
Trie TST. Llista ordenada de totes les claus.

X37516_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

```
list <string> llista_ordenada_dec () const;
```

que retorna una llista amb totes les claus del diccionari ordenades de forma decreixent.

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. Indica el cost en funció de *s* (nombre de símbols que té l'alfabet) i *l* (nombre mig de símbols que sol tenir una clau).

```
#include <iostream>
```

```
#include <list>
```

```
using namespace std;
```

```
typedef unsigned int nat;
```

```
class dicc {
```

```
public:
```

```
    // Constructora per defecte. Crea un diccionari buit.  
    dicc ();
```

```
    // Destructora  
    ~dicc ();
```

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

```
    list <string> llista_ordenada_dec () const;  
    // Pre: True  
    // Post: Retorna una llista amb totes les claus ordenades decreixentment.
```

```
private:
```

```
    struct node {  
        char _c;    // Símbol posició i-èssima 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 llista_ordenada_dec 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 *llista_ordenada_dec* (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 la llista ordenada de totes 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 la llista amb totes les claus ordenades de més gran a més petita, cada clau en una línia diferent.

Observació

Només cal enviar la classe requerida, la implementació del mètode *llista_ordenada_dec* i el cost en funció de *s* (nombre de símbols que té l'alfabet) i *l* (nombre mig de símbols que sol tenir una clau). Pots ampliar la classe amb mètodes privats. Segueix estrictament la definició de la classe de l'enunciat.

Exemple d'entrada 1

Exemple de sortida 1

Exemple d'entrada 2

OCA

Exemple de sortida 2

OCA

Exemple d'entrada 3

CASA

CAS

Exemple de sortida 3

CASA
CAS

Exemple d'entrada 4

DAU
DIT
AU
AVI
CASA
COP
CAP
CAPA
OU
OLA

UN
EXTRAMUR
FUM
FOC
ILLA
ALA
AL

Exemple de sortida 4

UN
OU
OLA
ILLA
FUM
FOC
EXTRAMUR
DIT

DAU
COP
CASA
CAPA
CAP
AVI
AU
ALA
AL

Exemple d'entrada 5

A

OU
DAU
DIT
AU
AI
ILLA
ALA
AL
I

Exemple de sortida 5

OU
ILLA
I
DIT
DAU
AU
ALA
AL
AI
A

Informació del problema

Autoria: Jordi Esteve

Generació: 2026-01-25T15:30:08.378Z

© *Jutge.org*, 2006–2026.

<https://jutge.org>