Série d'exercices N^{o}7

ING1 – Algorithmique procédurale

Objet de la série : Récursivité, Diviser pour régner.

1 Algorithmes récursifs simples

Écrire des algorithmes récursifs pour les problèmes suivants :

1. Calcul de la suite de Fibonacci.
2. Inversion des éléments d'un tableau.
3. Recherche dans un tableau non trié.
4. Recherche dans un tableau trié.

2 Récursivités terminale et non terminale

Écrire des algorithmes récursifs terminaux pour les problèmes suivants :

1. Calcul de la suite de Fibonacci.
2. Approximation du nombre π par la formule d'Euler définie comme suit :

$$\frac{\pi^2}{6} = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \ldots + \frac{1}{n^2}$$

3 Analyse du tri rapide (d'après [1])

Dans cet exercice nous analysons l'algorithme du tri rapide (quicksort) présenté en annexe.

1. Faites tourner l'algorithme sur le tableau A = {2, 8, 7, 1, 3, 5, 6, 4}.
2. Prouvez la procédure Partition à l'aide de l'invariant de boucle suivant :
   — Au début de chaque itération de la boucle (**Pour** j **de** p **à** $r-1$) nous avons :
      — Pour $k = p, \ldots, i$, $A[k] \leq x$.
      — Pour $k = i+1, \ldots, j-1$, $A[k] > x$.
      — $A[r] = x$.
3. Calculez la complexité de la procédure Partition.

4. Donnez la récurrence définissant la complexité du tri rapide dans les trois cas suivants :
   (a) À chaque étape la partition conduit à ranger tous les élements du même côté du pivot. On admettra que ce cas est le plus défavorable.
   (b) À chaque étape la partition est équilibrée : elle range un nombre quasi-égal d'éléments des deux côtés du pivot. On admettra que ce cas est le plus favorable.
   (c) À chaque étape la partition conduit à ranger $\frac{9}{10}$ des éléments d'un côté et $\frac{1}{10}$ de l'autre.
5. Évaluez le nombre d'opérations du tri rapide dans chacun des 3 cas donnés ci-dessus.
6. Quelle est la complexité au pire des cas de cet algorithme ?
7. À quelle complexité moyenne devons-nous nous attendre ? Pourquoi ?

4 Annexe : Description du tri rapide

Le tri rapide est basé sur le paradigme 'Diviser pour régner'. Pour un sous-tableau $A[p..r]$ son fonctionnement est le suivant :

Diviser : $A[p..r]$ est arrangé de manière à avoir un indice q tel que chaque élément de $A[p..q-1]$ est inférieur ou égal à un élément donné $A[q]$ qui lui même est strictement inférieur à chaque élément de $A[q+1..r]$. $A[q]$ s'appelle le **pivot**.

Régner : les deux tableaux $A[p..q-1]$ et $A[q+1..r]$ sont triés récursivement.

Combiner : Les deux sous-tableaux $A[p..q-1]$ et $A[q+1..r]$ étant triés, le tableau $A[p..r]$ l'est aussi. Il n'y a rien de plus à faire.

L'algorithme peut donc s'écrire de la manière suivante :

Algorithme Algo-Tri-Rapide
Debut
  ....
  Tri-Rapide(A, 1, n)
Fin

Procedure Tri-Rapide(E/S A : Tableau d'élément, p : entier, r : entier)
Variables
  q : entier
Debut
  Si (p < r) Alors
    q = Partition(A, p, r)

```
    Tri-Rapide(A, p, q-1)
    Tri-Rapide(A, q+1, r)
  FinSi
Fin
```

Fonction Partition(E/S A : Tableau d'élément, p : entier, r : entier) :Entier

Variables

  x : élément

  i,j : entier

Debut

  x = $A[r]$

  i = p - 1

  Pour j de p à r-1

    Si $A[j] \leq x$ Alors

      i = i + 1

      permuter($A[i]$, $A[j]$)

    FinSi

  FinPour

  permuter($A[i+1]$, $A[r]$)

  Retourner (i + 1)

Fin

Références

[1] Cormen, T. H., Leiserson, C., Rivest, R., & Stein, C. (2010). *Algorithmique*. Dunod.