Tries

Uma *trie* [1] é uma estrutura em árvore, na qual cada nodo corresponde a um caracter de uma cadeia de caracteres. Note-se que não se perde generalidade ao usar alfabetos inteiramente aleatórios.

> Por exemplo, pode-se ter um alfabeto composto pelos dez dígitos numéricos, 0..9. Ou um alfabeto binário, cujos caracteres são 0..1. Ou finalmente, o alfabeto convencional formado por A..Z.

O nome *trie* vem das letras centrais da palavra *retrieval* que significa recuperação.

Esta estrutura se mostra robusta quando um grande número de elementos a serem recuperados compartilham prefixos comuns. Este é o caso, por exemplo, de arquivos de palavras, ou dicionários. No português, como em qualquer idioma ocidental, dois fenômenos ocorrem:

- Muitas palavras compartilham prefixos. Por exemplo caso, casa, casamento, casinha, castiço, castiçal, casaca, cascudo, casca,cascalho, cascavel, cascata, caspa, castanha, castor, etc compartilham o prefixo inicial *cas*.
- Inúmeros prefixos possíveis, inexistem no dicionário. Por exemplo, não há palavras em português começando com sdr, nem com jhu, nem com lpr, nem com... Note-se que há muitos mais prefixos inexistentes do que o contrário

Na estrutura *trie* cada caminho da raiz até uma folha é uma palavra válida no universo de chaves. Para impedir confusão entre duas palavras de mesmo prefixo, por exemplo, MES e MESA, usa-se um símbolo terminador das cadeias, por exemplo, "∗". Com isso, as palavras completas seriam MES∗ MESA∗, evitando-se a confusão. Note-se que o grau da árvore é igual à cardinalidade do alfabeto. Assim, por exemplo, em um dicionário português, a cardinalidade poderia ser 23 (o alfabeto menos K, Y e W, ignorando-se os caracteres acentuados e o ç). Deve-se observar que a *trie* não é a única maneira de resolver este problema no mundo da programação. Outra candidata séria e que muitas vezes é usado em implementações reais é a tabela *hash*.

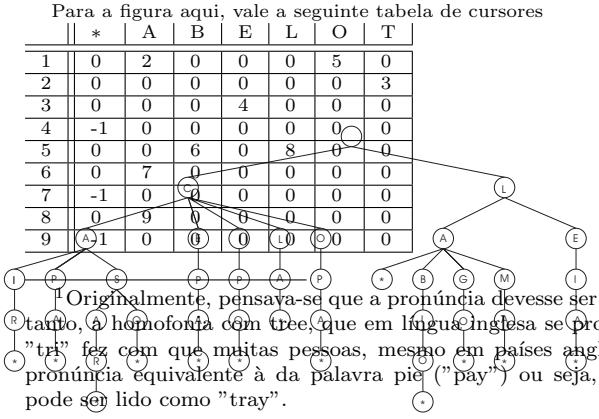
Seja, por exemplo a *trie* do conjunto de palavras cair, capa, casar, caso, cepa, cipó, clã, copa, lã, lábio, lago, lama, leia.

Implementação tipo 1

A princípio, pode-se pensar em uma implementação na qual os nodos da árvore tenham grau=cardinalidade como acima explicado. Fica evidente o grande desperdício de espaço que esta técnica vai apresentar, já que todos os nodos terão (por exemplo) 26 endereços e a grande maioria deles vai usar 1 ou 2 deles. Veja-se a seguir uma trie assim implementada, válida para as palavras "ATE∗", "OBA∗"e "OLA∗", já devidamente adaptada para conciliar o desenho com a implementação na forma de cursores.

Para a figura aqui, vale a seguinte tabela de cursores

	*	A	B	E	L	O	T
1	0	2	0	0	0	5	0
2	0	0	0	0	0	0	3
3	0	0	0	4	0	0	0
4	-1	0	0	0	0	0	0
5	0	0	6	0	8	0	0
6	0	7	0	0	0	0	0
7	-1	0	0	0	0	0	0
8	0	9	0	0	0	0	0
9	-1	0	0	0	0	0	0



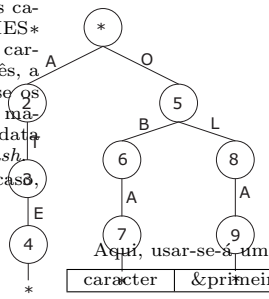
[1] Originalmente, pensava-se que a pronúncia devesse ser "tri". Entretanto, a homofonia com tree, que em língua inglesa se pronuncia como "tri" fez com que muitas pessoas, mesmo em países anglófonos use a pronúncia equivalente à da palavra pie ("pay") ou seja, o nome *trie* pode ser lido como "tray".

Implementação tipo 2

Uma proposta alternativa razoável (Aho, Data Structures & Algorithms, pág. 169) do seguinte tipo

estrutura nodo
letra CARAC
endereço FILHO
endereço IRMAO
fim{estrutura}

Esta estrutura considera cada nodo da *trie* como uma lista encadeada de caracteres. O primeiro endereço aponta para a próxima letra daquele nivel e o segundo aponta para o primeiro filho deste nodo. Veja-se um desenho desta estrutura para o mesmo universo de palavras: "ATE∗", "OBA∗"e "OLA∗"



Aqui, usar-se-á uma lista encadeada de nodos. Cada um terá o formato

caracter	&primeiro filho	&próximo irmão

Eis como ficaria a estrutura da *trie* acima:

	Car	Filho	Irmão
1		2	-1
2	A	3	6
3	T	4	-1
4	E	5	-1
5	*	-1	-1
6	O	7	-1
7	B	8	10
8	A	9	-1
9	*	-1	-1
10	L	11	-1
11	A	12	-1
12	*	-1	-1

☞ Para você fazer

Implemente uma *trie* que contenha as palavras

`CALO*,CANA*,PEDAL*,POPA*.`

Usando ambos os tipos (1 e 2). Pergunta-se: qual a sétima linha da matriz de cursores da implementação tipo 1?

E qual a décima linha da matriz de cursores da implementação tipo 2?