Berney Alec, Forestier Quentin, Herzig Melvyn

ASD2 – 05.12.2020

Tables de hachage et correcteur orthographique

Laboratoire 4



# Questions partie 1

## 1.1

Bien que les unordered\_set de la STL proposent d’insérer, chercher et supprimer des clés en O(1), ils sont non ordonnés.

Admettons que nous voulions stocker un grand nombre de données et régulièrement nous cherchons à trier le contenu de la structure (qui change constamment).

Posons N = #données distinctes stockés dans le set et l’unordered\_set à un moment M.

Posons T = #de fois que nous désirons trier le contenu.

### Génération de la structure

|  |  |
| --- | --- |
| Set O(N\*log(N)) | Unordered\_set O(N) |

### Récupération de la structure

|  |  |
| --- | --- |
| Set O(N) | Unordered\_set O(N) |

A cette étape, le set a l’avantage d’être déjà trié.

Quant à l’unordered\_set, son contenu doit encore être trié en O(N\*log(N))

### Complexité totale

|  |  |
| --- | --- |
| Set O(T\*N+N\*Log(N)) | Unordered\_set O(N+T\*N\*Log(N)) |

Ainsi dans ce genre de situation, la structure set est préférable à l’unordered\_set.

## 1.2

Les collisions sont gérées par chainage. Les valeurs qui génèrent la même clé sont stockées dans le même « bucket » (équivalent des listes dans la théorie du cours).

Le max\_load\_factor est le ratio entre le nombre d’élément réellement stocké et le nombre de buckets. Quand le ratio, par défaut à 1, est dépassé, la nombre de buckets et augmenté.

## 1.3

Cela permet de ne pas doubler inutilement la taille de la table de hachage. La table garde ainsi une taille relative au nombre d’élément que nous y insérons/supprimons.