Hashtables

Bourgeois Noé

16 mai 2021

Contents

1 Introduction

Il existe différentes méthodes de résolution des collisions des tables de hachage. Pour étudier les caractéristiques de 3 d'entre elles, nous avons implémenté une classe pour chacune, représentant un dictionnaire dont les clefs (uniques comme dans un dict en Python, ce qui implique que lors d'une insertion à une clef k donnée dans le conteneur, si cette clef existe déjà, il faut remplacer la valeur stockée) et valeurs sont des chaînes de caractères (donc des objets de type str):

classe	DictOpenAddressing	DictChainingLinkedList	DictChainingSkipList
gestion des collisions	double hachage	liste chaînée	skiplist
algorithme(s) de hachage	djb2		
	KR		
methodes	__init__(self, m): initialise l'objet		
	__len__(self): retourne le nombre d'éléments stockés (et non la taille de la table)		
	insert(self, key, value): insère l'élément value associé à la clef key lance une OverflowError si l'insertion est impossible)		
	search(self, key): renvoie la valeur associée à la clef key si cette dernière existe dans le conteneur lance une KeyError sinon		
	__setitem__(self, key, value): appelle insert		
	__getitem__(self, key): appelle search		
	delete(self, key): retire l'élément de clef key du conteneur s'il existe et lance une KeyError sinon		
	__delitem__(self, key): appelle delete		
propriété	load_factor: renvoie le facteur de charge: nombre d'éléments stockés dans le conteneur/taille de la table		

2 Méthodes

2.1 KR

Algorithm 1 Algorithme de Kernighan & Ritchie

```
1: procedure KR(string s)
2:     hash ← 0
3:     for all char c in s do
4:         hash ← hash + c
5:     end for
6:     return hash
7: end procedure
```

2.2 djb2

2.2.1 Sous-sous-section

Algorithm 2 Algorithme djb2 de Daniel J. Bernstein

```
1: procedure DJB2(string s)
2:     hash ← 0x1505
3:     for all char c in s do
4:         hash ← 33x hash + c
5:     end for
6:     return hash
7: end procedure
```

3 Résultats

statistiques sur le temps nécessaire pour l'exécution des méthodes insert, search et delete de la classe DictOpenAddressing pour différentes valeurs de ainsi que des représentations graphiques

comparaison du nombre de sondages effectués par les méthodes insert, search et delete des classes DictChainingLinkedList et DictChainingSkipList ;

discussion sur les avantages et inconv´enients de ces deux classes

1ère ligne	test
2ème ligne	test 2

comparaison de ces valeurs ainsi que du nombre moyen de sondages avant de trouver une cellule libre avec les r´esultats th´eoriques vus au cours

4 Discussion

l'insertion par chaînage ne peut pas lancer d'exception car l'insertion y est toujours possible discussion sur l'hypoth'ese de hachage uniforme pour les diff´erentes fonctions de hachage impl´ement´ees

5 Conclusion

L'adressage ouvert est plus performant que le chaînage mais est limité par la taille de son dictionnaire.