SAE S1.02 – Comparaison d'approches algorithmiques

Objectif

L'objectif principal est la mise en œuvre de différents algorithmes pour résoudre un même problème. Par comparaison d'approches algorithmiques distinctes, il est demandé de mettre en évidence à la fois de façon théorique mais également par des mesures empiriques la rapidité d'exécution des différents algorithmes.

Il est fortement conseillé de s'inspirer pour cela des techniques de comparaison enseignées dans la ressource R1.01.P2.

Modalités pratiques

Le projet se déroule en binôme.

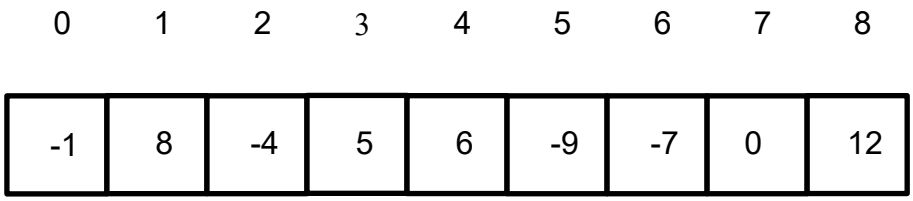
Début de la SAE : semaine 48 (29 novembre).

Date limite du rendu : **vendredi 14 janvier 2022 à 23h55 au + tard** (**attention** : -2 points de malus par jour de retard, définitivement 0/20 à partir du lundi 17/01 à 0h).

Plus grande somme

Le problème à résoudre est relativement simple à comprendre mais sa mise en œuvre sous forme algorithmique peut prendre des formes radicalement différentes : c'est précisément le but de ce projet !

On considère, comme exemple, le tableau d'entiers (positifs ou négatifs) suivant :



Problème à résoudre : on désire trouver dans ce tableau la sous-séquence dont la somme des entiers est maximale. Une sous-séquence est définie comme un ensemble de cases adjacentes. Il s'agit donc de trouver la somme maximale en additionnant les entiers d'une sous-séquence d'un tableau. Il vous est demandé de fournir la valeur de cette somme maximale ainsi que les indices qui délimitent la sous-séquence. Dans le cas de l'exemple ci-dessus, la plus grande somme est 15 et la sous-séquence correspondante se trouve entre les indices 1 et 4.

Remarques :

- Il s'agit de trouver le maximum en additionnant les entiers d'une sous-séquence d'un tableau. On demande de fournir ce maximum et les indices qui délimitent la sous-séquence. Elle peut avoir une taille allant de zéro case (sous-séquence vide) jusqu'à l'entièreté du tableau (sous-séquence de taille maximum). Si plusieurs sous-séquences ont un même maximum alors renvoyer celle dont la longueur est la plus petite.

- Par convention, lorsque le tableau ne contient que des nombres strictement négatifs, la somme maximale vaut zéro et la sous-séquence est vide.

Travail demandé

Ecrire 4 algorithmes et les tester

Il y a quatre algorithmes à écrire correspondant à quatre efficacités bien distinctes :

- un algorithme « plusGrdeSomme1 » en $\Theta(n^3)$
- un algorithme « plusGrdeSomme2 » en $\Theta(n^2)$
- un algorithme « plusGrdeSomme3 » en $\Theta(n\log_2 n)$
- un algorithme « plusGrdeSomme4 » en $\Theta(n)$

Chaque algorithme doit être testé sur des exemples simples et facilement vérifiables (*void testPlusGrdeSomme1()*, *void testPlusGrdeSomme2()*, *etc*).

Evaluer l'efficacité des 4 algorithmes

Pour chaque algorithme écrit, il ne suffit pas d'affirmer une efficacité Θ au hasard il faut avant tout la prouver et la comprendre.

C'est essentiellement sur la maîtrise de vos propos « scientifiques » qui montrent cette efficacité que votre travail sera évalué !

Pour évaluer cette efficacité, vous devez pour chacun des 4 algorithmes :

1. Calculer le nombre d'opérations exactes en fonction de *n* (la taille du tableau) et en déduire son efficacité théorique pour *n* grand (exactement comme pratiqué lors des TDs).

2. Confirmer cette efficacité de façon empirique sur des tableaux de grandes tailles remplis de valeurs aléatoires positives et négatives. Ensuite, procéder comme pour le TP2 avec un compteur *cpt* qui sera incrémenté de 1 à chaque passage dans la boucle la plus imbriquée. Quel que soit le contenu du tableau (rempli de valeurs aléatoires !) de grande taille, vous devez observer que :
   - pour « plusGrdeSomme1 » : $cpt/n^3 = k1$ (k1 étant une constante que vous devez déterminer)
   - pour « plusGrdeSomme2 » : $cpt/n^2 = k2$ (k2 étant une constante que vous devez déterminer)
   - pour « plusGrdeSomme3 » : $cpt/n\log_2 n = k3$ (k3 étant une constante que vous devez déterminer)
   - pour « plusGrdeSomme4 » : $cpt/n = k4$ (k4 étant une constante que vous devez déterminer)

3. Représenter graphiquement les courbes d'efficacité obtenues en mesurant empiriquement le temps d'exécution de vos algorithmes (*abscisse = n* la taille du tableau, *ordonnée = temps d'exécution de l'algorithme*).

Rendu de votre travail

Votre travail sera rendu sur Moodle sous forme d'une archive *nomBinôme1_nomBinôme2.zip* (pas de *.rar* !) qui contiendra plusieurs fichiers :

- Votre source java (1 classe *PlusGrandeSomme.java*), soigné, documenté (*javaDoc*) qui doit non seulement contenir les 4 algorithmes mais également les tests associés et les tests d'efficacité. **Le nom des variables et la javaDoc doivent être écrits en anglais**.
- Un diaporama au format PDF, qui présente :
  - brièvement les objectifs du travail,
  - le code des 4 algorithmes (ou les parties de codes les plus significatives),
  - l'explication mathématique des valeurs obtenues pour les 4 constantes k1, k2, k3, k4,
  - un tableau récapitulatif permettant la comparaison de l'efficacité des 4 algorithmes,
  - tous les graphiques qui représentent les courbes d'efficacité des algorithmes.

Une partie de ce diaporama sera présenté lors de votre évaluation orale.

Evaluation de votre travail

- Votre source java est évalué selon les critères habituels : clarté de codage, documentation embarquée javaDoc, qualité des tests et de la mise en évidence des efficacités théoriques.

- **Un oral de 10 minutes par binôme aura lieu en semaine 03 (le 17 janvier)**. Lors de cet oral, vous utiliserez votre diaporama comme support. On vous demandera de présenter un algorithme (parmi ceux que vous avez réussi à développer) tiré au hasard. Les deux étudiants du binôme doivent intervenir obligatoirement. Vos explications doivent montrer que vous avez compris à la fois le principe de l'algorithme et la notion d'efficacité. Il vous sera également demandé le pourcentage du travail de chacun dans la réalisation de la SAE (50% chacun signifie que le travail a été parfaitement équitable dans le binôme).