Módulo Diccionario(*string*, σ) sobre Trie

El módulo Diccionario sobre Trie provee un diccionario en el que se puede definir, borrar, y encontrar una clave en tiempo lineal en relación a la longitud de la clave. Al estar implementado sobre la estructura Trie, no se provee un iterador sino que se utiliza el iterador del Módulo Conjunto Lineal para recorrer el conjunto de claves. Con este propósito, las claves definidas en el diccionario están almacenadas como conjunto en la estructura. Para describir la complejidad de las operaciones, llamaremos $copy(s)$ al costo de copiar el elemento $s \in \sigma$ (i.e., *copy* es una función de σ en $\mathbb{N}$).

Interfaz

función COPIAR(**in** s : σ) $\rightarrow$ res : σ
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}} s$}
Complejidad: $\Theta(copy(s))$
Descripción: función de copia de σ's

se explica con: DICCIONARIO(κ, σ) donde κ es STRING

géneros: `diccTrie`(κ, σ).

Operaciones básicas de diccionario sobre Trie

VACÍO() $\rightarrow$ res : `diccTrie`($string, \sigma$)
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}}$ vacio}
Complejidad: $\Theta(1)$
Descripción: genera un diccionario vacío.

DEFINIR(**in/out** d: `diccTrie`($string, \sigma$), **in** k: $string$, **in** s: σ) $\rightarrow$ res : `bool`
Pre $\equiv$ {$d =_{\text{obs}} d_0$}
Post $\equiv$ {$d =_{\text{obs}}$ definir(k, s, d) $\wedge$ $res =_{\text{obs}}$ def?(k, d_0)}
Complejidad: $\Theta(|k| + copy(s))$
Descripción: define la clave k con el significado s en el diccionario. Si la clave ya estaba definida y se sobreescribió el significado, devuelve *true*, y $false$ si no estaba definida y se agregó una clave nueva.
Aliasing: s se define por copia.

DEFINIDO?(**in** d: `diccTrie`($string, \sigma$), **in** k: `string`) $\rightarrow$ res : `bool`
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}}$ def?(k, d)}
Complejidad: $\Theta(|k|)$
Descripción: devuelve `true` si y sólo k está definido en el diccionario.

SIGNIFICADO(**in** d: `diccTrie`($string, \sigma$), **in** k: $string$) $\rightarrow$ res : σ
Pre $\equiv$ {def?(k, d)}
Post $\equiv$ {$res =_{\text{obs}}$ obtener(k, d)}
Complejidad: $\Theta(|k|)$
Descripción: devuelve el significado de la clave k en d.
Aliasing: res es modificable si y sólo si d es modificable.

BORRAR(**in/out** d: `diccTrie`($string, \sigma$), **in** k: $string$)
Pre $\equiv$ {$d = d_0 \wedge$ def?(k, d)}
Post $\equiv$ {$d =_{\text{obs}}$ borrar(k, d_0)}
Complejidad: $\Theta(|k| + size(\sigma))$, donde size($\sigma$) corresponde al costo de utilizar el destructor del significado asociado a k.
Descripción: elimina la clave k y su significado de d.
Aliasing: libera el espacio asignado a k y a su significado.

CLAVES(**in** d: `diccTrie`($string, \sigma$)) $\rightarrow$ res : `conj`($string$)
Pre $\equiv$ {true}
Post $\equiv$ {$res =_{\text{obs}}$ claves(d, k)}
Complejidad: $\Theta(1)$, dado que la estructura de representación del diccionario incluye un conjunto de *strings* que va almacenando las claves a medida que son definidas.

Descripción: devuelve el conjunto de claves definidas en d.
Aliasing: res no es modificable y se devuelve por copia.