

## TD n°4

Où il est question de balance, de crêpes et d'un monstre

### 1 Des pièces et une balance

Il est rare que l'on puisse prouver l'**optimalité** d'un algorithme de façon autre que «notre algorithme est linéaire en la taille des données, or il faut lire toutes les données pour conclure». Voilà un problème où une telle analyse est possible.

On a  $n$  pièces. On sait que l'une d'entre elle est fautive (elle est plus légère que les autres). On dispose d'une balance à plateaux.

#### Exercice 1

1. Combien de pesées faut-il faire pour trouver la fautive pièce parmi 4 pièces ? 8 ? 9 ?
2. Proposez un algorithme résolvant le problème
3. Montrez sa correction
4. Donnez sa complexité
5. Prouvez son optimalité

### 2 Le tri-crêpes (*pancake sorting*) un tri qui peut vous rendre riche

On dispose d'une pile de  $n$  crêpes dont les diamètres sont tous différents. Le problème consiste à trier la pile de manière à obtenir la crêpe la plus large en bas de la pile et la crêpe la moins large en haut de la pile. Pour éviter de trop manipuler les crêpes, on ne s'autorise qu'une seule opération : retourner les  $p$  premières crêpes du haut de la pile ( $1 \leq p \leq n$  et  $p$  peut changer d'un retournement à l'autre).

**Exercice 2** Proposez un algorithme qui trie les crêpes en faisant le moins de retournements possible. Dans le pire des cas, à combien êtes-vous du nombre *optimal* de retournements ?

### 3 Encore un tri

On a  $p$  listes triées (chaque liste est triée en ordre croissant). Elles contiennent  $n$  nombre en tout (la somme des longueurs des listes est  $n$ ).

**Exercice 3** Proposez un algorithme qui donne la liste des  $n$  nombres triés en  $O(n \log p)$ .

## 4 La fonction d'Ackermann

Soit  $A_i(a, b)$  la fonction  $\mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  définie par

$$A_i(a, b) = \begin{cases} b + 1 & \text{si } i = 0 \\ a & \text{si } b = 0 \text{ et } i = 1, \text{ ou si } b = 1 \text{ et } i > 1 \\ A_{i-1}(a, A_i(a, b-1)) & \text{sinon} \end{cases}$$

On l'appelle le *monstre d'Ackermann* pour une raison qui va vous apparaître.

### Exercice 4

- 0- Identifiez et donnez un nom à la fonction  $A_0(a, b)$
- 1- Identifiez et donnez un nom à la fonction  $A_1(a, b)$
- 2- Identifiez et donnez un nom à la fonction  $A_2(a, b)$
- 3- Identifiez et donnez un nom à la fonction  $A_3(a, b)$
- 4- Identifiez et donnez un nom à la fonction  $A_4(a, b)$

**Exercice 5** Quelle est la complexité asymptotique de calcul de  $A_i(a, b)$ ? Pour quelles valeurs un ordinateur *échouera* certainement à faire le calcul?

## 5 Complexité en moyenne

**Exercice 6** Soit  $n$  un entier strictement positif et soit  $T$  un tableau de  $n$  nombres distincts. On appelle *maximum provisoire* de  $T$  tout indice  $i$  dans  $[1, n]$  tel que, pour tout  $j$  dans  $[1, i-1]$ ,  $T[i]$  est plus grand que  $T[j]$ .

On veut calculer le nombre moyen de maxima provisoires, sur tous les tableaux  $T$  qui sont des permutations de  $[1, n]$  (*i.e.*, des bijections de  $[1, n]$  dans  $[1, n]$ ). C'est le nombre moyen d'affectations à faire dans l'algorithme classique de recherche du maximum.

On appelle  $P_{n,k}$  le nombre de permutations de  $[1, n]$  qui ont  $k$  maxima provisoires. Montrer que l'on a les relations :

- $P_{1,1} = 1$ , et  $P_{1,k} = 0$  pour tout  $k \neq 1$ ;
- $P_{n,k} = P_{n-1,k-1} + (n-1)P_{n-1,k}$ .

Soit  $S_n$  le nombre moyen de maxima provisoires d'une permutation de  $[1, n]$ . Montrer que :

$$S_n = H_n = \sum_{k=1}^n \frac{1}{k}.$$

Exprimer  $S_n$  en  $\Theta$  d'une fonction en  $n$ .