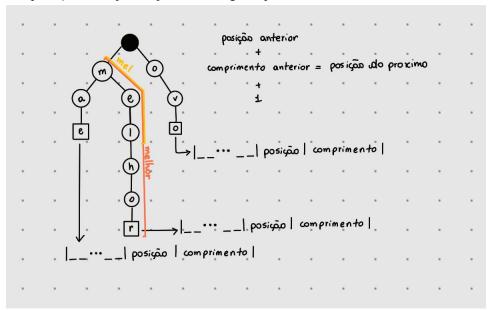
Relatório projeto 2

Disciplina: Estrutura de dados

Alunos: Déborah Raquel Bussolo Ferreira e Enzo Nicolás Spotorno Bieger

As imagens dadas pelo professor no trabalho foram uma das partes significativas para a melhor interpretação do objetivo, junto às imagens que existiam nos links oferecidos.



- ⇒ Tenta abrir o arquivo para fazer a leitura, se não consegue é printado um erro. Se consegue ele lê o arquivo linha por linha, a partir da posição inicial = 0 e o comprimento sendo linha.length(), conseguimos calcular a posição da próxima palavra somando a posição anterior com o comprimento anterior + 1.
- ⇒ Criamos a estrutura do Nó da Trie que tem um vetor com os 26 filhos, a posição e o comprimento.
- ⇔ Com um nó auxiliar percorrendo o vetor de filhos, se o carácter procurado não existe no índice, ele é criado, no fim o auxiliar criado se torna o filho, e assim por diante.
- ⇒ Depois desse processo se chegamos no ultimo caracter, então preenchemos a posição e o comprimento.
- ⇒ Criamos uma função que conta de quantas palavras que esse nó é prefixo. Com base em quantas terminações de palavra esse nó tem conexão.
- ⇒ Para buscar a palavra na Trie, com ajuda de um nó auxiliar, percorremos cada caracter e vemos seu índice, se esse índice não possui filho, retornamos que ele não é um prefixo e não encontramos a palavra. Se possui filho, faz o auxiliar ser o filho, e assim por diante.
- ⇒ Se chegamos ao ultimo caracter dessa palavra, vemos a quantidade de palavras para as quais ela é prefixo. Se a palavra existe e está no dicionário printa o resultado da função num_prefixos(). Também pega a posição e o comprimento para printar.
- ⇒ Pegamos todas as palavras de entrada até chegar ao "0".
- ⇒ Busca cada palavra na Trie usando o método search() que printa todas as informações necessárias de cada uma.

Conclusão:

Tivemos poucas dificuldades na hora de implementar o código, já que tentamos entender muito bem a lógica antes de começar a implementação. Mas uma das nossas principais confusões foi conseguir fazer os prints serem iguais aos que foram requisitados, por exemplo, não printar o número de prefixos quando é zero.

Referências:

Os links dados pelo professor como apoio já foram o suficiente para a elaboração do trabalho e muito úteis.

https://www.ime.usp.br/~pf/estruturas-de-dados/aulas/tries.html

https://towardsdatascience.com/implementing-a-trie-data-structure-in-python-in-less-than-100-lines-of-code-a877ea23c1a1

https://www.geeksforgeeks.org/trie-insert-and-search/