TP ALGO n°1
Complexité

Partie I

Exercice 1

On considère l'algorithme ci-dessous où *n* est un entier.

```
Pour i de 1 à n
        Pour j de i à n
                Afficher ("Salut")
```

Calculez (comme une fonction de n) le nombre de fois où la chaîne de caractères "Salut" est affichée. En déduire la complexité en notation asymptotique.

Exercice 2

On considère l'algorithme ci-dessous où *n* est un entier.

```
Pour i de 1 à n
        Pour j de 1 à n
                Afficher ("Salut")
```

Calculez (comme une fonction de n) le nombre de fois où la chaîne de caractères "Salut" est affichée. En déduire la complexité en notation asymptotique.

Exercice 3

On considère l'algorithme ci-dessous où *n* est un entier.

$i \leftarrow 1$
Tant que $i \leq n$
        Afficher ("Salut")
        $i \leftarrow 2i$

Calculez (comme une fonction de n) le nombre de fois que la chaîne de caractères "Salut" est affichée. En déduire la complexité en notation asymptotique.

Exercice 4

On considère l'algorithme ci-dessous où *n* est un entier.

$i \leftarrow 1$
Tant que $i \leq n$
    Pour j de 1 à i
        Afficher ("Salut")
    $i \leftarrow 2i$

Calculez (comme une fonction de n) le nombre de fois que la chaîne de caractères "Salut" est affichée. En déduire la complexité en notation asymptotique.

Partie II

Exercice 5

1) Donnez deux algorithmes (l'un itératif, l'autre récursif) calculant la puissance x^y pour deux nombres entiers x et y.
2) Quelles sont les complexités de ces algorithmes ?
3) Prouvez les algorithmes précédents. Pour cela, identifiez l'invariant de boucle dans le cas itératif (et un équivalent dans le cas récursif) et utilisez un raisonnement par récurrence.
4) Proposez une variante de l'un des deux algorithmes précédents utilisant une stratégie Divide and Conquer ou par exemple on divise y par 2 à chaque étape. Quelle est sa complexité ?

Exercice 6

1) Donnez deux algorithmes (l'un itératif, l'autre récursif) calculant le maximum d'un tableau de nombres.
2) Quelles sont leurs complexités.
3) Prouvez ces algorithmes. Pour cela, identifiez l'invariant de boucle dans le cas itératif (et un équivalent dans le cas récursif) et utilisez un raisonnement par récurrence.

Exercice 7

On considère un tableau **trié** d'entiers $A[1,\ldots,n]$, et un entier x que vous devez trouver dans ce tableau.

1) Ecrivez un algorithme de recherche dichotomique qui fonctionne en divisant par 2 la zone du tableau dans lequel il recherche x à chaque itération.

2) Quelle est sa complexité ?

Exercice 8

On vous donne l'algorithme ci-dessous.

```
Fonction MYSTERE(n ∈ N)
    if n == 0 then
        return 0
    else
        return MYSTERE(n − 1) + 2n − 1
```

1) Que fait cet algorithme ?

Partie III

Exercice 9 *

On rappelle le principe de l'algorithme de tri par sélection. C'est un algorithme itératif.
Sur un tableau de n éléments (numérotés de 0 à n-1), le principe du tri par sélection est le suivant :

- rechercher le plus petit élément du tableau, et l'échanger avec le premier élément (d'indice 0) ;
- rechercher le plus petit élément de la portion du tableau comprise entre les indices 1 et n-1, et l'échanger avec l'élément d'indice 1 ;
- rechercher le plus petit élément de la portion du tableau comprise entre les indices 2 et n-1, et l'échanger avec l'élément d'indice 2 ;
- continuer de cette façon jusqu'à ce que le tableau soit entièrement trié.

1) Illustrer le principe du tri par sélection sur la liste [3,19,7,1,5]
2) Ecrire l'algorithme du tri par insertion.
3) Calculer la complexité de l'algorithme dans le pire et dans le meilleure des cas (sans la prouver). Est ce que celle-ci dépend de l'ordre du tableau initial ?

Exercice 10 *

On rappelle le principe de l'algorithme du tri à bulles. Le tri à bulles ou tri par propagation consiste à comparer répétitivement les éléments consécutifs d'un tableau, et à les permuter lorsqu'ils sont mal triés. Il doit son nom au fait qu'il déplace rapidement les plus grands éléments en fin de tableau, comme des bulles d'air qui remonteraient rapidement à la surface d'un liquide.

1) Ecrire l'algorithme du tri à bulles.
4) Combien d'étapes nécessite cet algorithme ?

Exercice 11 *

L'algorithme de tri fusion consiste, à partir de deux listes triées, à construire une liste triée comportant les éléments issus de ces deux listes (leur fusion). Le principe de l'algorithme de tri fusion repose sur cette observation : le plus petit élément de la liste à construire est soit le plus petit élément de la première liste, soit le plus petit élément de la deuxième liste. Ainsi, on peut construire la liste élément par élément en retirant tantôt le premier élément de la première liste, tantôt le premier élément de la deuxième liste (en fait, le plus petit des deux, à supposer qu'aucune des deux listes ne soit vide, sinon la réponse est immédiate). L'algorithme du tri fusion est le suivant :

1. Si la liste n'a qu'un élément (ou aucun), elle est déjà triée.
2. Sinon, séparer la liste en deux parties à peu près égales.
3. Trier récursivement les deux parties avec l'algorithme de tri fusion.
4. Fusionner les deux listes triées en une seule liste triée.

1) Illustrer le principe du tri fusion sur la liste [3,19,7,1]
2) Écrire en pseudo-code une fonction fusion qui prend en entrée deux listes l1 et l2 supposées triées et renvoie une liste l contenant les mêmes éléments que l1 et l2 mais rangés par ordre croissant.
3) Écrire en pseudo-code la fonction récursive de tri qui, si la liste contient 0 ou 1 élément, la renvoie telle quelle (puisqu'elle est déjà triée) et sinon renvoie le résultat de la fusion de sa partie gauche triée et de sa partie droite triée.
4) En vous aidant de vos réponses aux deux questions précédentes, écrire en pseudo-code l'algorithme de tri fusion.
5) Quelle est la complexité dans le pire des cas de cet algorithme ?

Partie IV

Exercice 12 *

La fonction d'Ackermann s'écrit de la façon suivante pour tout couple d'entiers *m* et *n* positifs:

$$\begin{aligned} A(m,n) &= n+1 \; si \; m = 0 \\ &= A(m-1,1) \; si \; n = 0 \\ &= A(m-1, A(m, n-1)) \; sinon \end{aligned}$$

1) Vérifiez que cette fonction est valide (i.e. est définie $\forall m, n$)
2) Quel est la complexité du calcul de $A(m,n)$?

Exercice 13 **

Le problème des tours de Hanoi consiste à déplacer n disques de diamètres différents empilés sur une tour (tige) de départ vers une tour d'arrivée en utilisant une tour intermédiaire et en suivant les règles suivantes:

- On ne peut déplacer qu'un disque à la fois
- On ne peut placer un disque sur une tour que si le dernier disque sur la tour est de plus grande taille

On suppose que la configuration de départ vérifie ces règles également (c'est-à-dire que les disques sont empilés sur la tour de départ par ordre décroissant des diamètres).

1) Donnez un algorithme permettant de résoudre le problème.
2) Quelle est sa complexité ?