TD 3 - Complexité

Objectif : Savoir dérouler un algorithme et calculer sa complexité.

Exercice 1 - Rappel sur quelques propriétés des logarithmes :

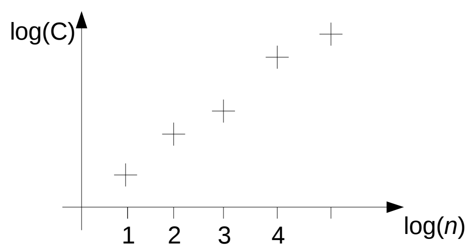
1. $\log_m(x) = \frac{\ln(x)}{\ln(m)}$ pour tout entier $m > 0$
2. Le domaine de ln est $\mathbb{R}_+^*$
3. $\ln(x \times y) = \ln(x) + \ln(y)$ pour tous réels x, y
4. ln est strictement croissante
5. ln est une bijection de $\mathbb{R}_+^*$ dans $\mathbb{R}$
6. $\forall x \in \mathbb{R}_+^*, \quad p^{\log_p(x)} = x$
7. $\forall x \in \mathbb{R}, \quad \log_p(p^x) = x$

Questions :

a- Vaut-il mieux une complexité de 10^n ou n^{10} ? (indication : considérer n grand, de la forme 10^p)

b- Si $\log_{10}(n) = 5$, que vaut n ?

c- On fait tourner un algorithme sur plusieurs jeux de données et on calcule, pour chaque valeur de $\log(n)$, la moyenne empirique du nombre d'opérations significatives. La figure ci-dessous représente le log de la complexité moyenne en fonction de $\log(n)$. La droite autour de laquelle se trouvent les points a un coefficient directeur supérieur à 1. A-t-on une complexité constante, en $\log(n)$, en n, en $n\log(n)$ ou en $n^{p>1}$?



Exercice 2 - Pour chacun des algorithmes suivants,

1. Dérouler l'algorithme sur un exemple et déterminer son rôle
2. Quelle est la mesure de complexité à utiliser pour évaluer la complexité de cet algorithme ?
3. Calculer la complexité dans le meilleur et dans le pire cas

Pour l'algorithme 4, calculer aussi la complexité moyenne. Pour cela, vous pourrez utiliser un paramètre q désignant la probabilité que le vecteur soit trié.

Algorithme 1 :

début
    /* ENTRÉES : Un vecteur V de taille n */
    /* SORTIE : A DETERMINER */
    pour *i de 1 à n* **faire** $T(n-i+1) \leftarrow V(i)$
    retourner T
fin

Algorithme 2 :

```
début
    /* ENTRÉES : Un vecteur d'entiers V de taille n */
    /* SORTIE : A DETERMINER */
    s ← V(1)
    pour i de 2 à n faire
        si s > V(i) alors s ← V(i)
    retourner s
fin
```

Algorithme 3 :

```
début
    /* ENTRÉES : Un vecteur d'entiers V de taille n */
    /* SORTIE : A DETERMINER */
    T(1) ← V(1) ; T(2) ← V(1) ; T(3) ← V(1)
    pour i de 2 à n faire
        T(3) ← T(3) + V(i)
        si T(2) < V(i) alors T(2) ← V(i)
        si T(1) > V(i) alors T(1) ← V(i)
    T(3) ← T(3)/n
    retourner T
fin
```

Algorithme 4 :

```
début
    /* ENTRÉES : Un vecteur d'entiers V de taille n */
    /* SORTIE : A DETERMINER */
    rep ← Vrai ; i ← 1
    tant que i < n faire
        si V(i) > V(i+1) alors rep ← Faux ; i ← n
        sinon i ← i + 1
    retourner rep
fin
```

Algorithme 5 :

```
début
    /* ENTRÉES : Deux vecteur d'entiers U et V de taille n. */
    /* Tous les éléments d'un vecteur sont distincts*/
    /* SORTIE : A DETERMINER */
    s ← 0
    pour i de 1 à n faire
        j ← 1
        tant que U(i) ≠ V(j) et j < n faire
            j ← j + 1
        si U(i) = V(j) alors
            s ← s + 1
    retourner s
fin
```