Algorithmique

TP 1 : Programmation dynamique

Calcul de la distance d'édition

Renaud Marlet
Laboratoire LIGM-IMAGINE

http://imagine.enpc.fr/~marletr

Distance de Levenshtein (1965) = distance d'édition

- Nombre minimum de modifications pour passer d'une chaîne à une autre :
  - suppression d'un caractère
  - insertion d'un caractère
  - remplacement d'un caractère par un autre
    - ex. d_L(ponts,hotes) = 3
- Mesure de la similarité de deux chaînes de caractères
  - applications : vérificateur orthographique, OCR...
  - généralisation du pb de plus longue séquence commune
- Complexité : $O(mn)$ pour deux mots de taille m et n

Distance de Damerau-Levenshtein

- Nombre minimum de modifications pour passer d'une chaîne à une autre :
  - suppression d'un caractère
  - insertion d'un caractère
  - remplacement d'un caractère par un autre
  - transposition de deux caractères successifs
    - ex. d_{DL}(écoles,éclose) = 2
      d_{L}(écoles,éclose) = 3

écoles, o ↔ l =
écloes, s ↔ e =
éclose

écoles, +l =
écloles, l → s =
écloses, -s =
éclose

- Couvrirait ≈80% des fautes d'orthographes (anglais)
- Complexité : $O(mn)$ pour deux mots de taille m et n

TP 1 : distance d'édition

1. **Fermez votre ordinateur !**
2. Étudiez le calcul par programmation dynamique de la distance de Levenshtein $d_L(s,s')$ entre deux chaînes s et s'
   - **suivez la méthodologie du cours !**
   - **indice :** étudiez la distance des préfixes
   - **rédigez** la preuve de la sous-structure optimale (qq lignes)

chaîne $s = c_1 \ldots c_n$

préfixe $s_i = c_1 \ldots c_i \; 1 \leq i \leq n$

3. Implémentez des solutions <u>récursives</u>, sans et avec mémoïsation [**≈ 10 LOC !**]
4. Quelles sont leur complexité en espace & en temps (approx : polyn. ou expon.) ?
5. Comparez et discutez leur temps d'exécution
6. Implémentez un algorithme <u>itératif</u> [**≈ 10 LOC !**]
7. Quelle est sa complexité en espace et en temps ?
8. Comparez et discutez son temps d'exécution
9. Affichez la suite de modifs élémentaires pour aller de s à s'
10. Étendez ce travail à la distance de Damerau-Levenshtein
11. Proposez et discutez une version itérative linéaire <u>en espace</u>

```
// Temps d'exéc.
#include <time.h>
clock_t t1 = clock();
traitement();
clock_t t2 = clock();
cout << (t2-t1) /
  CLOCKS_PER_SEC ;
```