I

void removeUltOcorrencia(DLista \*lst, char \*s)

{

Ligacao \*x;

for (x = lst->ultimo; x != NULL; x = x->anterior)

{

if (strcmp(x->text, s) == 0)

{

break; /\*parar quando x->text == s, ou seja temos de remover x\*/

}

}

/\* se elemento a remover for a head \*/

if (x->anterior == NULL)

{

lst->primeiro = x->proximo;

lst->primeiro->anterior = NULL;

}

else if (x->proximo == NULL)

{

/\* se elemento a remover for o ultimo \*/

lst->ultimo = x->anterior;

lst->ultimo->proximo = NULL;

}

else

{

/\*se elemento a remover for no meio\*/

x->anterior->proximo = x->proximo;

x->proximo->anterior = x->anterior;

}

free(x->text);

free(x);

}

Análise Pior caso:

O(n) -> ter de percorrer a doubly linked list toda, n = tamanho da doubly linked list

-----------------------------------------------------

II

Após remoção do elemento com chave 14:

Indice | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

Elementos| 0 | 12 | 13 | 25,36 | 15,26 | | 28,39 | | | 20 | |

1a colisao: 15 colide com 26 no index 4 da tabela

2a colisao: 28 colide com 39 no index 6 da tabela

3a colisao: 25 colide com 36 no index 3 da tabela

total colisoes: 3

Para remover o elemento com chave k, na tabela, temos de fazer k mod M, ou seja

neste caso k mod 11, o resultado dessa operação dá-nos o index da tabela onde devemos

procurar k. Depois temos de percorrer uma linkedlist, cuja head, é apontada pelo ponteiro

presente no index da tabela que nos deu a operação k mod 11, até achar o elemento k, e posteriormente

removê-lo da lista, tal e qual se remove um elemento numa linkedlist.