TD 11 : Tri à bulles

Exercice 1 (Récursivité et coefficient binomiaux)

Soit $k, n \in \mathbb{N}$ tel que $k \leqslant n$.

1. Ecrire une fonction `binom(k,n)` calculant $\binom{n}{k}$ en utilisant la fonction factorielle. On pourra utiliser la fonction factorielle du module math (`from math import factorial`). Attention, le nom de la fonction est en anglais.
2. Compter le nombre de multiplications (une division est comptée comme une multiplication).
3. Ecrire une fonction récursive calculant $\binom{n}{k}$ en utilisant $\binom{n}{k} = ... \times \binom{n}{k-1}$. Compter le nombres de multiplications.

On considère un tableau `T` **de taille fixe et homogène** que l'on souhaite trier par ordre croissant. La taille du tableau `T` est `N`.

Exercice 2 (Tri à bulle – Bubble sort)

On étudie dans ce TD le tri à bulle. Dans l'eau, une bulle remonte jusqu'à la surface. Le principe du tri à bulle est simple : on parcours le tableau **en échangeant systématiquement deux éléments consécutifs** s'ils sont mal triés. Ce qui fait qu'après le premier parcours du tableau, la plus grande valeur est forcément dans la dernière case du tableau à sa place. Le reste du tableau est a priori non trié, on recommence donc à le parcourir, en s'arrêtant à l'avant-dernier élément. A ce moment, les deux plus grandes valeurs sont à leur place dans le tableau. Et ainsi de suite.

1. Dessiner le principe de la méthode sur papier en faisant apparaître les 2 boucles mises en jeu : indice `i` pour la boucle externe et `j` pour la boucle interne.
2. Ecrire une fonction `tri_bulle(T)` qui modifie en place le tableau `T` (et donc ne renvoie rien)
3. Pouvez-vous prouver que votre algorithme termine ?
4. Pour la boucle interne, quel invariant pourrait-on choisir ? A quelle conclusion arrive-t-on à la fin de la boucle interne (lorsque la boucle externe en est à l'itération `i`) ?
5. Pour la boucle externe, à l'itération `i`, le tableau `T` est réparti en une partie triée et une partie non triée. Quel invariant de boucle prendre pour cette boucle externe ?
6. Complexité temporelle : quel tableau `T` choisir pour être dans le pire des cas (celui qui demandera le plus de comparaison et le plus d'échange) ? On utilise le symbole Θ pour décrire une majoration asymptotique du nombre de comparaisons et d'échange, lors que la taille du tableau `N` devient de plus en plus grande. A l'aide de cette notation, évaluer le nombre de comparaisons et d'échange dans le pire des cas, dans le meilleur des cas et en moyenne (on tire aléatoirement les valeurs du tableau `T`).
7. Complexité spatiale (en mémoire) : évaluer la complexité spatiale de votre algorithme en utilisant la notation vue ci-dessus.

Exercice 3 (Tri par comptage)

Le tri par comptage d'un tableau `T` de taille `N` n'est pas un tri par comparaison, comme le tri à bulle. Les tris par comparaison n'ont pas d'aucune connaissance particulière sur les données à trier, ce qui n'est pas le cas du tri par comptage qui présuppose qu'**on connaisse un majorant strict** des données noté `m` : $\forall x \in T, x < m$.
On se restreint ici à trier des valeurs entières pour comprendre le concept.

1. Ecrire une fonction `comptage(T,m)` qui renvoie un tableau de taille `m` dont le k-ième élément désigne le nombre d'occurrence de `k` dans `T`.
2. Ecrire une fonction `tri_comptage(T,m)` qui trie en place le tableau `T` en utilisant la fonction `comptage`.
3. Tester avec un tableau contenant des entiers aléatoirement tirés dans l'intervalle [0,10] en utilisant le module `random`.
4. Evaluer la complexité temporelle de cet algorithme.
5. Evaluer la complexité spatiale de cet algorithme.
6. Conclure quant à l'intérêt de ce tri.

Exercice 4 (Défi : tri à bulle, améliorations)

1. Ecrire une fonction `tri_bulle2(T)` avec l'amélioration consistant à ne pas continuer à balayer le tableau si aucun échange n'a été fait lors d'un parcours.
2. Ecrire une fonction `tri_bulle3(T)` qui n'écrit dans le tableau que lorsque la bulle a fini de remonter. Cela évite des lectures-écritures inutiles en mémoire.