TD 1

RÉCURSION (ET BASES DE DONNÉES)

Mode d'emploi : commencez par traiter tous les exercices « fléchés ». Une fois que c'est fait, traitez les autres dans la limite du temps disponible et dans l'ordre que vous voulez.

1 Bases de données

Pour commencer, allez sur le site bianquis.org et récupérez le fichier `bac_1` qui contient une base de données de résultats du baccalauréat en France. Il y a une seule table, `resultats`, dont le schéma est le suivant :

```
resultats(nomL, ville, public, tauxL, tauxES, tauxS, tauxGlobal, academie, departement)

nomL : texte (nom du lycée)
ville : texte (ville du lycée)
public : entier (1 pour public et 0 pour privé)
tauxL : entier (taux de réussite au bac dans la filière L)
tauxES : entier (taux de réussite au bac dans la filière ES)
tauxS : entier (taux de réussite au bac dans la filière S)
tauxGlobal : entier (taux global de réussite au bac)
academie : texte (nom de l'académie du lycée)
departement : texte (nom du département du lycée)
```

⇝ Exercice 1

1. *Combien y a-t-il de lycées dans la base ?*
2. *Et combien d'académies ?*
3. *Donner la liste des lycées de Mâcon.*
4. *Et les lycées publics de Lyon ?* La fonction `instr(s, t)` renvoie 0 si `t` n'est pas une sous-chaîne de `s`, un entier strictement positif sinon).
5. *Classer les lycées du Rhône par ordre décroissant de taux de réussite global (on donnera le nom, la ville et le taux de réussite).*
6. *Quels lycées publics (en France) ont obtenu 100% de réussite ?*

2 Exercices élémentaires

⇝ Exercice 2 — Suite récurrente d'ordre 1

Écrire des fonctions récursives calculant le n-ème terme des suites suivantes :

1. $u_n = n!$
2. $v_0 = 3$, $v_{n+1} = 2v_n + 1$
3. $w_0 = 5$, $w_{n+1} = 2nw_n + 3$ *(vous pouvez aussi l'écrire en itératif et vérifier que vous trouvez bien $w_3 = 39$ dans les deux cas).*
4. $t_n = \sum_{k=1}^{n} \frac{1}{k^2}$

⇝ Exercice 3 — Exponentiation rapide

1. *Écrire une fonction récursive* `expo(x, n)` *calculant « bêtement » x^n.*
2. *Écrire une fonction récursive* `expo_rapide(x, n)` *calculant x^n par exponentiation rapide, c'est-à-dire en utilisant les relations suivantes :*
   - $x^{2n} = x(^n)^2$ *(ou $x^{2n} = (x^2)^n$, au choix)*
   - $x^{2n+1} = x \cdot x^{2n}$

⇝ Exercice 4 — Écriture en base b

On rappelle que si b est un entier supérieur ou égal à 2, tout entier naturel x non nul s'écrit de manière unique $x = \sum_{k=0}^{n-1} a_k b^k = \overline{a_{n-1} \dots a_0}^b$: on parle d'écriture en base b.

Par exemple, on a $\overline{71}^{10} = \overline{1000111}^2 = \overline{2122}^3$.

1. *Écrire une fonction récursive* `chiffres(x, b)` *qui renvoie la liste des chiffres de x en base b (chiffre de poids fort en premier).*

```
In [73]: chiffres(71, 10)
Out[73]: [7, 1]
In [74]: chiffres(71, 2)
Out[74]: [1, 0, 0, 0, 1, 1, 1]
In [75]: chiffres(71, 3)
Out[75]: [2, 1, 2, 2]
```

2. *Écrire une fonction récursive* `valeur(liste, b)` *qui prend en entrée une liste de chiffres (et une base) et renvoie l'entier correspondant :*

```
In [80]: valeur([7, 1], 10)
Out[80]: 71
In [81]: valeur([2, 1, 2, 2], 3)
Out[81]: 71
```

3. *Écrire une fonction* `convertit(liste, b1, b2)` *qui prend en entrée un entier représenté par la liste de ses chiffres en base b_1 et renvoie la liste de ses chiffres en base b_2.*

```
In [83]: convertit([2, 1, 2, 2], 3, 2)
Out[83]: [1, 0, 0, 0, 1, 1, 1]
```

3 Tri fusion

⇝ Exercice 5

1. *Écrire une fonction* `fusionne(s, t)` *qui prend en entrée deux listes s et t (supposées triées) et renvoie la liste (triée) obtenue en fusionnant s et t. Cette fonction doit impérativement s'exécuter en temps* $O(|s|+|t|)$.
2. *Écrire alors une fonction* `tri_fusion(s)` *qui trie la liste s par la méthode du tri fusion.*

Exercice 6

1. *Écrire un tri quadratique de votre choix (insertion, sélection, bulles…).*
2. *Comparer l'efficacité des deux tris que vous avez écrits sur des listes de différentes longueurs. Idéalement, on voudrait un joli graphique…*

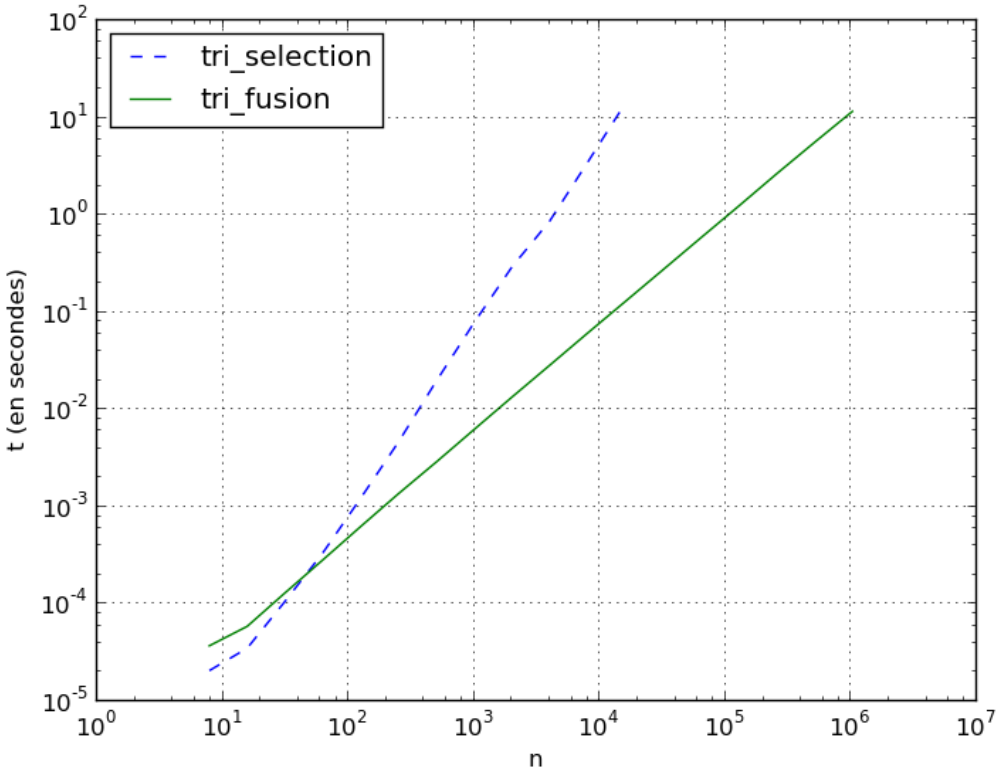

Figure 1.1 – Comparaison de temps d'exécution, tri fusion et tri sélection

4 Si vous avez le temps

Exercice 7 **Évaluation d'expressions post-fixes**

1. *Traduire les expressions suivantes en notation post-fixe :*
   - **a.** $(2 + 3) \times (4 - 6)$
   - **b.** $1 + ((3 + 4) - (5 \times 6))$

c. $(3+2)\times 5+4$

2. *Traduire les expressions suivantes en notation infixe (*i.e. *usuelle) :*

a. `1 2 + 3 *`

b. `1 3 5 4 + * 7 - *`

3. *Écrire une fonction* `evalue(expr)` *qui calcule la valeur d'une expression post-fixe. L'expression sera donnée sous la forme d'une chaîne de caractère (par exemple* `expr = '1 2 + 3 *'`*, et l'on se limitera aux expressions n'utilisant que* $+$*,* $\times$ *et* $-$*.*
On rappelle que :

a. *si* `s` *est une chaîne caractères représentant un entier, alors* `int(s)` *donne l'entier en question ;*

```
In [3]: s = '12'
In [4]: t = '53'
In [5]: s + t
Out[5]: '1253'
In [6]: int(s) + int(t)
Out[6]: 65
```

b. *si* `s` *est une chaîne de caractères,* `s.split()` *renvoie la liste de chaînes obtenue en coupant* `s` *à chaque espace*[a] *:*

```
In [7]: s = '12 x   3 bonjour   '
In [8]: s.split()
Out[8]: ['12', 'x', '3', 'bonjour']
```

a. à chaque bloc de caractères de *whitespace* en fait, donc espaces, retours à la ligne…

EXERCICE 8 — Tours de Hanoï

Dans le jeu des tours de Hanoï, on dispose de n disques percés (dont les tailles vont de 1 *à n) et de trois « piliers ». Au départ, les disques sont empilés sur le pilier de gauche (le plus grand en bas, le plus petit eu sommet), et le but est de tous les transférer sur le pilier de droite. Les règles sont les suivantes :*

- *à chaque étape, on ne peut déplacer qu'un seul disque ;*
- *on ne peut empiler un disque que sur un disque plus grand.*

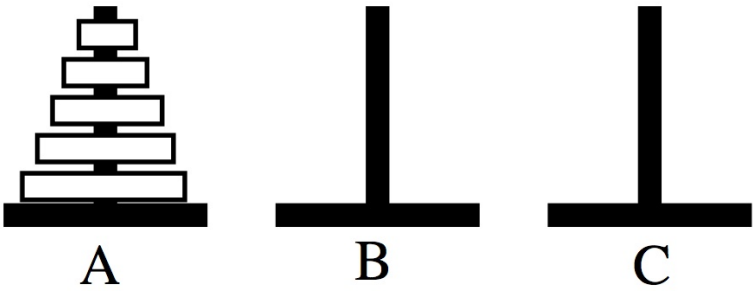

FIGURE 1.2 – Les disques sont dans la position initiale, dans la position finale ils doivent tous être sur le pilier C.

Écrire une fonction `hanoi(n)` *qui affiche la liste des déplacements à effectuer :*

```
In [65]: hanoi(3)
Déplacer un disque de 1 vers 3
Déplacer un disque de 1 vers 2
Déplacer un disque de 3 vers 2
Déplacer un disque de 1 vers 3
Déplacer un disque de 2 vers 1
Déplacer un disque de 2 vers 3
Déplacer un disque de 1 vers 3
```

Deux indications :

- *on utilisera une fonction auxiliaire* **récursive** `hanoi_aux(n, init, final, autre)` *qui affiche la liste des déplacements pour transférer n disques du pilier* `init` *vers le pilier* `final` *en supposant que le pilier* `autre` *ne contient au départ que des disques plus grands que tous ceux qu'il faut déplacer (ou qu'il est vide, ce qui revient au même) :*

```
In [66]: hanoi_aux(2, 3, 1, 2)
Déplacer un disque de 3 vers 2
Déplacer un disque de 3 vers 1
Déplacer un disque de 2 vers 1
```

- *plutôt que de concaténer des morceaux de chaînes de caractères avec des +, on pourra préférer utiliser la méthode* `format` *:*

```
In [67]: s = "Nous sommes à {}, dans le département numéro {}"
In [69]: s.format("Lyon", 69)
Out[69]: 'Nous sommes à Lyon, dans le département numéro 69'
```

EXERCICE 9 — Recherche par interpolation

On suppose maintenant que l'on dispose d'une liste triée, mais qu'en plus les valeurs des éléments sont assez équitablement répartis : autrement dit, si le minimum vaut 1000 *et la maximum* 2000*, on s'attend à ce que* 1800 *se trouve à peu près aux* $\frac{4}{5}$ *de la liste (s'il apparaît).*

Dans ce cas, on peut faire un peu mieux (en moyenne) qu'une recherche dichotomique classique : au lieu de partager la liste en deux parties de même longueur, on va couper à l'endroit approximatif où on s'attend à trouver x.

1. *Si l'on fait une recherche dans t entre les indices i et j en supposant que les valeurs sont uniformément réparties, où faut-il couper ?*
2. *Adapter le code de la recherche dichotomique en utilisant cette méthode.*
3. Un peu long : *mettre en évidence le gain apporté par la méthode avec interpolation quand la situation s'y prête bien.*

EXERCICE 10

On reprend la base de données du début.

1. *Quel est le lycée ayant obtenu le plus faible taux de réussite ?*
2. *Classer les académies par nombre décroissant de lycées ayant 100% de réussite.*
3. *Classer les académies par taux décroissant de lycées privés.*