorna o seu valor numérico correspondente. Considere que a string de entrada:

- é válida e contém apenas inteiros, parênteses esquerdos, parênteses direitos, e símbolos para as quatro operações aritméticas,
- os elementos básicos da expressão (números inteiros, parênteses e símbolos das operações) são separados por um espaço em branco.

Questão 10: Use uma pilha para inverter a ordem dos nós de uma lista simplesmente encadeada.

REFERÊNCIAS

[1] https://www.ime.usp.br/~pf/algoritmos/aulas/pilha.html