Seja T uma árvore TRIE, onde cada nó representa um determinado caractere c pertencente a um alfabeto α, e suas subárvores representam o resto de uma cadeia de caracteres cc, onde a raiz representa o primeiro caractere de cc, e assim recursivamente.

Problema 1: Inserir um valor V em T, em uma determinada chave $K = (c_1, c_2, \ldots, c_n)$

V precisa ser inserido no nó de T que representa c_n, sendo esse um filho do nó que representa c_{n-1}, e assim sucessivamente até c_1. Além disso, no pior caso, todos esses nós precisam ser criados.

c_k só pode ser acessado a partir de c_{k-1}, portanto a função que modela a quantidade de passos tomados por qualquer forma de acesso ao nó que representa c_n (tal que essa forma crie os nós) seria:

$$T(n) = a + T(n-1) = a + a + T(n-2) = a(n-1) + T(1)$$

Já o acesso a c_1 é dado por: $T(1) = a$

onde a é a quantidade necessária para se criar um determinado nó

Logo, $T(n) = a\,n - a + a = a\,n$

resolver o problema de inserir V nem T necessariamente segue a ordem de

1. acessar o nó que representa c_n
2. inserir V no valor desse nó
   (esse passo tem uma quantidade b constante de passos)

Portanto a complexidade é modelada por

$$P_1(n) = T(n) + b = a\,n + b$$

onde n *é o tamanho da chave, e* q *é uma constante*

Portanto, a complexidade é $\Theta(n)$, pois $P_1(n)$ é um polinômio de grau 1.

Problema 2: Remover um determinado valor V de T, cuja sua chave é $K = (c_1, c_2, \ldots, c_n)$

Para remover um valor de T na chave K, precisa-se acessar o nó que representa c_n. Além disso, no pior caso, os nós que correspondem ao caminho daquela chave precisam ser todos removidos de T.

Seguindo a mesma forma de acesso do problema anterior, porém considerando a complexidade da remoção de um nó

$$T(n) = \alpha_t + T(n-1) = \alpha_t + \alpha_t + T(n-2) = \alpha_t(n-1) + T(1)$$

Já o acesso a c_1 é dado por: $T(1) = \alpha_t$

onde a é a quantidade necessária para se criar um determinado nó

Logo, $T(n) = \alpha_t n - \alpha_t + \alpha_t = \alpha_t n$

Onde a_t é uma constante e denota o tamanho do alfabeto.

Podemos dizer que remover V do nó que representa c_n , dado que temos esse nó, seja 1. Então a complexidade do problema 2 é dada pela complexidade de achar o nó, e então remover o valor, ou seja:

$$P_2(n) = T(n) + 1 = \alpha_t n + 1$$

Portanto, a complexidade é $\Theta(n)$, pois $P_2(n)$ é um polinômio de grau 1.

Problema 3: Encontrar o valor V contido no nó localizado na chave $K = (c_1, c_2, \ldots, c_n)$

V só pode ser acessado no nó que representa c_n. O pior caso seria quando o nó existe. Nesse caso, o acesso a c_n, como já vimos, só é possível através do acesso a c_{n-1}. Logo, a função que modela o acesso seria:

$$T(n) = 1 + T(n-1) = 1 + 1 + T(n-2) = (n-1) + T(1)$$

Se o custo para obter V, dado que temos o nó c_n pode ser denotado por e, então a função que representa a quantidade de passos tomados para resolver o problema 3 seria:

$$P_3(n) = (n-1) + 1 = n$$

Portanto, a complexidade intrínseca do problema 3 seria $\Theta(n)$, pois $P_3(n)$ é um polinômio de grau 1, e além disso, é o próprio n.