Structures de données et algorithmes

pakpake

2 Avril 2020

Algo de recherche de motif

texte = T

sous-chaine égale à un motif = M

Algo de brute-force

alphabet (sigma) = contient les mots présent dans la chaine de caractère (dans le texte T)

le motif = M

et on compare le motif à chaque caractère (à chaque itération) du texte T

"Matching entre sous-chaine et autre chaine de caractère"

Complexité : en fonction du nb de caractère du motif et et taille du texte (combien de décalage)

càd : n-m :

- nb décalage
- m : taille du motif

donc la compléxité est :

$$0(m.(n-m))$$

Algorithme de force brute:

```
def bruteforce(T,M):
    """ Entrée : texte T de taille n et motif M de taille m
        Sortie : indice de début d'une sous-séquence de T qui n'est pas trouvée dans T """
        n = len(T)
        m = len(M)
        for i in range(0,n-m+1):
            j=0
            while j<m and T[i+j] == M[j]:
                j = j+1
            if (j==m):
                return i
        return -1
```

Algo de Boyer-Moore

Permet l'amélioration de la performance des algos de recherche de motif Cet algo permet de pré-traiter le motif.

On part de la fin du motif, on compare le motif avec le texte T et on effectue des sauts de taille du motif si le caractère est différents, sinon on compare le caractère le plus proche de la fin et on recommence.

Si on compare avec l'algo de bruteforce, on a :

15 positions à tester et avec Boyer-Moore on a 5 positions à tester, pour un **motif de taille 6** et un **texte de taille 20**.

La complexité asymptotique est donc :

$$0(n.m)$$

avec **n la taille du texte** et **m la taille du motif**

Cet algorithme sera efficace plus le motif sera grand, par exmple pour les textes litéraires, car les sauts seront grands et donc il y aura une meileure efficacité.

pretaraiter le texte: algo de préfixe et de suffixe

pattern = motif

Arbre lexicographique :

On peut avoir autant de fils que l'on veut.

Arbre de préfixe

L'arbre est ordonné, tel que :

- les valeurs des noeuds sont des caractères du texte
- les fils de cahque noeuds sont classés par ordre alphabétique
- Tout chemin partant de la racine et arrivant à une feuille correspond à un mot du texte
- Apres pré-traitement du texte (construction de l'arbre) le temps d'éxecution de la recherche d'un motif est proportionnel à la taille du motif

La racine est représentée par un '/' car les mots commencent par une lettre différente

On peut aussi avoir un arbre de préfixe compressé :

on met la 1ere lettre en commun, et on regroupe par mot ensuite, exemple :

bear, but bull => b : -> ears ; u -> lls ; u-> t

TP 11 : (avant dernier tp)

Exercice 2

Arbre de préfixe il faut compléter un arbre de préfixe

vrai si la lettre est la dernire letre d'un mot

De la racine '/', on a 2 sous arbres qui commence par 'a' ou par 'b', modélisation à compléter

Les premiers exercices avaient été demandé pour le controle sur table de l'année dernière.

pour la partie vérificatueur orthographique c'est à ce moment là qu'on fera l'analyse préfixe