Radix Tree

Michel gomes de souza 1858351
Luis Otavio O. Capelari 1858335
Juliano Cesar Petini 1858319

Um exemplo prático e simples que pode ser aplicado em cima do nosso algoritmo é com um dicionário que contém as seguintes palavras:

dicionario.txt
canela
caneta
canario
cantor
camaro
carro
carra
banana
banal
abacaxi
xicara
dado
xiado
xines
xi

Esse dicionário aplicado em uma árvore ficará como representado na Figura 1.

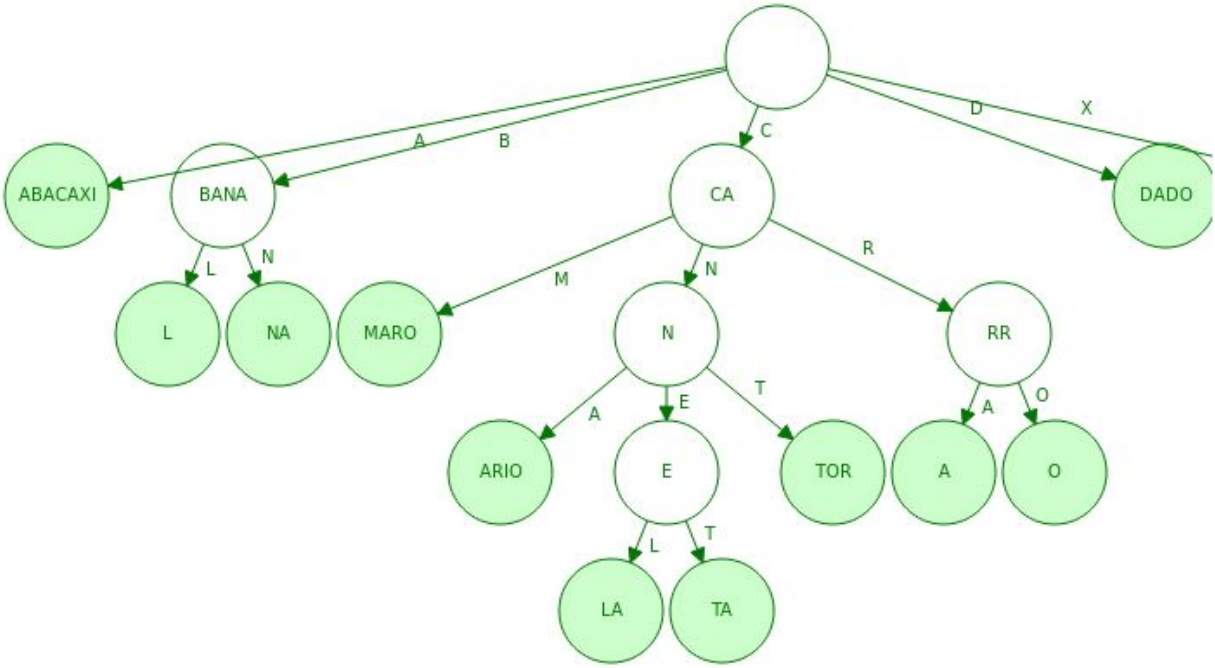

Figura 1 - Radix Tree montada em cima do dicionario.txt

Em nosso algoritmo a representação da Árvore Radix Tree ficará como apresentado na Figura 2, para executar digite no terminal:

```
$ python3 main.py dicionario.txt
```

```
--------PRINT Todos---------------
[abacaxi][xi][dado][camaro][canario][cantor][caneta][canela][carra][carro][banal]
[banana][xiado][xicara][xines]

--------PRINT com Nos---------------

*  False
[ca * False *][bana * False *][abacaxi * True *][xi * True *][dado * True *]
* ca False
[maro * True *][n * False *][rr * False *]
* n False
[ario * True *][e * False *][tor * True *]
* e False
[ta * True *][la * True *]
* rr False
[a * True *][o * True *]
* bana False
[l * True *][na * True *]
* xi True
[ado * True *][cara * True *][nes * True *]
```

Figura 2 - Representação do nosso algoritmo em cima do dicionario.txt

Depois disso você poderá buscar palavras que contêm certos prefixos, exemplo iremos digitar o sufixo “c” e retornará todas as palavras que começam com “c”, como Demonstrado na Figura 3.

```
-Digite um Prefixo para Buscar? (Para Sair Digite 0)
c

-------BUSCANDO------
camaro
canario
caneta
canela
cantor
carra
carro
```

Figura 3 - Resultado da busca pelo prefixo “c”

O mesmo ocorre se digitarmos a palavra “cam”, como a Figura 4 ja mostra a unica palavra que irá mostrar em sua saída é “camaro”.

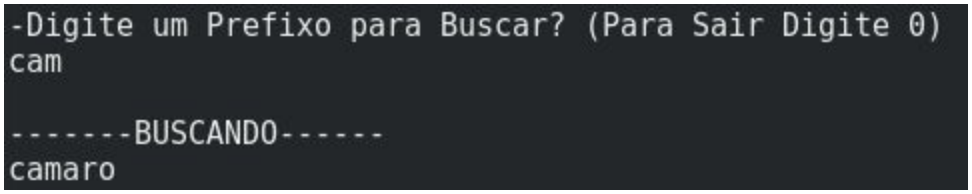

Figura 4 - Resultado pelo sufixo “cam”.

O algoritmo Aho-Corasick é um algoritmo sequencial de busca inventado por Alfred V.Aho e Margaret J.Corasick. É um tipo de algoritmo de correspondência de dicionário que localiza elementos de um conjunto finito de strings dentro de um texto de entrada .

Informalmente o algoritmo constrói uma máquina de estados finitos que se assemelha a uma árvore Trie com links adicionais entre os vários nós internos. Esses links internos extras permitem transições rápidas entre coincidências de string com falha (por exemplo, uma pesquisa por cat em um trie que não contém cat , mas contém cart falharia no nó prefixado por ca), para outros ramos do trie que compartilham um prefixo comum (por exemplo, no caso anterior, um ramo para atributo pode ser a melhor transição lateral). Isso permite que o autômato transite entre as sequências de caracteres sem a necessidade de retrocesso.

A complexidade do algoritmo é linear no comprimento das strings mais o comprimento do texto pesquisado mais o número de correspondências de saída. Observe que, como todas as correspondências são encontradas, pode haver um número quadrático de correspondências se todas as correspondências de substring (por exemplo, dicionário = a , aa , aaa , aaaae string de entrada é aaaa)

A Similaridade usando trigrams consiste em um grupo de três caracteres consecutivos retirados de uma string. Podemos medir a similaridade de duas strings contando o número de trigramas que elas compartilham. Esta ideia simples acaba por ser muito eficaz para medir a semelhança de palavras em muitos texto para ver se houve plágio ou não .
Obs: Uma cadeia é considerada como tendo dois espaços prefixados e um espaço com sufixo ao determinar o conjunto de trigramas contidos na cadeia. Por exemplo, o conjunto de trigramas na string (cat) - “c”,”ca”,”cat” e “at”.

A Radix Tree representa uma trie otimizada para espaço em que cada nó é único filho mesclado de seu pai, ao contrário das tries regulares as arestas podem ser rotuladas com sequência de elementos ou com elementos únicos. Isso torna a radix tree muito mais eficiente para conjuntos pequenos (especificamente se às sequência forem longas) e também para conjuntos de sequência que compartilham prefixos longos.

Ao contrário das árvores regulares (onde as chaves inteiras são comparadas desde o início até o ponto de desigualdade), a chave em cada nó é comparada com o pedaço de bits por pedaço de bits, onde a quantidade de bits nesse pedaço de bits esse nó é a raiz r da raiz. Quando o r é 2, o radix trie é binário (isto é, compara a porção de 1 bit desse nó da chave), o que minimiza a dispersão à custa de maximizar a profundidade.