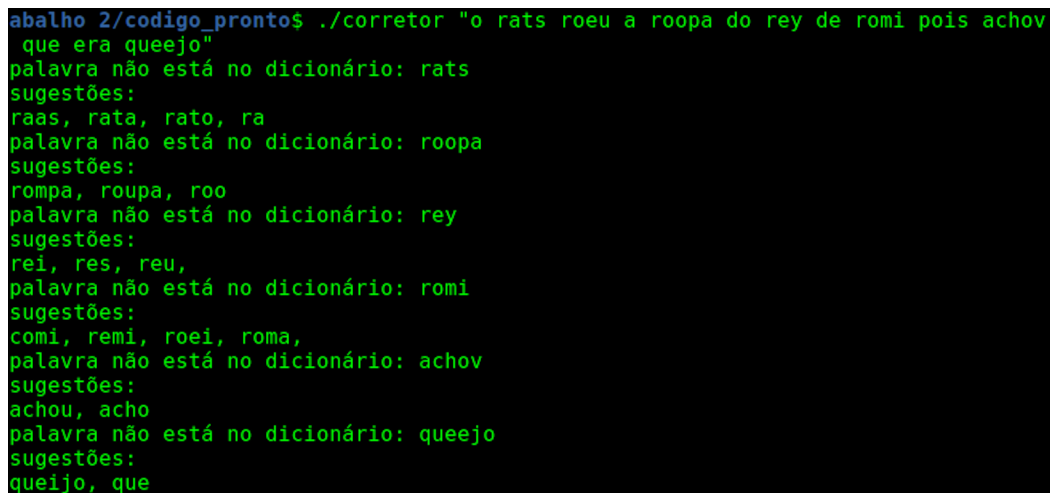


Trabalho Prático 2: Corretor Ortográfico

Prof. Dr. Juliano Henrique Foleis

Introdução

O corretor ortográfico é uma funcionalidade presente em muitos aplicativos do dia-a-dia. Neste trabalho você vai implementar um corretor ortográfico. Seu corretor deve receber um texto sem acentos em língua portuguesa e deve analisar cada palavra. Caso uma palavra não esteja no dicionário, seu programa deve fornecer sugestões de palavras do dicionário. O dicionário fornecido contém 261797 palavras da língua portuguesa. A Figura 1 mostra um exemplo:



```
Trabalho 2/codigo_pronto$ ./corretor "o rats roeu a roopa do rey de romi pois achov
que era queejo"
palavra não está no dicionário: rats
sugestões:
raas, rata, rato, ra
palavra não está no dicionário: roopa
sugestões:
rompa, roupa, roo
palavra não está no dicionário: rey
sugestões:
rei, res, reu,
palavra não está no dicionário: romi
sugestões:
comi, remi, roei, roma,
palavra não está no dicionário: achov
sugestões:
achou, acho
palavra não está no dicionário: queejo
sugestões:
queijo, que
```

Figure 1: Exemplo de Entrada e Saída do Corretor Ortográfico

Como a lista de palavras é razoavelmente grande, é necessário usar uma estrutura de dados apropriada para fazer a busca das palavras no dicionário. As *Tries* podem ser usadas para esse propósito. Além disto, as *Tries* também são adequadas para implementar operações que facilitem a sugestão de palavras do dicionário. Em específico, neste trabalho você irá implementar três operações de busca aproximada para este fim:

- Busca de chaves que contém um determinado prefixo;
- Busca de chaves que casam com um padrão contendo coringas; e
- Busca de uma chave com o maior prefixo comum com outra chave.

Estas três operações são explicadas neste [vídeo](#). A seção “Operações especiais: `keysWithPrefix()`, `longestPrefixOf()`, etc.” deste [link](#) também discute algumas dessas operações.

Especificação

1. Modifique a Trie estudada em sala para que cada nó tenha apenas 26 filhos, um para cada letra do alfabeto. Como as palavras da língua portuguesa tem acentos, é necessário tirá-los para representar todas as palavras em uma trie de grau 26. Os acentos já estão removidos da lista de palavras fornecida. Durante a inserção “normalize” as palavras para apenas letras maiúsculas ou minúsculas.

2. Implemente uma função *TRIE* ConstruirDicionario(char* arq_lista_palavras)*, que recebe como entrada o caminho para o arquivo com a lista de palavras e devolva uma Trie com todas as palavras. Esta assinatura da função é apenas uma sugestão. Fique a vontade para adicionar mais parametros caso necessário.
3. Implemente a função *TRIE_ChavesComPrefixo(Trie* T, char* prefix)* que retorne a lista de chaves da Trie que possuem a string *prefix* como prefixo. A lista de chaves deve ser uma implementação adequada de uma TAD lista, com as operações que julgar necessário.
4. Implemente a função *TRIE_ChavesQueCasam(Trie* T, char* padrao, int n_extras)* que retorne a lista de chaves que casam com o padrão *padrao*. Considere o caractere “*” como coringa com até *n_extras* caracteres extras após o coringa. Por exemplo, para *TRIE_ChavesQueCasam(Trie* T, char* padrao, int n_extras)* com *padrao = "ac"* e *n_extras = 2*, sua função deve retornar todos as chaves que casam com o padrão "ac*". A lista de chaves deve ser uma implementação adequada de uma TAD lista, com as operações que julgar necessário.
5. Implemente a função *char* TRIE_ChaveMaiorPrefixoDe(Trie* T, char* s)*, que retorna a chave da trie que casa com o maior prefixo possível de *s*.
6. Implemente a função *void CorrigirOrtografia(Trie* dicionario, char* texto)*, que analisa a ortografia de cada palavra da string *texto* e. Caso a palavra não esteja no dicionário, sugira outras palavras de acordo com as regras abaixo. Sugira uma regra também! Você pode criar outras regras também! Entretanto as regras abaixo devem estar implementadas. Use as funções implementadas nos itens 3, 4 e 5 acima de acordo com a necessidade de cada regra.

Seja ρ uma palavra ortograficamente incorreta encontrada no texto.

- Regra 1: Retorne palavras que casem com ρ substituindo uma letra de cada vez pelo coringa. Por exemplo, se $\rho = \text{"ratp"}$, sugira palavras que casem com “*atp”, “r*tp”, “ra*p” e “rat*”.
 - Regra 2: Esta regra só deve ser usadas para encontrar sugestões para palavras com mais que 5 letras. Sugira as palavras que são os prefixos de ρ que vão até a posição $n-3$ e $n-2$ de ρ , tal que n é o comprimento de ρ . Por exemplo, se $\rho = \text{"saladq"}$, sugira palavras com os prefixos “salad” e “sala”.
 - Regra 3: Sugira a palavra que case com o maior prefixo possível de ρ .
 - Regra 4: Invente sua regra! Ela deve usar uma combinação de pelo menos 2 das três regras acima. Explique seu raciocínio e a razão pela qual você acha que essa regra encontrará boas sugestões.
7. Usando as funções implementadas nos itens 2 e 6, escrever o programa principal do corretor ortográfico. Este programa recebe via linha de comando o nome de um arquivo contendo o texto a ser analisado. Leve em consideração:
- Seu programa principal deve executar sem interação com o usuário, exceto por sua invocação pela linha de comando. As sugestões devem ser impressas para cada palavra que estiver ortograficamente incorreta.
 - O texto da entrada pode conter pontuação, portanto é necessário descartar a pontuação antes de verificar a ortografia. As palavras da entrada estarão sem acentos.
 - Para cada palavra incorreta, não sugira palavras repetidas que sejam encontradas por mais que uma regra! Você pode usar uma segunda Trie para evitar as repetições.
 - As sugestões devem ser impressas na ordem crescente do número de letras na palavra e em ordem alfabética. **DICA:** A saída dos algoritmos dos itens 2, 3 e 4 já garantem que as chaves são inseridas na lista em ordem alfabética. Se você usar um algoritmo de ordenação estável, basta ordenar pelo tamanho das palavras!

Análise

Textos de teste são fornecidos para você testar o seu corretor ortográfico. Preencha a tabela abaixo com informações sobre a análise de cada texto usado para testar seu programa.

	Total de palavras	Palavras incorretas	Número médio de sugestões por palavra incorreta	Tempo de Execução (s)
casmurro1.txt				
casmurro2.txt				
casmurro3.txt				
casmurro4.txt				
memposthumas1.txt				
memposthumas2.txt				
memposthumas3.txt				
casmurro_all.txt				
memposthumas_all.txt				
all.txt				

Figure 2: Alguns resultados da execução do corretor ortográfico sobre os testes propostos

Veja as correções sugeridas pelo seu corretor. Em quais situações ele propõe boas sugestões? Em quais situações ele falha? Faça uma breve análise, citando as regras discutidas no item 6 acima.

Regras

- Siga a especificação acima!
- O trabalho pode ser feito (no máximo) em duplas.
- Se o trabalho for feito em duplas, ambos alunos tem que escrever parte do código. Em outras palavras, não é pra um aluno fazer o código e o outro fazer o relatório.
- Os arquivos de teste não devem ser alterados de nenhuma forma antes de serem usados no programa.
- O trabalho deve ser implementado em linguagem C, usando modularização e Makefile para compilar.
- As listas de chaves retornadas pelas funções *TRIE_ChavesComPrefixo* e *TRIE_ChavesQueCasam* devem ser implementadas como TADs. Outros TADs são encorajados para modularizar adequadamente seu código.
- Trabalhos plagiados uns dos outros ou da internet serão totalmente desconsiderados, sem possibilidade de recuperação.
- Use a biblioteca padrão da linguagem C de entrada e saída para fazer as operações com arquivos (stdio.h).
- Comente o código apenas nos pontos mais importantes. Não é necessário fazer texto :)

Entrega

- Um arquivo .zip (ou .tar.gz) contendo todo o código do trabalho e o pdf de um relatório de implementação e análise, especificado abaixo.
- Um relatório de implementação e análise, com no máximo 10 páginas, descrevendo a implementação desenvolvida. Essa descrição deve indicar como que o código foi modularizado e quais foram os TADs adicionais implementados. Inclua como foi realizada a divisão do trabalho entre os membros da equipe. Inclua também as tabelas obtidas na seção Análise, e discuta os resultados.
- Prazo máximo: **1/12/2020 às 23:55**, via Moodle. Somente um integrante da equipe precisa enviar.

Bom Trabalho!