

ASD 2 - Labo 5

Mini-projet

Correcteur Orthographique



Application: correction + suggestions orthographiques

dictionary.txt

afterward
among
analog
apologize
behavior
catalog
center
color
color's
colors
defense
favor
favorite
flavor
gray
judgment
labeled
labeling
Labor
learned
...

+

input_sh.txt

Project Gutenberg's The Adventures of Sherlock Holmes, by Arthur Conan Doyle

This eBook is for the use of anyone anywhere at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at www.gutenberg.net

Title: The Adventures of Sherlock Holmes

Author: Arthur Conan Doyle

Posting Date: April 18, 2011 [EBook #1661]

First Posted: November 29, 2002

Language: English

...

=

output.txt

*gutenberg's
2:guttenberg's
2:guttenberg's
*boscombe
*trepoff
*eglow
1:glow
2:reglow
3:aglow
3:eglon
*eglonitz
*egria
1:eria
2:egeria
3:agria
3:egrid
*kramm
1:kram
1:kram
3:dramm
3:krama
3:krams
...

Entrées / Sorties

- **Les données d'entrée**
 - Un dictionnaire anglais (fourni)
 - Le fichier textuel à corriger (certains sont fournis)
- **En sortie**
 - Un fichier mentionnant tous les mots mal orthographiés et les suggestions trouvées à l'aide du dictionnaire
 - Le temps de chargement du dictionnaire en mémoire
 - Le temps d'exécution nécessaire pour corriger un texte

Mise en œuvre

- **2 versions**
 - Avec un container STL de votre choix
A justifier
 - Avec un *Ternary Search Trie*
Classe à mettre en œuvre par vous même

Marche à suivre

- Lire, nettoyer et stocker le fichier `dictionary.txt` dans la structure de données
- Lire mot par mot le fichier texte en entrée
 - Convertir en minuscule toutes les lettres de chaque mot
 - Garder uniquement les caractères a-z et les apostrophes en milieu de mot

Marche à suivre (suite)

- **Vérifier l'existence de chaque mot dans le dictionnaire**
 - Présent → rien à faire
 - Absent → générer toutes les variantes possibles du mot en se basant sur quatre hypothèses
 - Vérifier l'existence de chaque variante dans le dictionnaire
 - Si la variante existe dans le dictionnaire → elle sera proposée comme correction possible
- **Afficher les mots mal orthographiés avec les suggestions dans un fichier, gardez l'ordre des mots du fichier initial**

Variantes orthographiques

1. L'utilisateur a tapé une lettre supplémentaire
a**c**queux → aqueux (il y a un c en trop)
2. L'utilisateur a oublié de taper une lettre
aqueux → aq**u**eux (il manque la lettre u)
3. L'utilisateur a mal tapé une lettre
a**w**ueux → aq**u**eux (il y a un w à la place du q)
4. L'utilisateur a échangé 2 lettres (**consécutives**)
au**q**ueux → aq**u**eux (u et q intervertis)

Format fichier de sortie

- Pour chaque texte pour lequel vous vérifierez l'orthographe, vous générerez un fichier texte comportant
 - les mots mal orthographiés (**préfixés d'une étoile**)
 - suivi immédiatement des propositions de correction vérifiées (**préfixées du numéro de l'hypothèse**)

```
*lates  
1:ates  
1:ats  
1:late  
2:alates  
2:elates  
2:plates  
...  
3:bates  
3:cates  
3:dates  
...  
4:altes
```



Ternary Search Tries?

- **Structure de données apparentée aux arbres de recherche**
- **Brève introduction ici, et diverses ressources externes proposées**
- **Classe *TernarySearchTrie* à mettre en œuvre et utiliser**


Tries. [from **re**trieval, but pronounced "try"]

- Store characters in nodes (not keys).
- Each node has R children, one for each possible character.
- For now, we do not draw null links.


for now we do not
draw null links




ll keys
that start with s



link to trie for all keys
that start with she



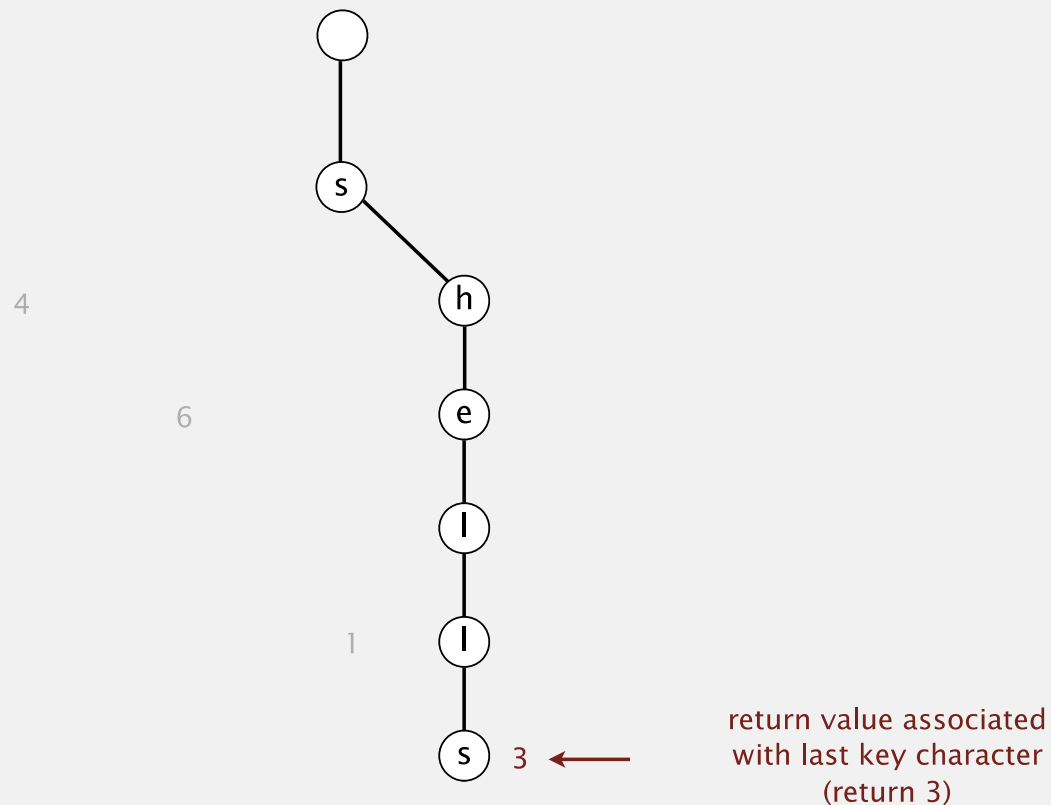
value for she in node
corresponding to last
key character



key	value
by	4
sea	6
sells	1
she	0
shells	3
shore	7
the	5

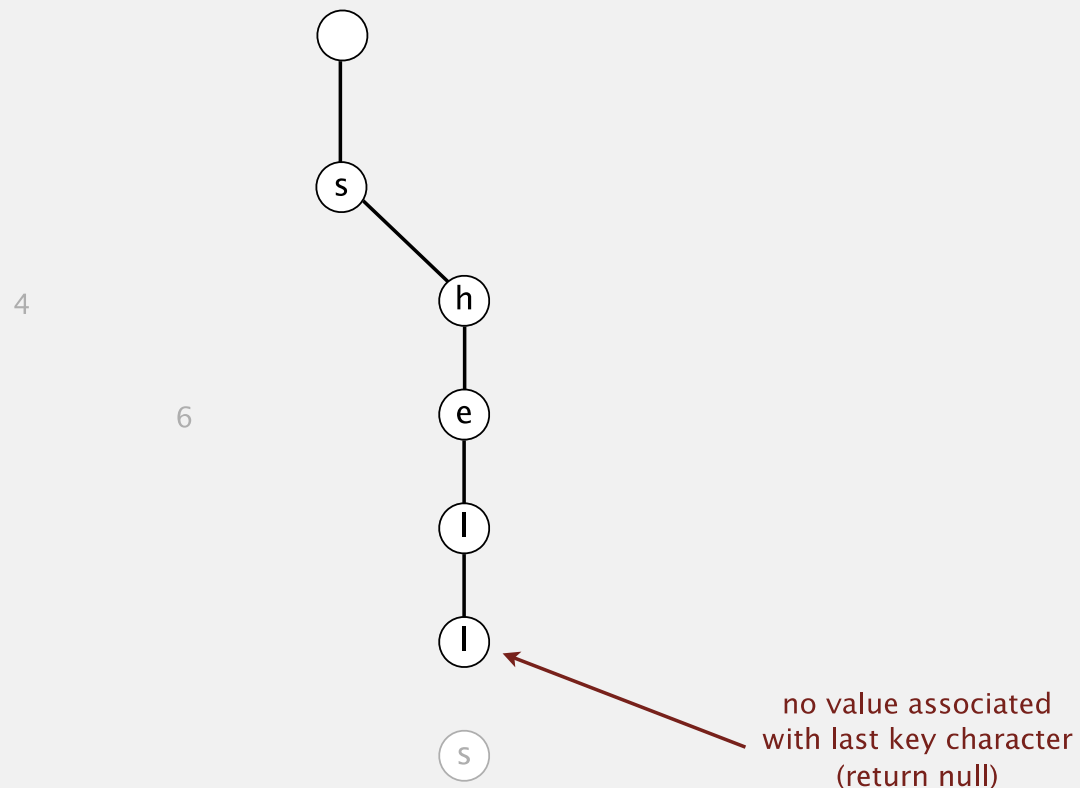
Follow links corresponding to each character in the key.

- **Search hit:** node where search ends has a non-null value.
- Search miss: reach null link or node where search ends has null value.



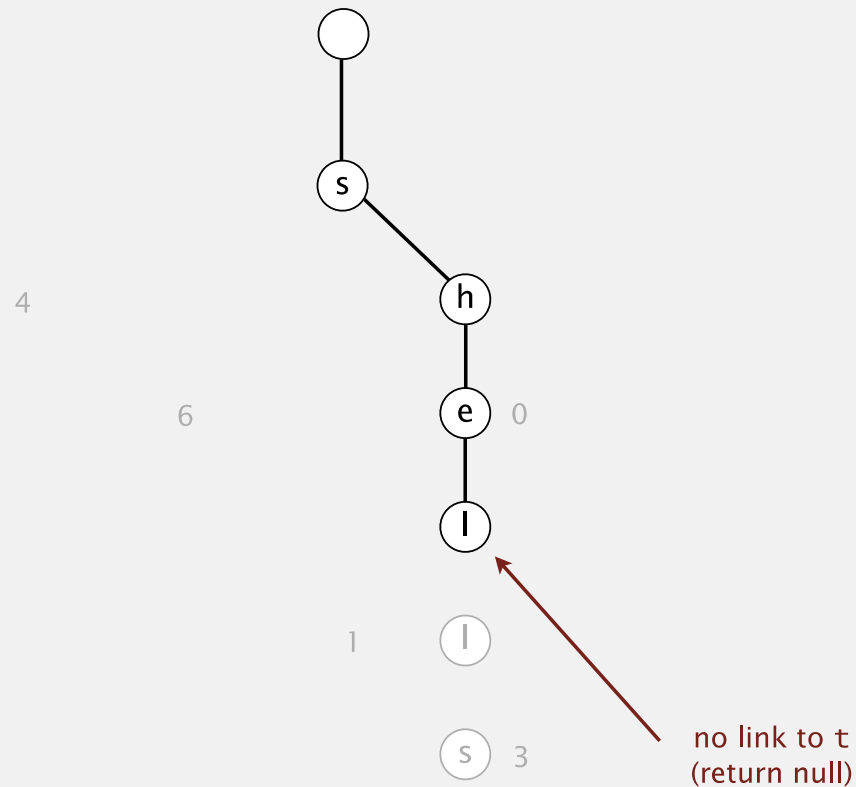
Follow links corresponding to each character in the key.

- Search hit: node where search ends has a non-null value.
- **Search miss:** reach null link or node where search ends has null value.



Follow links corresponding to each character in the key.

- Search hit: node where search ends has a non-null value.
- **Search miss:** reach null link or node where search ends has null value.



Follow links corresponding to each character in the key.

- Encounter a null link: create new node.
- Encounter the last character of the key: set value in that node.



4

6

0

5

1

7

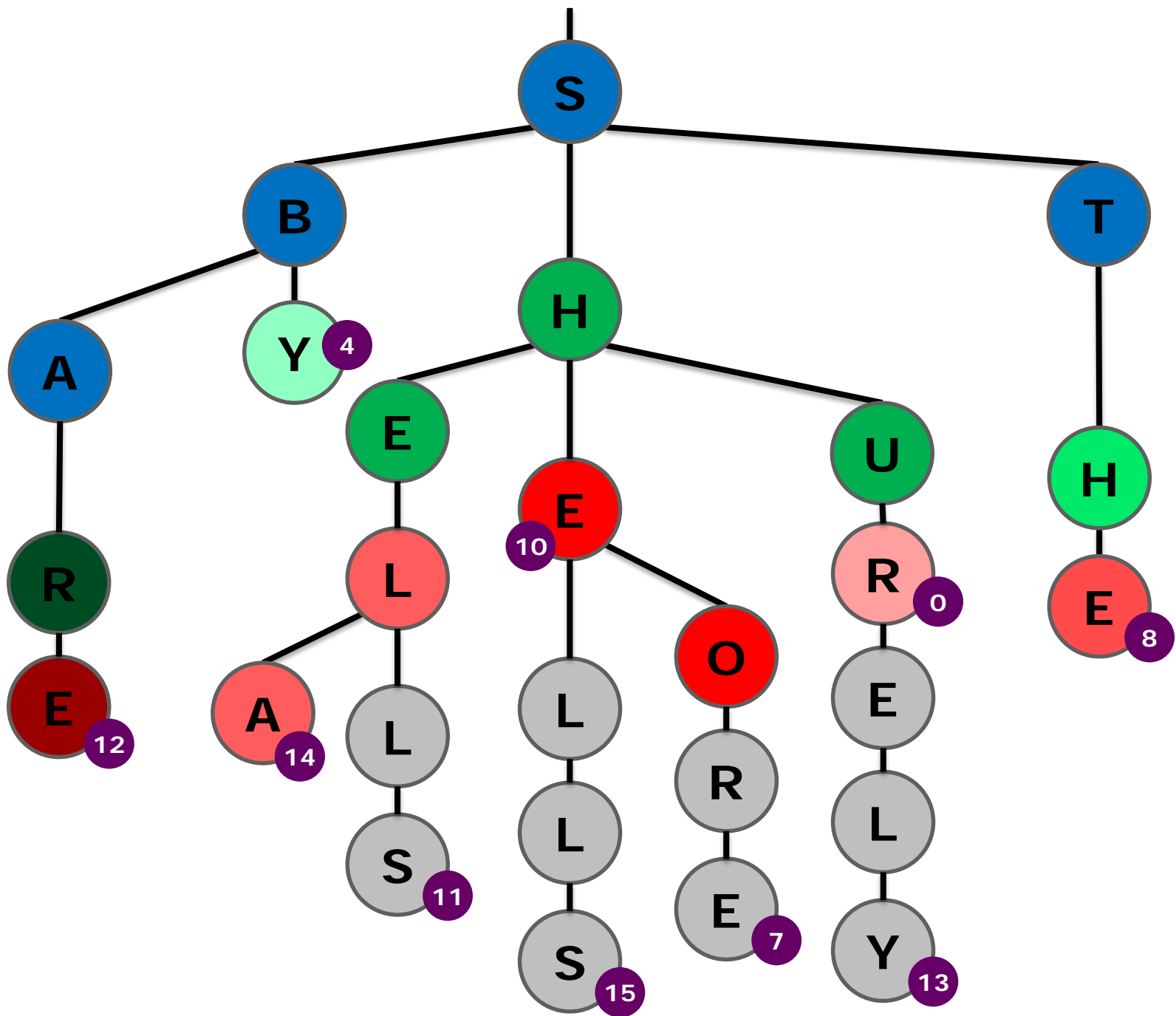
3

Limite des tries

- On doit réserver des tableaux de taille 26 ou même 256 pour chaque nœud
- Dans un tries avec beaucoup de mots, les nœuds éloignés du sommet seront très peu dense. Nous trouverons donc principalement des tableaux composés de près de 256 `nullptr`
→ Très gourmand en mémoire
- Nous allons donc voir dans ce laboratoire une structure à mi chemin entre les arbres de recherche et les tries → Ternary Search Tries

- Store characters and values in nodes (not keys).
- Each node has **3** children: smaller (left), equal (middle), larger (right).

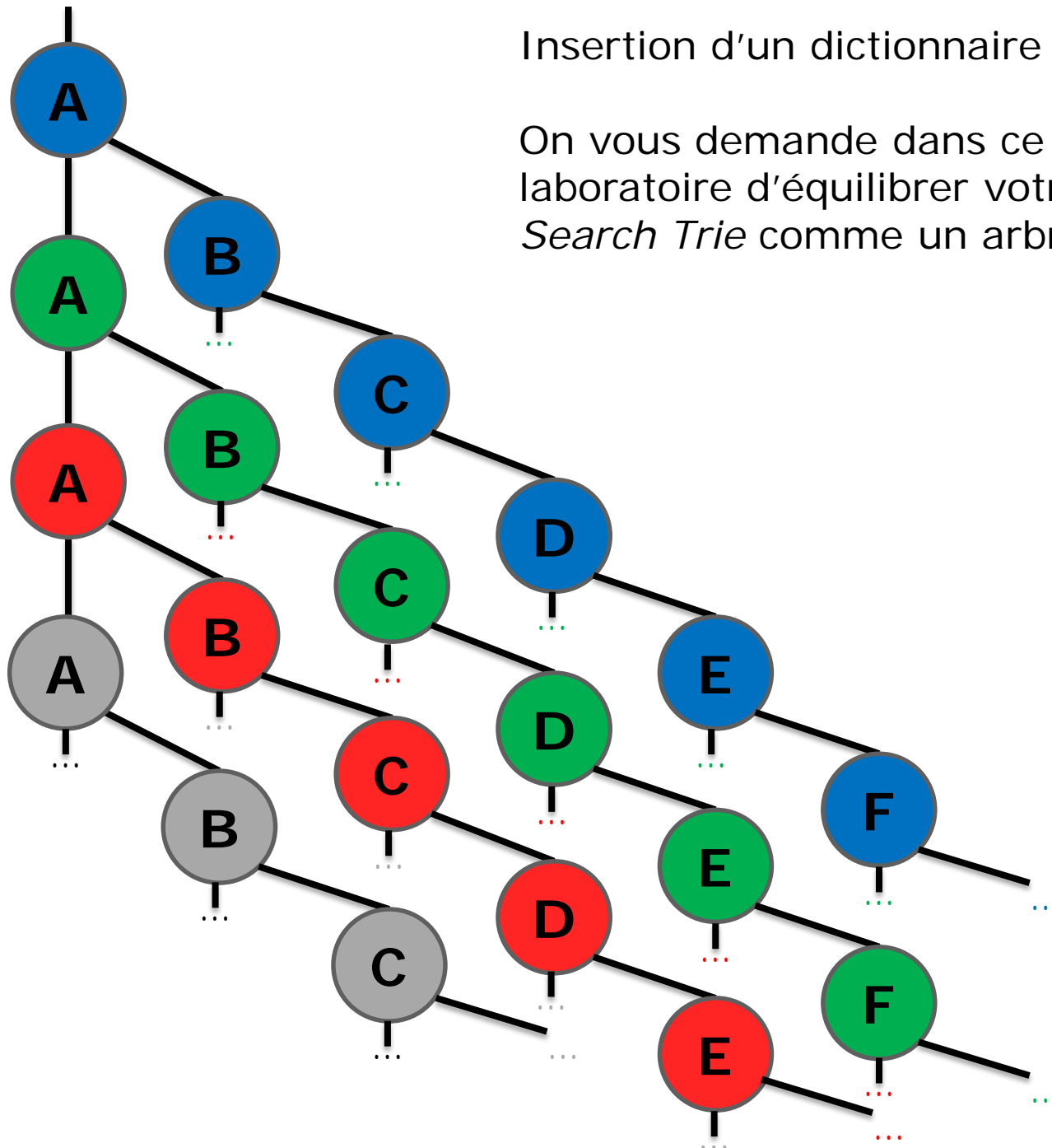






Insertion d'un dictionnaire trié...

On vous demande dans ce laboratoire d'équilibrer votre *Ternary Search Trie* comme un arbre AVL



Ressources

cppreference.com

Create account

Search

Page

Discussion

View

View source

History

C++ reference

C++98, C++03, C++11, C++14

ASCII chart
Compiler support

Language

Preprocessor
Keywords
Operator precedence
Escape sequences
Fundamental types

Headers

Concepts

Utilities library

Type support
Dynamic memory management
Error handling
Program utilities
Date and time
bitset
Function objects
pair – tuple (C++11)
integer_sequence (C++14)

Strings library

basic_string
Null-terminated byte strings
Null-terminated multibyte strings
Null-terminated wide strings

Containers library

array (C++11)
vector – deque
list – forward_list (C++11)
set – multiset
map – multimap
unordered_set (C++11)
unordered_multiset (C++11)
unordered_map (C++11)
unordered_multimap (C++11)
stack – queue – priority_queue

Algorithms library

Iterators library

Numerics library

Common mathematical functions
Complex numbers
Pseudo-random number generation

Input/output library

basic_streambuf
basic_filebuf
basic_stringbuf
ios_base
basic_ios
basic_istream
basic_ostream
basic_iostream
basic_ifstream
basic_ofstream
basic_fstream
basic_istreamstream
basic_ostringstream
basic_stringstream
I/O manipulators
C-style I/O

Localizations library

Regular expressions library (C++11)

Atomic operations library (C++11)

Thread support library (C++11)

Experimental standard libraries

Standard library extensions (library fundamentals TS)

optional
any
basic_string_view
Polymorphic allocator library

Filesystem library (filesystem TS)

External Links – Non-ANSI/ISO Libraries – Index

Ressources

BentleyAndSedgewick – Fast Algorithms for Sorting and Searching Strings.pdf

Fast Algorithms for Sorting and Searching Strings

Jon L. Bentley*

Robert Sedgewick#

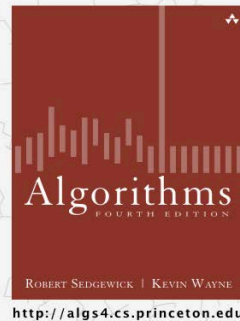
Abstract

We present theoretical algorithms for sorting and searching multikey data, and derive from them practical C implementations for applications in which keys are character strings. The sorting algorithm blends Quicksort and radix sort; it is competitive with the best known codes. The searching algorithm blends tries search trees; it is faster than hashing and other used search methods. The basic ideas behind algorithms date back at least to the 1960s, but their utility has been overlooked. We also present more complex string problems, such as pattern searching.

that is competitive with the most efficient string sorting programs known. The second program is a symbol table implementation that is faster than hashing, which is commonly regarded as the fastest symbol table implementation. The symbol table implementation is much more space-efficient than multikey trees, and executes more

Algorithms

ROBERT SEDGEWICK | KEVIN WAYNE



<http://algs4.cs.princeton.edu>



5.2 TRIES

- *R-way tries*
- *ternary search tries*
- *character-based operations*

Chapitre 5.2

52Tries.pdf

Ressources

AVLTree.h

```
//
// AVLTree.h
// SearchTrees
//
// Created by Olivier Cuisenaire on 25.11.14.
// Copyright (c) 2014 Olivier Cuisenaire. All rights reserved.
//

#ifndef SearchTrees_AVLTree_h
#define SearchTrees_AVLTree_h

#include <algorithm>

template < typename KeyType, typename ValueType >
class AVLTree {
private:
    //
    // Noeux de l'arbre. contient une cle, une valeur, et les liens vers les
    // sous-arbres droit et gauche.
    //
    struct Node {
    public:
        KeyType key;
        ValueType value;
        Node* right; // sous arbre avec des cles plus grandes
        Node* left; // sous arbre avec des cles plus petites
        int nodeSize;
        int nodeHeight;
        Node(KeyType key, ValueType value) : key(key),value(value), right(nullptr), left(nullptr) {}
    };
};
```

BinarySearchTree.h

```
//
// BinarySearchTree.h
// SearchTrees
//
// Created by Olivier Cuisenaire on 24.11.14.
// Copyright (c) 2014 Olivier Cuisenaire. All rights reserved.
//

#ifndef SearchTrees_BinarySearchTree_h
#define SearchTrees_BinarySearchTree_h

#include <algorithm>

template < typename KeyType, typename ValueType >
class BinarySearchTree {
private:
    //
    // Noeud de l'arbre. contient une cle, une valeur, et les liens vers les
    // sous-arbres droit et gauche.
    //
    struct Node {
    public:
        KeyType key;
        ValueType value;
        Node* right; // sous arbre avec des cles plus grandes
        Node* left; // sous arbre avec des cles plus petites
        int nodeSize;
        Node(KeyType key, ValueType value) : key(key),value(value), right(nullptr), left(nullptr), nodeSize(1) {}
    };

    //
    // Racine de l'arbre
    //
```

Ressources

- [**http://algs4.cs.princeton.edu/52trie/**](http://algs4.cs.princeton.edu/52trie/)
- [**http://en.wikipedia.org/wiki/Trie**](http://en.wikipedia.org/wiki/Trie)
- [**http://en.wikipedia.org/wiki/Ternary_search_tree**](http://en.wikipedia.org/wiki/Ternary_search_tree)