

Programmation séquentielle

Série 13 - Table de hachage

28.02.2023

Buts

Dans ce travail vous verrez les concepts suivants:

- Créer et utiliser une librairie de table de hachage.
- Création de fonction de hachage simple.
- Evaluation des performances d'une implémentations.

Vous avez vu la théorie sur les tables de hachages au cours d'algorithmique.

Énoncé

Dans ce travail pratique il s'agit d'implémenter une table de hachage en utilisant la méthode du double hachage pour traiter les collisions.

Vous implémenterez une librairie de table de hachage permettant de stocker des paires clé-valeur de type chaîne de caractères (les clés et les valeurs sont des chaînes de caractères).

Concernant les structures de données et les fonctions de l'API de votre librairie, référez-vous au cours théorique.

Vous écrirez un programme qui permet de créer une table de hachage, et permet à l'utilisateur d'ajouter / supprimer / consulter des éléments de la table de hachage ainsi que d'afficher la table de hachage.

Gestion des collisions

Un des objectifs de ce travail est d'observer l'impact des fonctions de hachage utilisées sur le comportement de la table. Dans le cours, on a vu que l'utilisation d'une mauvaise fonction de hachage pouvait conduire à l'apparition de regroupements (ou "clusters" dans la table). Une bonne fonction de hachage produira beaucoup de petits regroupements, alors qu'une mauvaise fonction produira peu de grands regroupements.

Vous devrez faire des essais avec au moins deux fonctions de hachages différentes et deux fonctions de double hachage différentes. Voici un exemple de fonction de hachage avec lequel vous pouvez commencer à travailler :

```

val = 0;
for (int i = 0; i < strlen(c); ++i) {
    val = (43 * val + c[i]) % length;
}
return (val % length);

```

Et comme fonction de double hachage, vous pouvez commencer par utiliser un décalage constant.

Pour pouvoir faire des expérimentations, ajoutez les deux fonctions suivantes à votre librairie :

```

// retourne le nombre de regroupements trouvés dans la hm
int hm_count_clusters(hm h);

// retourne la taille moyenne des regroupements
double hm_mean_cluster_size(hm h);

```

Puis écrivez un programme qui remplit votre table de hachage avec un certain nombre de paire clé / valeur. Observez comment le changement de la / des fonctions de hachage influe sur le nombre de regroupements et leurs tailles.