

Structures de données et algorithmes

Projet de seconde session 2018-2019:

Résolution de problèmes

Pierre GEURTS – Jean-Michel BEGON – Romain MORMONT

9 juillet 2019

L'objectif du projet est de vous exercer à l'utilisation des techniques de résolution de problèmes. Les deux problèmes visés concernent des séquences d'objets :

Une séquence S de taille m , $S = \langle x_0, \dots, x_{m-1} \rangle$, est une suite ordonnée d'objets. Nous représenterons les séquences par des tableaux en C.

Les objets sont quelconques et la *seule* opération à notre disposition est le test d'égalité¹.

1 Élément majoritaire

Un élément x est majoritaire dans une séquence S de taille m s'il est présent à *plus de* $\frac{m}{2}$ positions :

$$|\{x_i \in S \mid x_i = x\}| > \frac{m}{2}$$

Par exemple, l'élément majoritaire de la séquence $S = \langle a, b, a, c, a \rangle$ est a alors que la séquence $S = \langle a, b, a, c \rangle$ ne contient pas d'élément majoritaire (a n'est pas présent strictement plus que deux fois). On cherche à écrire une fonction `getMajEl(S)` renvoyant l'élément majoritaire de S si un tel élément existe, `NULL` sinon.

Analyse théorique

Dans le rapport, nous vous demandons de répondre aux questions suivantes :

- Décrivez une solution par force brute pour trouver l'élément majoritaire.
- Donnez le pseudo-code d'une solution plus efficace basée sur le principe du diviser-pour-régner.
- Analysez la complexité au pire cas de ces deux solutions.

Implémentation

Nous vous demandons d'implémenter une fonction `getMajElDC` répondant à l'interface décrite dans le fichier `Problem.h` et implémentant la recherche de l'élément majoritaire en utilisant la solution par diviser-pour-régner développée au point (b) ci-dessus.

1. Une fonction `getCode` est fournie dans le fichier `Element.c`. Elle *ne peut pas* être utilisée pour d'autres raisons que l'affichage. Seule la fonction `areEqual` peut être utilisée au sein des algorithmes du projet.

Analyse empirique

Dans le rapport, répondez aux questions suivantes sur base de votre implémentation :

- (a) Tracez une courbe de l'évolution des temps de calcul en fonction de la taille du tableau pour des tableaux générés aléatoirement et ensuite des tableaux générés de manière à contenir un élément majoritaire. Des fonctions vous sont fournies pour ça dans les fichiers `Element.c/h`.
Remarques : Choisissez les tailles de tableaux et le nombre de répétitions de l'expérience de manière appropriée. Attention à ne pas prendre en compte l'étape de génération des tableaux dans les temps de calcul.
- (b) Comparez les courbes observées avec les résultats de l'analyse de complexité théorique

2 Coupure minimum

Soit une séquence d'objets S . On cherche à partitionner la séquence S en un nombre minimum de sous-séquences contiguës telles que chaque sous-séquence démarre et se termine par le même objet.

Une sous-séquence contiguë d'une séquence $S = \langle x_0, \dots, x_{m-1} \rangle$ est une séquence $\langle x_i, x_{i+1}, \dots, x_{i+k} \rangle$ d'éléments de S où $0 \leq i \leq i+k < m$. Une partition d'une séquence est une séquence de sous-séquences telle que la concaténation des sous-séquences redonne la séquence d'origine.

Par exemple, la séquence $S = \langle i, c, g, h, b, c, d, e, f, e, f, d, b \rangle$ admet la partition

$$\langle \langle i \rangle, \langle c, g, h, b, c \rangle, \langle d, e, f, e, f, d \rangle, \langle b \rangle \rangle$$

En outre, cette partition est telle que chaque sous-séquence démarre et termine par le même objet. Cette partition est optimale puisqu'il n'est pas possible de découper S avec moins de sous-séquences en conservant l'égalité du premier et dernier élément.

Analyse

Dans le rapport, nous vous demandons de répondre aux questions suivantes :

- (a) Donnez une formulation mathématique récursive pour le nombre minimum $M(i)$ de sous-séquences pour partitionner la séquence $S = \langle x_0, \dots, x_{i-1} \rangle$. Précisez bien le(s) cas de base.
- (b) En déduire le pseudo-code d'un algorithme *efficace* `MINCUT(S)` renvoyant le nombre minimum de sous-séquences pour partitionner S .
- (c) Analysez la complexité de votre algorithme.
- (d) Complétez cet algorithme pour qu'il affiche la position du premier élément de chaque sous-séquence d'une solution.

Implémentation

Nous vous demandons d'implémenter une fonction `getMinCutDP` répondant à l'interface décrite dans le fichier `Problem.h` et qui implémente la recherche de la partition optimale par programmation dynamique.

Analyse empirique

Dans le rapport, répondez aux questions suivantes sur base de votre implémentation :

- (a) Tracez une courbe de l'évolution des temps de calcul en fonction de la taille de la séquence pour des séquences générées à l'aide de la fonction `createFewUniquesSequence` fournie dans les fichiers `Element.c/h`. *Remarques : Choisissez les tailles de tableaux, le nombre de répétitions de l'expérience, et le nombre d'éléments uniques de manière appropriée. Attention à ne pas prendre en compte l'étape de génération des tableaux dans les temps de calcul.*
- (b) Comparez la courbe observée avec les résultats de l'analyse de complexité théorique

3 Deadline et soumission

Le projet est à réaliser **individuellement** pour le **18 août 2019, 23h59** au plus tard. Il doit être soumis via la plateforme de soumission (<http://submit.montefiore.ulg.ac.be/>).

Le projet doit être rendu sous la forme d'une archive `tar.gz` contenant :

- (a) Votre rapport (4 pages maximum) au format PDF. Soyez bref mais précis et respectez bien la numérotation des (sous-)questions.
- (b) Le fichier `Problem.c` reprenant votre implémentation des deux fonctions `getMajELDC` et `getMinCutDP`.

L'interface de ces deux fonctions est définie dans un fichier `Problem.h`. Nous vous fournissons également les fichiers `Element.h`, `Element.c` et `main.c`. Les deux premiers reprennent l'interface et l'implémentation des éléments ainsi que des séquences d'éléments. Le dernier est un petit programme qui affiche les résultats des deux fonctions à implémenter.

Vos fichiers seront compilés sur les machines `ms8xx` avec la commande :

```
gcc main.c Problem.c Element.c --std=c99 --pedantic -Wall -Wextra -Wmissing-prototypes DNDEBUG -o seq
```

Un projet non rendu à temps recevra automatiquement une cote nulle. En cas de plagiat avéré l'étudiant se verra affecter une cote nulle également. Les critères de correction sont précisés sur la page web des projets.

Bon travail !