I

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

{

Ligacao \*novo = (Ligacao \*)malloc(sizeof(Ligacao));

Ligacao \*x;

novo->text = (char\*)malloc(sizeof(char)\*strlen(s));

strcpy(novo->text, s);

novo->proximo = NULL;

novo->anterior = lst->ultimo;

/\*lst vazia\*/

if (lst->primeiro == NULL)

{

lst->primeiro = novo;

lst->ultimo = novo;

return;

}

for (x = lst->primeiro; x != NULL; x = x->proximo)

{

/\*parar quando x->text > s, ou seja s tem de ser colocado antes de x->text\*/

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

{

break;

}

}

novo->anterior = x->anterior;

novo->proximo = x;

x->anterior->proximo = novo;

x->anterior = novo;

}

Análise Pior caso:

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

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

II

Indice | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

Elementos| 26,39,13,0 | X | 28,15| X | 30 | X | X | X | X | X | 36 | X | 12,25 |

A ordem dos elementos em cada posicao da tabela esta pela ordem inversa à insercao da tabela,

uma vez que é a maneira de resolver as colisões na resolucao por encadeamento externo,

isto é, quando ocorre colisao, o elemento é adicionado ao inicio da lista.