

# TP 5.5 optionnel : méthodes de tri, analyse numérique

Informatique Fondamentale (IF1)

Semaine du 4 Décembre 2006

## 1 Algorithmes de tri

Les méthodes demandées dans cette partie prendront un tableau  $\mathbf{t}$  d'entiers en argument.

**1. Minimum** Écrivez une méthode qui détermine le plus petit élément de  $\mathbf{t}$ .

**Tri par insertion** Le tri par insertion procède en séparant un tableau  $\mathbf{t}$  en deux parties : la partie de gauche, qui est déjà triée, et la partie de droite, qui ne l'est pas encore. Cette séparation est matérialisée par une variable  $i$  qui indique la taille de la partie triée.

Initialement, la partie de gauche est vide ( $i$  vaut 0). À chaque itération, on détermine l'indice  $j \leq i$  auquel doit être inséré le premier élément  $e = \mathbf{t}[i]$  de la partie non-triée ; on décale ensuite les éléments  $\mathbf{t}[j]$  à  $\mathbf{t}[i - 1]$  du tableau d'un cran, et on insère  $e$  à la position  $j$ . On incrémente ensuite  $i$ .

L'algorithme se termine lorsque  $i$  est égal à la taille du tableau.

**2.** Écrivez une méthode

```
int position(int e, int[] t, int i)
```

qui retourne le plus petit  $j < i$  tel que  $\mathbf{t}[j] \geq e$ , ou  $i$  si tous les  $\mathbf{t}[j]$  pour  $j < i$  sont inférieurs à  $e$ .

**3.** Écrivez une méthode `triInsertion(t)` qui trie par insertion le tableau  $\mathbf{t}$ .

## 2 Analyse numérique

### 2.1 Développements de Taylor-MacLaurin

Le développement de Taylor de la fonction exponentielle à l'ordre  $n$  est la somme infinie (appelée aussi série entière)

$$1 + x + x^2/2 + x^3/6 + \dots + x^n/n! + \dots$$

**4.** Montrez que pour  $2x < n$ , on a  $\left| \sum_{k=0}^{\infty} x^k/k! \right| \leq x^n/n!$ , déduisez-en que

$$\left| \exp(x) - \sum_{k=0}^n x^k/k! \right| \leq x^n/n!.$$

Vous pouvez admettre ce résultat sans démonstration si vous êtes paresseux.

5. Écrivez une méthode qui prend deux arguments  $x$  et  $\varepsilon$  et qui renvoie  $\exp(x)$  avec une erreur inférieure à  $\varepsilon$ .
6. Calculez  $\exp(5)$  avec une erreur inférieure à  $10^{-4}$ .
7. Écrivez un programme qui calcule  $10^8$  fois  $\exp(5)$  à l'aide de la méthode de la question 2, et un autre programme qui fait la même chose à l'aide de `Math.exp`. Comparez ensuite la vitesse d'exécution des deux programmes avec la commande `time` du shell.

## 2.2 Méthode de Newton-Raphson

La méthode de Newton-Raphson, est un algorithme efficace pour trouver des approximations d'une racine d'une fonction continûment dérivable  $f$ , c'est-à-dire approcher  $\alpha$  tel que  $f(\alpha) = 0$ .

On commence par choisir une valeur arbitraire  $x_0$ . Ensuite, on définit par récurrence la suite suivante :

$$x_{n+1} = x_n - (f(x_n)/f'(x_n))$$

où  $f'$  désigne la dérivée de la fonction  $f$ . La suite  $(x_n)$  va converger vers  $\alpha$ . De plus, la convergence est quadratique, ce qui signifie intuitivement que le nombre de chiffres corrects est approximativement doublé à chaque étape.

8. Programmez une méthode correspondante à cet algorithme dans le cas où  $f(x) = \sqrt{x} - 3$  : on se donne  $erreur > 0$ ,  $N > 0$  et  $x_0$ , puis on itère le calcul des terme de la suite  $(x_n)$  tant que  $|x_n - x_{n-1}| > erreur$  et  $n < N$ .
9. Calculez une valeur approchée du zéro de  $f$ .