Universidade Braz Cubas

Projeto e Análise de Algoritmos

Carlos Alberto Gonçalves Abreu – 126749 – 11º EN

Prof. Ângelo Pássaro

Introdução

Examinando algoritmos de inserção e remoção em Listas Ordenadas e Encadeadas.

Análise do Problema 3

A análise de um algoritmo baseia-se onde ele se torna crítico, do ponto de vista do tempo de execução, que no caso da inserção/remoção, é na própria rotina que a faz, sendo referenciada várias vezes, pois em uma lista ordenada, os valores são inseridos de forma que a lista fique ordenada, movendo os dados posteriores ao elemento a ser inserido, e a remoção, os dados posteriores ao removido, é deslocado para a posição do elemento removido.

Listas Ordenadas

As listas ordenadas são caracterizadas por serem de implementação limitada, pois utiliza-se de alocação estática, ou seja, o tamanho da lista pode ficar sub ou superdimensionada. Pode ser comparada a um vetor de registros, onde o próximo item ficará em um endereço contíguo da memória.

Abaixo temos o algoritmo de inserção para Listas Ordenadas:

```
Procedimento  Inserir(Lista:tipo lista ordenada;Elemento:tipo do elemento)
Variáveis
  posição, i: Inteiro sem sinal
Inicio
   Se Lista estiver cheia Então
     Mostra mensagem 'Lista Cheia'
   Senão
       Inicio
              Para i =1 até ultimo elemento da Lista faça
                     Se informação da Lista >= Elemento Então
                            Inicio
                                   Posição = i
                                   Sai do laço
                            Fim
                     Senão
                            Posição=0
              Fim Para
              Se Posição = 0 Então
                     Posição = Ultima Posição da Lista + 1
              Senão
                     Para i = Ultima Posição da Lista +1 até Posição faça
                            Informação da Lista = Informação Anterior da Lista
   Fim Senão
   Informação da Posição da Lista = Elemento
   Ultimo Elemento da Lista + 1
Fim Inserir
```

O texto em negrito, mostra que esta é a parte crítica do algoritmo de inserção, onde irá fazer uma pesquisa do valor a ser inserido aos que já estão na lista, e posteriormente, fazer o deslocamento dos itens.

Na pesquisa, temos no pior caso, que seria um elemento a ser inserido maior que o maior valor da lista, N-1; no caso médio, teremos (N-1)/2.

No deslocamento, temos no pior caso, que seria o elemento menor que o menor valor da lista, N; no caso médio, teremos (N/2).

Abaixo, temos o algoritmo de remoção para Listas Ordenadas:

```
Procedimento Remover(Lista: tipo Lista Ordenada;Elemento:tipo do elemento)
Variáveis
  posição,i: Inteiro sem sinal
Inicio
        Se Lista estiver Vazia Então
                Mostra mensagem 'Lista Vazia'
        Senão
                Inicio
                        Para i =1 até ultimo elemento da Lista faça
                                Se informação da Lista >= Elemento Então
                                        Inicio
                                                Posição = i
                                                Sai do laço
                                        Fim
                                Senão
                                        Posição=0
                        Fim Para
                        Se Posição = 0 Então
                                Mostra mensagem 'Elemento não encontrado'
                        Senão
                            Inicio
                                Para i = Posição até penúltimo elemento da Lista faça
                                    Informação da Lista = Informação posterior da Lista
                                Ultima Posição da Lista  - 1
                            Fim Senão
                Fim Senão
Fim Remover
```

Assim como na inserção, temos inicialmente uma pesquisa, para certificar que o elemento a ser removido existe, o que nos dá no pior caso, que seria um elemento que não existe na lista, ou seja, N; no caso médio, teremos (N / 2).

Caso encontre o elemento a ser removido, será feito um deslocamento, para voltar a preencher o espaço deixado pelo elemento retirado. Com isso temos no pior caso, N; no caso médio, teremos (N / 2).

Listas Encadeadas

As listas encadeadas são a forma mais ‘elegante’ de utilizar listas em implementações de algoritmos, pois diferentemente das listas ordenadas, elas utilizam-se de ponteiros para sua implementação, não havendo necessidade de superdimensionar a lista, já que estas são alocadas dinamicamente na memória, sendo limitada pelo tamanho disponível de memória do sistema.

Os nós da lista, não ficam necessariamente contíguos na memória, com isso, só será alocada memória livre, independente da posição física da mesma.

Abaixo, temos a inserção em uma Lista Encadeada:

```
Procedimento Inserir(Lista: tipo Lista Encadeada;Elemento:tipo do Elemento);
Variáveis
  N,P: tipo Lista Encadeada
Inicio
        Cria um novo nó N
        Conteúdo da informação de N = Elemento
        Se Lista estiver Vazia ou Elemento for menor que Conteúdo da informação
           de Lista Então
                Inicio
                        Conteúdo do próximo nó em N  = Endereço de Lista
                        Conteúdo de Lista = Conteúdo de N
                Fim
        Senão
                Inicio
                        Conteúdo de P = Conteúdo de Lista
                        Enquanto Conteúdo do próximo nó em P diferente de nulo
                           faça
                                Endereço de P = Conteúdo do próximo nó em P
                        Conteúdo do próximo nó em N = Conteúdo do próximo nó
                           em P
                        Conteúdo do próximo nó em P = Endereço de N
                Fim
Fim Inserir
```

Na inserção, teremos , N-1 elementos para percorrer, pois as mudanças envolve ponteiros(endereços de memória), uma vez encontrada o final da lista(ponteiro nulo), basta fazer a atribuição do endereço do novo nó ao final da lista existente.

Abaixo, temos o algoritmo para remoção em Listas Encadeadas:

```
Função Remover(Lista: tipo Lista Encadeada;Elemento:tipo do Elemento):booleano
Variáveis
  P,Q: tipo Lista Encadeada
Inicio
      Se Lista estiver Vazia ou Elemento for menor do que  Conteúdo da
          Informação da Lista  Então
            Retorna Falso
      Senão
            Se Elemento = Conteúdo da informação da Lista Então
                  Inicio
                        Conteúdo de P = Conteúdo de Lista
                        Endereço da Lista = Conteúdo do próximo
                              endereço em Lista
                        Libera memória alocada por P
                        Retorna Verdadeiro
                  Fim
                              Senão
                  Inicio
                        Conteúdo de P = Conteúdo de Lista
                        Enquanto Conteúdo do próximo endereço em P <>
                              nulo faça
                           Endereço de P = Conteúdo do próximo endereço
                              em P
                        Se Conteúdo do próximo endereço em P Então
                              Inicio
                                    Endereço de Q = Conteúdo do próximo
                                          endereço em P
                                    Conteúdo do próximo endereço em P =
                                       Conteúdo do próximo endereço em
                                       Q
                                    Libera memória alocada por Q
                                    Retorna Verdadeiro
                              Fim
                        Senão
                              Retorna Falso
                  Fim Senão
Fim Remover
```

Na remoção, teremos a mesma situação da inclusão, ou seja N-1 elementos.

Conclusão

Neste documento, vimos que as listas encadeadas são menos eficientes em termos de eficiência, devido ao fato de se percorrer toda a lista para encontrar o final e além de ser um pouco mais difícil de implementar em relação ao de Listas Ordenadas.

Bibliografia

LAFORE, Robert. Aprenda em 24 Horas Algoritmos; tradução de Adriana Kramer. Rio de Janeiro: Campus, 1999. 501p. Bibliografia: p. 137-154, 325-377. ISBN 85-352-0475-X.

ZIVIANI, Nivio. Projeto de Algoritmos. 4. ed. São Paulo: Pioneira, 1999. 267p. Bibliografia: p. 35-41, 126-127. ISBN 85-221.

SEVERINO, Antônio Joaquim. Metodologia do Trabalho Científico. 21. ed. São Paulo: Cortez, 2000. 279p. Bibliografia: p. 86-132. ISBN 85-249-0050-4.

RUIZ, João Álvaro. Metodologia Científica. 4. ed. São Paulo: Atlas, 1996. 177p. Bibliografia: p. 74-86. ISBN 85-224-1465-3.

MARTINS, Gilberto de Andrade. Manual para Elaboração de Monografias e Dissertações. 2. ed. São Paulo: Atlas. 115p. Bibliografia: p. 49-62. ISBN 85-224-1087-9