Mauricio Konrath

Matricula: 20203635

Estrutura de Dados – Prova Teórica 1

1 –

A) Desenhe a pilha construída para a seguinte chamada:

B) Qual(is) a(s) vantagem(ns) em se trocar ArrayStack por ArrayList nessa implementação? Justifique.

O ArrayStack é uma matriz que usa memória contígua, ou seja, toda a pilha deve receber um bloco de memória. Dessa forma, para o ArrayStack, precisamos fazer um novo array e copiar tudo nele para redimensiona-lo. O que pode vir a ser um problema, visto que, muitas vezes não temos uma noção precisa do tamanho que a pilha pode vir a ficar. Já a LinkedList consiste em nós individuais vinculados por ponteiros e não tem esse problema, podemos resolver criando um novo nó e vinculando a lista como era antes. Tal troca, de ArrayStack para LinkedList afetaria tanto a memória quanto o desempenho.

C) Expresse a complexidade (assintótica) desse algoritmo em termos de N, α e β.

No melhor caso é big O(N), pois N, α e β são lineares e já no pior caso: big Ω (N * (α + β)).

2- Insere no fim:

```
void insere_no_fim(T dado):
        Node *novo = new Node (dado)
        if (tamanho == 0)
                novo->suc = novo
                novo->ant = novo
                movel = novo
        end if
        while (pos_movel != tamanho-1)
                movel = movel->proximo()
                pos_movel++
        end while
        primeiro = movel->proximo()
        primeiro->ant = novo
        novo->suc = primeiro
        novo->ant = movel
        movel->suc = novo
        tamanho++
fim
```

3- Analise a quantidade de atribuições de um ponteiro para nó de lista (tipicamente: it = it->next();) ao longo da execução dessa inserção para finalizar o processamento de geração da lista resultante, se for utilizada uma:

a) Fim -> 1 atribuição de ponteiro
   Meio ou Inicio -> 2 atribuições de ponteiro
b) Meio -> 4 atribuições de ponteiro
   Extremo -> 3 atribuições de ponteiro
c) Meio -> 5 atribuições de ponteiro
   Extremo -> 3 atribuições de ponteiro

4 –

a) Escreva a relação de recorrência que representa o custo total T(n) de Algoritmo dada pela soma de custos de todas as linhas (não é preciso resolve-la, ou seja, não é preciso encontrar a complexidade).

$$n/p * (T(n/p) + 2 * n + 3 + k)$$

b) Compare duas versões da implementação em termos de tempo computacional

Implementação 1: $n/3 * (T(n/p) + 2 * n + 3 + k)$
Implementação 2: $n/8 * (T(n/p) + 2 * n + 3 + k)$
R. A Implementação 2 é 2.6x mais rápida.