MOUSSAOUI Abdelrahmane && DOGGA Nidhal
DISCHLER Jean-Michel
May 8, 2018

Rapport du projet SDA2

(Co)occurrences de mots

1. Ensemble ordonné:

• L'ensemble ordonné n'est qu'un tableau dynamique de valeurs, encapsulé dans une structure qui tient le nombre d'éléments insérés et le nombre de cases disponibles avant de ré-allouer de la mémoire.

```
typedef unsigned long index_t;
struct ordered_set_t {
        index_t *values;
        size_t   count;
        size_t   _available;
};
```

La raison pour laquelle cette implémentation est optimale, est le temps constant d'accès aux cases pendant l'insertion et la recherche dichotomique des éléments dans l'ensemble.

• La fonction intersection entre plusieurs ensembles ordonnés fonctionne de manière très simple:

- Chercher le plus petit ensemble (moins d'éléments à tester avec les autres ensembles)
- Tester l'appartenance de chaque élément de cet ensemble dans les autres ensembles.
- Insérer si cet élément appartient à tous les ensembles.

• La complexité de quelques opérations sur les ensembles ordonnées:

- Insertion et test d'appartenance dans un ensemble ordonné:
  - ⋆ Complexité: O(log n)
  - ⋆ Pire cas: $\log_2(n)$ opérations
- Intersection de deux ensembles:
  - ⋆ Complexité: O(m log m)
  - ⋆ Pire cas: $\frac{m \,.\, \log_2(m)}{2}$ opérations

2. Arbre binaire de recherche:

• L'implémentation de l'arbre binaire de recherche est grâce à une structure récursive qui tient le mot du nœud et l'ensemble ordonné de ces positions dans le texte, plus les nœuds fils gauche et fils droit.

```
struct bst_t {
        char *word;
        ordered_set_t *positions;
        struct bst_t *left_child;
        struct bst_t *right_child;
};
```

Cette structure était largement suffisante vu que les opérations effectuées sur les arbres sont majoritairement récursives.

• La fonction isBalanced a été implémentée récursivement de telle sorte qu'elle renvoie la différence des balances obtenus depuis l'appel récursif de la fonction de ses sous-arbres (gauche et droit).

• La fonction getAverageDepth a été implémentée à l'aide d'une fonction auxiliaire _getTotalDepth qui calcule la profondeur totale de l'arbre, elle renvoie donc la profondeur totale divisée par le nombre de nœuds de l'arbre.

• Étude de complexité de FindCoocurrences:

- Un arbre non équilibré peut être sous forme de liste chainée, ce qui rend l'étude de la hauteur d'un nœud de complexité O(n), mais pour un arbre équilibré, c plutôt O(log n).

3. Testes:

• Pour compiler et lancer le jeu de données global, veuillez taper la commande:

```
make tests [DEBUG=1] [SILENT=1]
```

• Pour compiler et lancer les testes partiels de chaque fonctionnalité:

```
make tests [all|bst|bstops|ordset] [DEBUG=1] [SILENT=1]
```

* Le flag DEBUG permet d'afficher le jeu de données et les messages du teste (recommandé).

* Le flag SILENT permet d'ignorer les message de compilation pour plus de clarté.

* Pour compiler les testes patiels il faut au spécifier au moins une unité (fonctionnalité) de teste parmis [all|bst|bstops|ordset].