Trabalho 1 - Árvore de Busca Digital Binária
Detalhamento da E/S requisitada para o Trabalho 1.
Neste documento há exemplos de duas Bitwise Tries

E/S para a entrada do programa

Menu principal de opções

---

1 - Inserção,

2 - Remoção,
3 - Busca
4 - Visualização
5 - Fim

Escolha uma opção (1 a 5):

E/S para a Inserção
Observações para a inserção:
As strings de bits têm tamanhos variados.
Considere que a maior string terá até 16 dígitos, incluindo o '\0'.
***Não** implemente uma árvore de prefixo ou patrícia.*

>> Digite o binário para inserção: 10010011 >> Chave inserida com sucesso.

>> Digite o binário para inserção: 10010011 >> Chave repetida. Inserção não permitida.

>> Digite o binário para inserção: 12345678
>> Chave inválida. Insira somente números binários (ou -1 retorna ao menu).

>> Digite o binário para inserção: -1 >>
Retornando ao menu.

E/S para a Remoção
Observações para a remoção:

Não precisa liberar a memória (free) do nó removido. Apenas remova-o da árvore.

>> Digite o binário para remoção: 00
>> Chave encontrada na árvore: 00
>> Caminho percorrido: raiz, esq, esq >>
Chave removida com sucesso.

>> Digite o binário para remoção: 0
>> Chave não encontrada na árvore: -1.
>> Caminho percorrido: raiz, esq

>> Digite o binário para remoção: 120
>> Chave inválida. Insira somente números binários (ou -1 retorna ao menu).

>> Digite o binário para remoção: -1 >>
Retornando ao menu.

E/S para a Busca

>> Digite o binário para busca: 00
>> Chave encontrada na árvore: 00
>> Caminho percorrido: raiz, esq, esq

>> Digite o binário para busca: 1010
>> Chave encontrada na árvore: 1010
>> Caminho percorrido: raiz, dir, esq, dir, esq

>> Digite o binário para busca: 0
>> Chave não encontrada na árvore: -1
>> Caminho percorrido: raiz, esq

>> Digite o binário para busca: 120
>> Chave inválida. Insira somente números binários (ou -1 retorna ao menu).

>> Digite o binário para busca: -1
>> Retornando ao menu.

E/S para a Visualização

Instruções para a visualização:

1) Exiba a árvore diretamente ao entrar nesta opção.
2) Cada linha indica um **N**ível, começando do zero (raiz).
3) Os nós devem ser exibidos como ***(dígito, info, esq, dir)***, onde: ***dígito*** : indica se este nó representa ***raiz*** , dígito ***0*** ou dígito ***1*** ;

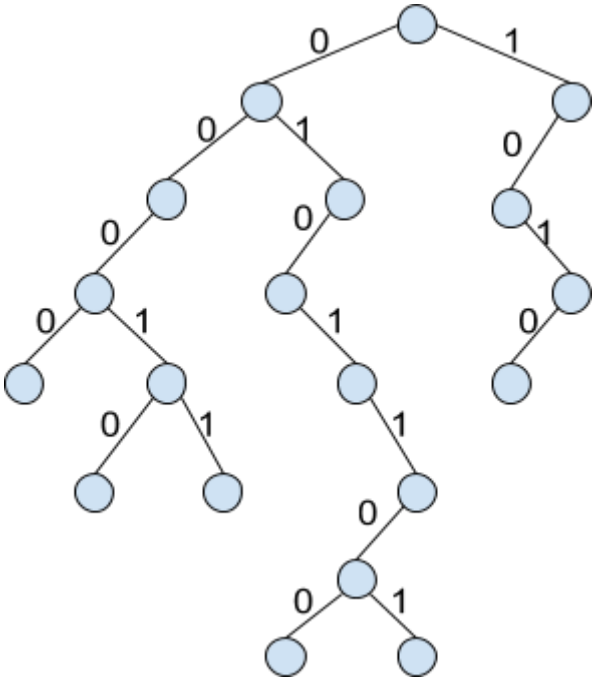
info: determina se o nó é terminal (***T***) ou Não Terminal (***NT***) a uma chave;

esq: ponteiro (***end de memória***) do filho da esquerda para o *dígito zero* ;

dir: ponteiro (***end de memória***) do filho da direita para o *dígito um*.

****

A título de exemplo, considere uma *bitwise trie* com estas chaves inseridas:

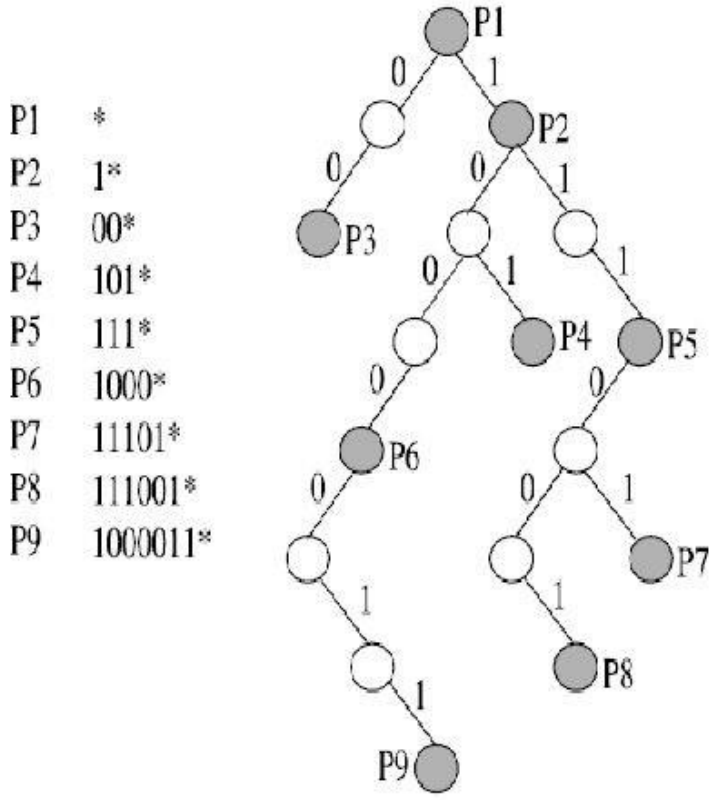
chaves: 00 0000 00010 00011 0101100 0101101 10 101 1010	

Referência para a Figura: Jaime Szwarcfiter; Lilian Markenzon "Estruturas de Dados e Seus Algoritmos", LTC, 2014.

Saída esperada para a visualização da árvore do primeiro exemplo (***&esq*** e ***&dir*** devem exibir o número do endereço de memória):

>> N0 (raiz, NT, &esq, &dir)
>> N1 (0, NT, &esq, &dir) (1, NT, &esq, null)
>> N2 (0, T, &esq, null) (1, NT, &esq, null) (0, T, null, &dir)
>> N3 (0, NT, &esq, &dir) (0, NT, null, &dir) (1, T, &esq, null)
>> N4 (0, T, null, null) (1, NT, &esq, &dir) (1, NT, null, &dir) (0, T, null, null)
>> N5 (0, T, null, null) (1, T, null, null), (1, NT, &esq, null)
>> N6 (0, NT, &esq, &dir)
>> N7 (0,T, null, null) (1, T, null, null)

Pode-se considerar como outro exemplo a *bitwise trie* representada pela Figura abaixo. O nó denominado P1 representa a raíz da árvore e os nós P2, P3, P4, P5, P6, P7, P8 e P9 os finais das chaves inseridas.



Referência para a Figura: Huang, Kun et al. “Memory-efficient IP lookup using trie merging for scalable virtual routers.” *J. Network and Computer Applications* 51 (2015): 47-58.

Saída esperada para a visualização da árvore do segundo exemplo (***&esq*** e ***&dir*** devem exibir o número do endereço de memória):

>> N0 (raiz, NT, &esq, &dir)
>> N1 (0, NT, &esq, null) (1, T, &esq, &dir)
>> N2 (0, T, null, null) (0, NT, &esq, &dir) (1, NT, null, &dir)
>> N3 (0, NT, &esq, null) (1, T, null, null) (1, T, &esq, null) >>
N4 (0, T, &esq, null) (0, NT, &esq, &dir)
>> N5 (0, NT, null, &dir) (0, NT, null, &dir), (1, T, null, null)
>> N6 (1, NT, null, &dir) (1, T, null, null)
>> N7 (1, T, null, null)
