

Prova prática: Índice remissivo(Python)

Guilherme de Almeida Costa

costaguila@gmail.com

Introdução

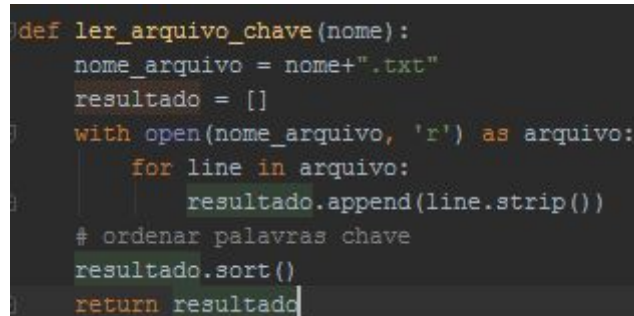
Neste arquivo estão contidas as considerações do autor durante o desenvolvimento do script python para a solução do problema de índice remissivo.

Desenvolvimento

Conforme descrito no problema foi criado dois arquivos de texto:

- Um com as palavras chave;
- Um com o texto;

Estes arquivos serão lidos pelo script para a realização da tarefa. O *script* foi organizado em funções. Existe uma função para cada tarefa da prova, duas funções para a leitura de arquivos e uma função com um menu para ser usado caso o programa seja executado em linha de comando. Vale ressaltar que na função de leitura das palavras chave, a lista de palavras é ordenada imediatamente. Isso foi feito para evitar que seja necessário realizar ordenações em outros momentos.



```
def ler_arquivo_chave(nome):  
    nome_arquivo = nome+".txt"  
    resultado = []  
    with open(nome_arquivo, 'r') as arquivo:  
        for line in arquivo:  
            resultado.append(line.strip())  
    # ordenar palavras chave  
    resultado.sort()  
    return resultado
```

Imagem 1: Função de leitura e ordenação da lista de palavras chave

A função criar índice espera como entrada dois arrays um com as linhas do texto e outro com a lista de palavras chave(1). Para o armazenamento da índice remisso foi escolhida o uso da estrutura de dados ‘dicionário’. O índice remissivo segue a mesma lógica de uma tabela de dispersão(*hashtable*), isto é uma chave está ligada a um ou mais valores. A estrutura dicionário é a implementação de uma tabela de dispersão em python fazendo assim o uso desta estrutura uma escolha natural(2). As inserções no dicionário são feitas com base nas palavras chave, para cada palavra chave é verificado sua existência em cada linha do texto e se ela está dentro daquela linha o dicionário recebe uma nova entrada com aquela chave e com o número da linha(3). Como a lista de palavras chave já está em ordem alfabética não é necessário reordenar o dicionário.

```

def criar_indice(texto, palavrasChave): 1
    result = {} 2
    """
    Recebemos a lista de palavras chave já ordenada. Basta agora verificar para cada linha do texto
    se a palavra existe e adiciona-la ao dicionario. Cada palavra chave já é adicionada em ordem alfabetica.
    """
    for palavra_chave in palavrasChave: 3
        for linha in texto:
            numero_linha_atual = linha.split().pop()
            if palavra_chave in texto[linha]:
                result.setdefault(palavra_chave, []).append(numero_linha_atual)
    print(result)
    print("## Índice criado. ##")
    return result

```

Imagem 2: Função de criação do índice remissivo

As operações de listagem e pesquisa são similares. A principal diferença é que a operação de pesquisa verifica se a chave existe no índice remissivo antes de pesquisar por ela(e recupera apenas os valores da chave). A operação de pesquisa recupera e lista os valores de cada chave no dicionário.

Executando o programa

- Instale o Python;
- Abra a linha de comando do windows ou bash;
- Navegue até a pasta que contém os arquivos(Exemplo `cd C:/Downloads`);
- Execute o comando: **python teste.py** (note que os arquivos `chaves.txt` e `texto.txt` devem estar na mesma pasta que o arquivo `teste.py`);
- **Para sair do programa selecione 4 no menu.**

Referências

TimeComplexity. Disponível em: < <https://wiki.python.org/moin/TimeComplexity> > Acesso em: 16 de junho de 2018