Structures de données et algorithmes

Projet de seconde session 2014-2015:
Résolution de problèmes

Pierre Geurts – Jean-Michel Begon – Romain Mormont

2 juillet 2015

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, $S = \langle x_0, \ldots, x_{m-1} \rangle$, 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'égalité.

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 :

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

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 un élément majoritaire de S si un tel élément existe, $NULL$ sinon.

Analyse

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

(a) Décrivez une solution par force brute pour trouver l'élément majoritaire.

(b) Donnez le pseudo-code d'une solution plus efficace basée sur le principe du diviser-pour-régner.

(c) 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.

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 $S = \langle x_0, \ldots, x_{m-1} \rangle$ est une séquence $\langle x_i, x_{i+1}, \ldots, x_{i+k} \rangle$ 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 récursive pour le nombre minimum $M(i)$ de sous-séquences pour partitionner la séquence $S = \langle x_0, \ldots, x_{i-1} \rangle$. 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.

3 Deadline et soumission

Le projet est à réaliser **individuellement** pour le **14 aout 2015, 23h59** au plus tard. Il doit être soumis via la plateforme *cicada*(http://cicada.run.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 numération 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 -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 à l'ensemble du projet. Les critères de correction sont précisés sur la page web des projets.

Bon travail !