Structure de données et algorithmes

Projet 1: Algorithmes de sélection

25 février 2022

L'objectif du projet est d'implémenter, d'analyser théoriquement et de comparer empiriquement différents algorithmes de sélection, c'est-à-dire permettant de trouver la `k`-ème plus petite valeur d'un ensemble de valeurs selon un certain ordre.

Algorithme de sélection

Le problème de sélection consiste à identifier dans un tableau de `N` valeurs numériques (dans ce projet, des entiers) la `k`-ième valeur la plus petite. On supposera dans ce projet que le tableau initial est non trié et qu'il est permis de permuter ses éléments. La fonction à implémenter `Select(A,k)` prendra comme argument le tableau `A` et la valeur de `k` et renverra la `k`-ième valeur la plus grande dans `A`.

On se propose d'étudier 4 solutions algorithmiques à ce problème:

1. Quicksort
2. Heapsort
3. Quickselect
4. Median-of-medians

Les deux premières solutions consistent simplement à appliquer respectivement les algorithmes `Quicksort` et `Heapsort` pour trier le tableau et ensuite à retourner la valeur se trouvant à la `k`-ième position du tableau trié. Les deux autres algorithmes sont décrits ci-dessous.

Quickselect. L'algorithme `Quickselect` est une adaption du `Quicksort` pour le problème de sélection. L'idée de cet algorithme est comme pour le quicksort de répartir les éléments du tableau en fonction d'une valeur pivot (choisie arbitrairement dans le tableau). Une fois la partition effectuée, plutôt que de faire les deux appels récursifs du `Quicksort`, il est possible de se passer de l'un ou des deux appels dans le cas de la sélection. En effet, si la position du pivot `q` est égale à `k`, l'algorithme peut renvoyer directement `A[q]` comme l'élément recherché. Si $\texttt{q} < \texttt{k}$ (resp. $\texttt{q} > \texttt{k}$), la `k`-ième valeur se trouve à droite (resp. à gauche) du pivot et un seul appel récursif peut être effectué sur le sous-tableau correspondant pour trouver la `k`-ième valeur.

Median of medians. Cet algorithme[1] est une variante du `Quickselect` qui effectue le choix du pivot de manière plus intelligente. Ce pivot est choisi de la manière suivante:

1. Si le tableau a une taille plus petite que 5, on utilise comme pivot la médiane du tableau.
2. Sinon, on divise le tableau en blocs de 5 élements et on détermine la médiane de chaque bloc.
3. On calcule la médiane des médianes de chacun des blocs et on utilise cette médiane comme valeur de pivot.

Le reste de l'algorithme est similaire à l'algorithme `Quickselect`. Pour l'étape 1 et pour l'étape 2, le calcul de la médiane d'un tableau d'au plus 5 éléments peut s'implémenter par exemple par un tri par insertion. Pour l'étape 3, le calcul de la médiane des médianes s'effectuera par un appel récursif à la fonction de sélection elle-même.

Implémentation

Vous devez remplir 4 fichiers implémentant chacun une des quatres solutions:

- `SelectByQuicksort.c`;
- `SelectByHeapsort.c`;
- `QuickSelect.c`
- `MedianOfMedians.c`

Ces quatre fichiers devront fournir la fonction `select` telle que définie dans le fichier `Select.h` (fourni). Vous pouvez ajouter autant de fonctions supplémentaires que nécessaires (qui devront être déclarées statiquement).

Vos fichiers seront évalués sur les machines ms8xx avec la commande[2]:

```
gcc main.c QuickSelect.c Array.c --std=c99 --pedantic -Wall -Wextra -Wmissing-prototypes
-DNDEBUG -lm -o test
```

Un fichier `Makefile` est fourni qui compile vos quatre fichiers de cette manière et lance un test sur les trois types de tableaux précisés ci-dessous pour le calcul du 10-ième centile.

Analyse théorique

Dans le rapport, on vous demande de répondre aux questions suivantes:

1. Donnez dans une table les complexités en temps et en espace dans le pire cas et le meilleur cas des 4 implémentations en fonction de la taille `N` du tableau. Justifiez brièvement les complexités en espace de `Quickselect` et `Median-Of-Medians` et les complexités en temps de `Quickselect`.
2. En adaptant l'analyse de complexité du cas moyen du `Quicksort` vue au cours et sous les mêmes hypothèses, montrez que la complexité moyenne de `Quickselect` est $\Theta(N)$ si on suppose que la valeur de k fournie en argument est une valeur aléatoire choisie dans $\{1, \ldots, N\}$.

[1] Voir ici par exemple: `https://en.wikipedia.org/wiki/Median_of_medians`

[2] En substituant adéquatement `QuickSelect.c` par l'implémentation de l'algorithme que nous voulons tester.

Expérimentations

1. Mesurez les temps de calcul des 4 algorithmes dans le cas de tableaux aléatoires et de tableaux ordonnés de manière croissante et décroissante, pour la recherche de la médiane ($\texttt{k} = \lfloor \texttt{N}/2 \rfloor$ et du 10-ième centile ($\texttt{k} = \lfloor \texttt{N}/10 \rfloor$). Reportez ces temps de calcul dans deux tables (une pour chaque valeur de k) au format ci-dessous:

Type de liste	aléatoire			croissante			décroissante		
Taille	10^4	10^5	10^6	10^4	10^5	10^6	10^4	10^5	10^6
SelectByQuicksort SelectByHeapsort Quickselect MedianOfMedians									

2. Commentez ces résultats. Pour chaque type de tableau:
   - comparez l'évolution des temps de calcul en fonction de la taille du tableau aux complexités théoriques,
   - commentez l'ordre relatif des différents algorithmes,
   - discutez de l'impact du paramètre `k`.

Remarques:

- Les fonctions pour générer les tableaux vous sont fournies dans le fichier `Array.c`.
- Les temps reportés doivent être des temps moyens établis sur base de 10 expériences au moins.
- Si vous n'avez pas implémenté la version standard des algorithmes vue au cours (en particulier dans le cas du `Quicksort` et de la fonction `Partition`), précisez dans vos commentaires les modifications que vous avez apportées, si elles ont une influence sur les résultats empiriques.

Date de remise et soumission

Le projet est à réaliser ***en binôme*** pour le ***jeudi 24 mars 2022 à 23h59*** au plus tard. Le projet est à remettre via la plateforme de soumission de Montefiore: `http://submit.montefiore.ulg.ac.be/`.

Il doit être rendu sous la forme d'une archive `tar.gz` contenant à sa racine:

1. votre rapport (5 pages maximum) au format PDF et nommé `rapport.pdf`. Soyez bref mais précis et respectez bien la numération des (sous-)questions;
2. les quatre fichiers `.c` précisés plus haut.

Respectez bien les extensions de fichiers ainsi que les noms des fichier `*.c` (en ce compris la casse). N'incluez aucun fichier supplémentaire.

Un projet non rendu à temps recevra une cote globale nulle. En cas de plagiat[3] avéré, l'étudiant se verra affecter une cote nulle à l'ensemble des projets.

Les critères de correction sont précisés sur Ecampus.

Bon travail !

[3] Des tests anti-plagiat seront réalisés.