Lista 5 - ED

Wellington Silva
Tiago Barradas
Escola de Matemática Aplicada, Fundação Getulio Vargas, Brasil.

May 31, 2023

Considerações: Submissão do código em um aquivo `.cpp` ou um zip com um `.cpp` por questão. Por simplificação vamos tratar **lista duplamente encadeada** só como LDE. Nesta lista, diferentemente da anterior, não estamos disponibilizando algum tipo de teste, mas seria interessante colocarem alguns testes junto ao código enviado.

Considere que queremos trabalhar com uma estrutura um pouco diferente do que inteiros. Temos uma língua esquisita composta pelor 2 tipos de letras: vogais "A", "U", "Y", e consoantes "B", "J", "S", "R" e "T". Uma palavra nessa língua é qualquer sequencia de 2 a 8 letras alternando entre vogais e consoantes (por exemplo, "ABUJY", "TURA" ou "SURATURY").

Considera uma LDE para guardar palavras dessa língua, vamos usar o seguinte nó para os problemas a seguir:

```cpp
struct Node {
    std::string data;
    Node *next;
    Node *prev;
};
```

Problema 1: (Funções úteis) Implemente as seguintes funções:

(a) (is_valid) Dado uma palavra, diga se essa palavra é válida (ou seja, segue as regras da língua esquisita).

(b) (insert) Dado uma palavra e uma LDE, adicione a nova palavra no fim da lista se ela for válida, ou printe um erro caso não for;

(c) (find) Dado uma palavra e uma LDE, verifique se a palavra está ou não na lista;

(d) (delete) Dado uma palavra e uma LDE, delete a palavra da lista;

(e) (print) Dado uma LDE, printe a lista elemento a elemento;

(f) (comparison) Dado duas palavras verifique se uma precede a outra na ordem alfabética (sendo $A > U > Y > B > J > S > R > T$ a ordem de consideração).

Problema 2: (sort) Na lista anterior implementamos o Bubble sort, que tem uma complexidade $O(n^2)$. Agora, implemente outro algoritmo de sort, o Merge sort[1], que tem complexidade $O(n \log n)$, para uma LDE com as palavras da língua que introduzimos no início.

Problema 3: (vai e volta) Faça uma função que primeiro percorre a LDE a partir da head da lista, encontrando a palavra que mais se repete e o número de repetições. Daí, de posse dessa palavra, você deve voltar a partir da tail, removendo da LDE as ocorrências dessa palavra até não haver mais ocorrências (note que, é até não haver ocorrências, lembre-se do contador).

Problema 4: (comparação) Na questão 2 mencionamos que o Merge sort tem uma complexidade menor, mas como isso interfere? Primeiro adapte o Bubble sort da lista anterior para funcionar com uma LDE que armazena palavras da língua estranha. Depois crie algumas LDE não ordenadas com diversos tamanhos (como 100, 1000, 10000) e verifique com ambos os sorts os tempos de execução. Certifique-se que as palavras sendo inseridas nas listas sejam o mais aleatórias possível, tendo um número de algarismos variável (entre 4 a 7 letras, por exemplo).

Deixe os resultados comentados no código, com tempo.

[1] https://en.wikipedia.org/wiki/Merge_sort