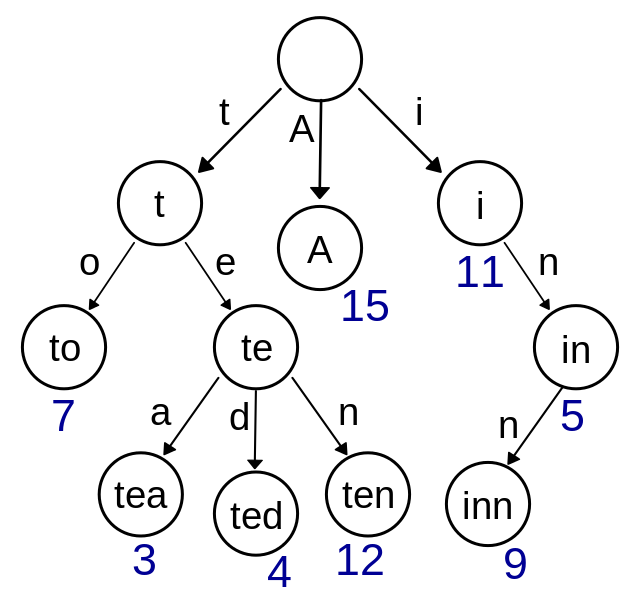
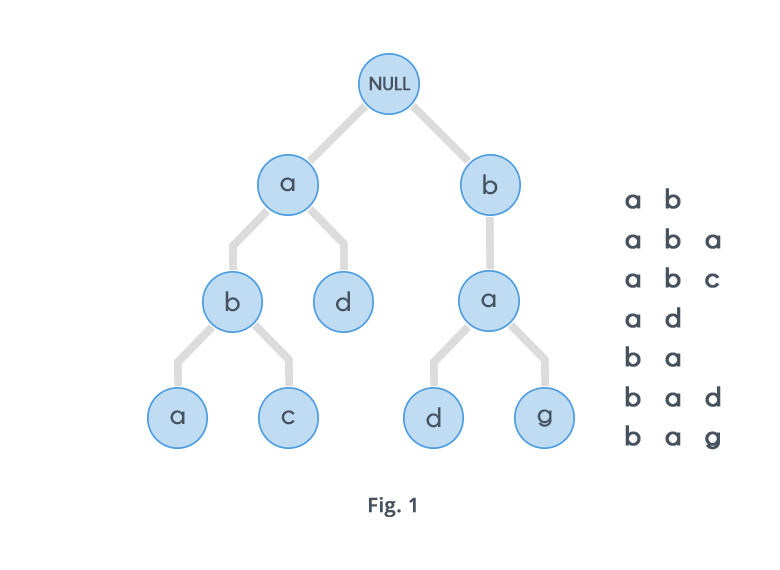
#### Etapa 2 - Trie de Sequências Binárias

Nesta etapa, vocês deverão implementar uma trie (de re**trie**val), uma árvore de prefixos, que pode ser usada para armazenar um array associativo em que as chaves são normalmente cadeias de caracteres. Tries organizam os dados em árvores, organizadas de acordo com a representação das chaves, por exemplo a a representação em binário ou decimal de um número ou os caracteres de uma string. Por exemplo, a trie para as chaves "A", "to", "tea", "ted", "ten", "i", "in" e "inn" poderia ser representada da seguinte forma (desconsidere os números):



Outro exemplo de trie:



A busca examina a chave caractere por caractere, percorrendo a árvore de acordo com cada caractere encontrado. Cada nó pode ter até um filho para cada possível caractere. Usa-se apenas a parte de cada chave necessária para diferenciá-la das outras chaves

No nosso caso, estamos interessados em um trie para representar sequências de digítos binários associados a um bloco (*bucket*) de arquivo.

Por exemplo, considere que:

– Chaves cujos endereços hashing começam com os bits 0 estão em um bucket A

– Chaves cujos endereços hashing começam com os bits 10 estão em um bucket B

– Chaves cujos endereços hashing começam com os bits 11 estão em um bucket C

A trie associada seria a seguinte:

0

1

1

0

A

B

C

Outro exemplo, considere que:

– Chaves cujos endereços hashing começam com bits 010 estão em um bucket A

– Chaves cujos endereços hashing começam com bits 011 estão em um bucket B

– Chaves cujos endereços hashing começam com bits 100 estão em um bucket C

– Chaves cujos endereços hashing começam com bits 101 estão em um bucket D

– Chaves cujos endereços hashing começam com bits 11 estão em um bucket E

A trie associada seria a seguinte:

0

1

1

0

A

B

C

1

1

0

D

0

1

E

Como representar a trie? Não queremos representar a trie em forma de árvore, pois teríamos que fazer comparações até achar a chave desejada. Estendemos a árvore até que todas as folhas estejam no mesmo nível, com vários caminhos levando ao mesmo bucket. Por exemplo, no caso da primeira trie com números binários, ela ficaria da seguinte forma:

0

1

1

0

A

B

C

1

0

Com isso, é possível fazer uma correspondência de um para um dentre os caminhos e números inteiros começando em 0, sendo possível utilizar um vetor para representar estes caminhos (este vetor é chamado de diretório):

|  |  |  |
| --- | --- | --- |
| 0 | 00 | A |
| 1 | 01 | A |
| 2 | 10 | B |
| 3 | 11 | C |

Nesse exemplo, a primeira coluna indica a posição do vetor, a segunda coluna sua representação em binário. A terceira coluna é o vetor propriamente dito (o que é armazenado de fato).

No caso da segunda árvore de dígitos, seria necessário acrescentar caminhos nulos, uma vez que os trechos iniciados em 00 encontram-se sem associação:

0

1

1

0

A

B

C

1

1

0

D

0

1

E

0

1

0

NIL

0

1

Essa trie, representada em um vetor, ficaria da seguinte forma:

|  |  |  |
| --- | --- | --- |
| 0 | 000 | -1 |
| 1 | 001 | -1 |
| 2 | 010 | A |
| 3 | 011 | B |
| 4 | 100 | C |
| 5 | 101 | D |
| 6 | 110 | E |
| 7 | 111 | E |

Os caminhos nulos, nesse caso, foram representados por -1.

Uma implementação adequada de uma trie permite a extensão de suas chaves. Por exemplo, imagine que na primeira trie com números binários, seja necessário agora separar elementos de B, levando parte para D. Ou seja, itens que começavam em 10 e iam para B, agora são separados em: itens começando em 100 vão para B e itens começando em 101 vão para D. Veja como ficará a trie após isso:

0

1

1

0

A

B

D

C

0

1

Nesse caso, o diretório será dado da seguinte forma:

|  |  |  |
| --- | --- | --- |
| 0 | 000 | A |
| 1 | 001 | A |
| 2 | 010 | A |
| 3 | 011 | A |
| 4 | 100 | B |
| 5 | 101 | D |
| 6 | 110 | C |
| 7 | 111 | C |

Assim, o objetivo desta etapa é montar uma trie a partir de um conjunto de pares chave / valor, onde as chaves são sequências binárias e valores são inteiros. A trie deverá se extensível, ou seja, começar com chaves a partir de um tamanho e possibilitar seu redimensionamento para um tamanho maior. Para isso, sua trie deverá possuir um métodos de inserção e redimensionamento, bem como uma aplicação de teste para apresentar seus métodos. Recomenda-se utilizar um campo para armazenar o tamanho (comprimento) da chave.

|  |
| --- |
| **Data de Entrega da Etapa 2:** 25 de Junho de 2018 |