
Trie TST. Freqüència de les longituds de les claus.

X84615_ca

Donada la classe *dicc* que permet gestionar diccionaris on només hi guardem claus úniques usant tries implementats amb la tècnica d'arbres ternaris de cerca (TST), cal implementar el mètode

```
vector<nat> freq_longituds() const;
// Pre: True
// Post: Retorna un vector amb les freqüències de les longituds de les claus.
// La mida del vector és igual a la longitud de la clau més llarga més un.
```

on a cada posició *i* del vector resultat conté la freqüència o quantitat de claus de longitud *i* del diccionari.

Les claus són del tipus string i els símbols utilitzats per construir el trie són els chars de les claus. S'ha usat el char especial '#' per indicar la fi de la clau.

Cal enviar a jutge.org la següent especificació de la classe *dicc* i la implementació del mètode dins del mateix fitxer. La resta de mètodes públics i privats ja estan implementats.

```
#include <iostream>
#include <vector>
using namespace std;
typedef unsigned int nat;

class dicc {
public:
    // Constructora per defecte. Crea un diccionari buit.
    dicc ();

    // Destructora
    ~dicc ();

    // Insereix la clau k en el diccionari. Si ja hi era, no fa res.
    void insereix (const string &k);

    vector<nat> freq_longituds() const;
    // Pre: True
    // Post: Retorna un vector amb les freqüències de les longituds de les claus.
    // La mida del vector és igual a la longitud de la clau més llarga més un.

private:
    struct node {
        char _c; // Símbol posició i-èsima de la clau
        node* _esq; // Fill esquerra, apunta a símbols mateixa posició formant un BST
        node* _cen; // Fill central, apunta a símbols següent posició
        node* _dre; // Fill dret, apunta a símbols mateixa posició formant un BST
        node(const char &c, node* esq = NULL, node* cen = NULL, node* dre = NULL);
    };
    node* _arrel ;
```

```

static void esborra_nodes(node* t);
static node* insereix (node *t, nat i, const string &k);

// Aquí va l'especificació dels mètodes privats addicionals
};

// Aquí va la implementació del mètode públic freq_longituds i privats addicionals

```

Degut a que jutge.org només permet l'enviament d'un fitxer amb la solució del problema, en el mateix fitxer hi ha d'haver l'especificació de la classe i la implementació del mètode *freq_longituds* (el que normalment estarien separats en els fitxers *.hpp* i *.cpp*). Per testejar la classe disposes d'un programa principal que insereix claus en un diccionari i després calcula i mostra les freqüències de les longituds de les claus del diccionari.

Entrada

L'entrada conté una llista de strings separats per canvis de línia: són les claus que tindrà el diccionari.

Sortida

Mostra les freqüències de les longituds de les claus del diccionari separedes per un espai.

Observació

Només cal enviar la classe requerida i la implementació del mètode *freq_longituds*. Pots ampliar la classe amb mètodes privats. Segueix estrictament la definició de la classe de l'enunciat.

Exemple d'entrada 1

OCA

Exemple d'entrada 2

CASA

CAS

Exemple d'entrada 3

DAU

DIT

AU

AVI

CASA

COP

CAP

CAPA

Exemple de sortida 1

Exemple de sortida 2

0 0 0 1

Exemple de sortida 3

1 0 0 1 1

OU
OLA
UN
EXTRAMUR
FUM
FOC
ILLA
ALA
AL

Exemple de sortida 4

0 0 4 9 3 0 0 0 1

Exemple d'entrada 5

A

OU

DAU

DIT

AU

AI

IL·LA

ALA

AL

I

Exemple de sortida 5

1 2 4 3 1

Informació del problema

Autoria: Jordi Esteve

Generació: 2026-01-25T16:48:28.126Z

© *Jutge.org*, 2006–2026.

<https://jutge.org>