ISM MPSI/PCSI, Informatique
Année 2024/2025

# TP Informatique 26

## Exercice 1

On complètera le fichier `fact.py` en ligne sur le site de la classe.

1. Écrire une fonction `fact1(n)` qui calcule $n!$ pour $n$ entier de manière itérative.
2. Écrire une fonction `prod(a,b)` qui calcule de $\prod_{k=a}^{b-1} k$ avec $a, b$ entiers de manière récursive suivant le paradigme « diviser pour régner » selon la décomposition suivante :
$$\prod_{k=a}^{b-1} k = \left(\prod_{k=a}^{c-1} k\right) \times \left(\prod_{k=c}^{b-1} k\right) \quad \text{avec} \quad c = \left\lfloor \frac{a+b}{2} \right\rfloor$$
Les cas de base à coder sont $b \leqslant a$ avec $\prod_{k=a}^{b-1} k = 1$ (convention) et $b = a+1$ avec $\prod_{k=a}^{b-1} k = a$.
3. En déduire l'écriture d'une fonction `fact2(n)` qui calcule récursivement $n!$ pour $n$ entier en suivant le paradigme « diviser pour régner ».
4. Tester les fonctions `fact1` et `fact2`.
5. On note $\mathrm{T}(a, b)$ le nombre de multiplications réalisées par `prod(a,b)`. Montrer par récurrence forte sur $n = b - a$ que $\mathrm{T}(a, b) = b - a - 1$ pour tout $b > a$.
6. En déduire que la fonction `fact2(n)` réalise le même nombre de multiplications que `fact1(n)` pour $n$ entier non nul.
7. Effectuer la comparaison temporelle des calculs de `fact1(n)` et `fact2(n)` pour `n` parcourant `range(1000,50000,1000)`.
8. Proposer une explication à la différence de performance observée entre les deux fonctions.

## Exercice 2

Soit `L` une liste d'entiers de taille $n$. Un élément de `L` est dit *strictement majoritaire* si son nombre d'occurrences est strictement supérieur à $n/2$.

1. Écrire une fonction `compter(L,x)` qui renvoie le nombre d'occurrences de `x` dans `L`.
2. Justifier que si `L` contient un élément strictement majoritaire, alors celui-ci est unique et qu'une des sous-listes `L[:n//2]` et `L[n//2:]` partage cet élément strictement majoritaire.
3. En déduire, en s'appuyant sur le paradigme « diviser pour régner », une fonction récursive `majo1(L)` qui renvoie `[0,-1]` si `L` ne contient pas d'élément strictement majoritaire et `[x,k]` sinon avec `x` l'élément strictement majoritaire et `k` son nombre d'occurrences. Tester la fonction `majo1`.
4. Améliorer éventuellement la fonction précédente (voir fichier `majo.py` pour scénarios de comparaison).