TD #5 – DEV

3 octobre 2022
Erwan Kerrien
Abdel Bourjij

1 Calculs de complexité

Pour chaque algorithme suivant, donnez la valeur finale de S puis donnez la complexité en $O(.)$, en fonction de N. On rappelle les formules utiles :

$$\sum_{k=1}^{N} 1 = N, \qquad \sum_{k=1}^{N} k = \frac{N(N+1)}{2}, \qquad \sum_{k=1}^{N} k^2 = \frac{N(N+1)(2N+1)}{6}$$

(a)

Entrée : N : Entier
Sortie : S : Entier
V. I. : i :Entier
Début
  $S \leftarrow 0$
  Pour i **allant de** *1* **à** N **faire**
    $S \leftarrow S + 1$
  FinPour
Fin

(b)

Entrée : N : Entier
Sortie : S : Entier
V. I. : i, j :Entier
Début
  $S \leftarrow 0$
  Pour i **allant de** *1* **à** N **faire**
    Pour j **allant de** *1* **à** N **faire**
      $S \leftarrow S + 1$
    FinPour
  FinPour
Fin

(c)

Entrée : N : Entier
Sortie : S : Entier
V. I. : i, j :Entier
Début
  $S \leftarrow 0$
  Pour i **allant de** *1* **à** N **faire**
    Pour j **allant de** *1* **à** $N * N$ **faire**
      $S \leftarrow S + 1$
    FinPour
  FinPour
Fin

(d)

Entrée : N : Entier
Sortie : S : Entier
V. I. : i, j :Entier
Début
  $S \leftarrow 0$
  Pour i **allant de** *1* **à** N **faire**
    Pour j **allant de** *1* **à** i **faire**
      $S \leftarrow S + 1$
    FinPour
  FinPour
Fin

(e)

Entrée : N : Entier
Sortie : S : Entier
V. I. : i, j, k :Entier
Début
  $S \leftarrow 0$
  Pour i **allant de** *1* **à** N **faire**
    Pour j **allant de** *1* **à** i **faire**
      Pour k **allant de** j **à** i **faire**
        $S \leftarrow S + 1$
      FinPour
    FinPour
  FinPour
Fin

2 Chaînes de caractères

En C, une chaîne de caractères est représentée par un tableau de caractères dont le dernier élément a pour code ASCII 0, que l'on représente par le caractère `'\0'`. Cette marque de fin par un caractère non imprimable permet de ne pas avoir à passer la taille de la chaîne de caractères en paramètre de fonction. Par exemple la chaîne de caractères pour le mot «tableau» est stockée comme le tableau `['t','a','b','l','e','a','u','\0']`. Aucune fonction de base ne donne donc la longueur d'un tableau en C. Les fonctions `strlen` et `strcmp` correspondent cependant à de vraies fonctions C disponibles dans la bibliothèque standard (qui nécessite une instruction `#include <string.h>` en début de code).

On supposera ici que l'instruction `Saisir(`S`)`, où S est une chaîne de caractères, produit en sortie une chaîne se terminant par `'\0'`.

1. Écrivez un algorithme qui calcule la longueur d'une chaîne de caractères `S`, saisie par l'utilisateur. Donnez sa complexité en fonction de la longueur de la chaîne (que l'on notera N).
2. Écrivez un algorithme qui compare deux chaînes de caractères `S1` et `S2`, entrées par l'utilisateur, et renvoie `Vrai` si `S1=S2` et `Faux` sinon. Donnez sa complexité. Est-il intéressant de débuter par un test sur les longueurs des chaînes, en considérant que la complexité pour calculer ces longueurs est la même que l'algorithme précédent (`S1` est différent de `S2` si elles sont de longueurs différentes) ?
3. Écrivez un algorithme qui compare deux chaînes de caractères `S1` et `S2`, saisies par l'utilisateur, selon leur ordre alphabétique et affiche : `avant` si `S1` est alphabétiquement avant `S2`, `apres` dans le cas contraire et `egal` si `S1=S2`. Donnez sa complexité.

3 Distances

On considère un tableau de réels T de taille n et on cherche à calculer le plus grand écart entre deux éléments de ce tableau, soit :

$$M = \max_{i,j}(T_i - T_j)$$

On supposera ici qu'on a une instruction `Init(`T`,`n`)` qui permet d'initialiser le contenu d'un tableau T (par exemple par saisie utilisateur) et renseigne sa taille n.

1. Écrivez un algorithme qui calcule M. Quelle en est la complexité ?
2. Même question pour $S = \sum_{i,j} (T_i - T_j)^2$

4 Polynômes

Un polynôme de degré n est de la forme $P(X) = p_0 + p_1X + p_2X^2 + \ldots + p_{n-1}X^{n-1} + p_nX^n = \sum_{i=0}^{n} p_iX^i$ avec $p_n \neq 0$. On peut donc le représenter sous la forme d'un tableau `P` qui contient les $n+1$ valeurs $(p_i)_{0 \leq i \leq n}$, tel que `P[i]`$=p_{i-1}$ pour tout i entre 1 et $n+1$ (le décalage est nécessaire pour stocker des indices négatifs ou nuls).

On supposera ici qu'on dispose d'une instruction `Init(`P`,`n`)` qui initialise les coefficients du polynome dans un tableau P et renseigne son degré n, par exemple via une saisie utilisateur.

(a) Écrivez un algorithme simple qui initialise un polynôme par un tableau `P`, ainsi qu'un entier `n`, et demande de saisir un réel `x`, puis calcule et affiche la valeur du polynôme $P(x)$. Donnez la complexité de cet algorithme en fonction de n.

(b) L'algorithme de Horner permet de minimiser le nombre de calculs nécessaires en remarquant l'identité suivante :

$$P(X) = p_0 + p_1X + p_2X^2 + \ldots p_nX^n = p_0 + X\left(p_1 + X\left(p_2 + \ldots X\left(p_{n-1} + Xp_n\right)\ldots\right)\right)$$

Réécrivez l'algorithme précédent qui calcule $P(x)$, en suivant la méthode de Horner. Donnez sa complexité.