Soutenance de projet

Algorithmique du texte

Etienne BINGINOT & Axelle MAZUY

Université de Rouen - UFR Science et Techniques - M1 GIL

27 décembre 2023

Contexte du projet

1. Construction de générateurs pseudo-aléatoires de texte et de mots
2. Implantation de l'algorithme Aho-Corasick pour compter les occurrences exactes d'un ensemble de k mots dans un texte. Les opérations ENFILER et DEFILER doivent s'exécuter en temps constant.
3. 2 méthodes doivent être utilisées pour représenter l'arbre : une matrice de transition et une table de hachage.
4. Génération aléatoire des textes de longueur 5 millions de la taille de l'alphabet demandé.
5. Génération aléatoire de 3 ensembles de mots de longueur 100 de la taille de l'alphabet demandé.
6. Utiliser les exécutables ac-matrice et ac-hachage sur les ensembles et les textes générés, relever les temps et les comparer par des courbes.

L'algorithme Aho-Corasick

- Permet de localiser toutes les occurrences d'un ensemble fini de k mots X dans un texte y de longueur n
- On commence par construire le trie prenant en entrée X
- On associe une fonction de sortie à chaque état terminal et on crée une boucle sur l'état initial pour tous les caractères dans l'alphabet n'étant pas préfixe d'un mot de X
- On associe à chaque état un état suppléant correspondant au plus long suffixe du mot de l'état, tel que le suppléant appartient à Pref(X) (parcours en largeur du trie en utilisant une file).
- On complète la fonction de sortie, si l'état suppléant est final alors l'état associé est final
- On recherche les occurrences des mots de X dans y en parcourant le trie avec y et signaler une occurrence quand on arrive dans un état final.

Structures de données utilisées

- Structure de trie : Trie, implémentée de deux façons par table de hachage et matrice de transition.
- Structure de file : permet de parcourir les noeuds en enfilant et défilant les numéros des noeuds afin de connaître les noeuds à traiter.
- Structure de pile : permet de stocker toutes les transitions depuis un noeud afin de calculer le noeud de suppléance de chaque noeud.
- Structure de transitions : permet de représenter une transition (son état initial, d'arrivée, son caractère).
- Tableau des états de suppléances : permet de récupérer depuis un état, son état de suppléance.

Modifications apportées à l'algorithme pour le sujet

- Modification de la fonction sortie : plutôt que de calculer des ensembles à renvoyer, nous avons choisi de sauvegarder le nombre de mots reconnus par un état dans un tableau
- Utilisation d'une pile : plutôt que d'utiliser une liste des états à traiter, une pile permettait une meilleure performance et une plus grande facilité d'implémentation, sans changer l'algorithme de base.

Expérimentations

- Calcul du nombre de mots reconnus par un état final :
  Première solution : le parcours des noeuds de suppléance d'un état,
  Problème : oubli de certains mots,
  Solution retenue : tableau d'occurrences indiquant pour chaque état le nombre de mots reconnus
- Factorisation via makefile :
  Première solution : pas de factorisation, mais deux fichiers C séparés
  Problème : beaucoup de duplication de code
  Solution retenue : utilisation du makefile pour créer deux exécutables en compilant notre fichier C avec des fichiers de trie différents en entrée.

Expérimentations

- Ajout de fonctions aux tries :
  Première solution : seules les fonctions de base du traitement des tries sont accessibles,
  Problème : impossibilité d'accéder à certaines informations nécessaires à l'algorithme,
  Solution retenue : factorisation du code des tries via des fonctions, rendues publiques
- Problème de génération d'un caractère aléatoire :
  Première solution : création de la graine en fonction du temps à chaque génération de mot
  Problème : les mots se ressemblaient voire étaient les mêmes en fonction des essais
  Solution retenue : multiplication de la graine par un nombre représentant la position du mot.

Conclusion

Conclusion sur les algorithmes :

- De manière générale, ac-matrice donne des temps plus rapides que ac-hachage
- En revanche ac-matrice nécessite un plus grand espace mémoire
- De plus, dans les deux cas, nous obtenons un résultat rapide ce qui signifie que Aho-Corasick est un algorithme performant.

Ce que l'on ressort de ce projet :

- l'apprentissage des différentes structures de données et leur imbrication,
- la gestion de la complexité en écrivant les algorithmes,
- apprentissage de la visibilité des fonctions

Cours d'algorithmique du texte [1]
Code du projet d'Algorithmique du texte [2]
Compte-rendu du projet d'Algorithmique du texte [3]

[1] LECROQ Thierry *Aho-Corasick* Recherche exacte d'un ensemble fini de mots

[2] BINGINOT Etienne MAZUY Axelle

[3] BINGINOT Etienne MAZUY Axelle *Compte-rendu*

Merci de votre attention !