TD1: Rappels de complexités, probabilités

Exercice 1. *Complexité*

✎ Calculer la complexité de chacun des algorithmes suivants. L'instruction `<op_elem>` désigne n'importe quelle opération élémentaire, qui a complexité $O(1)$ par définition. Si besoin, on pourra utiliser le '*master theorem*'.

```
ALGO1(n) :
Pour i = 0 à n-1 :
    Pour j = 0 à n-1 :
        Pour k = 0 à j :
            <op_elem>
Pour i = 0 à n-1 :
    <op_elem>
```

```
ALGO2(n) :
Si n = 0 : Renvoyer val
ALGO2(n-1)
<op_elem>
ALGO2(n-1)
<op_elem>
```

```
ALGO3(n) :
<op_elem>
Tant que n > 1 :
    n ← n/3
    <op_elem>
<op_elem>
```

```
ALGO4(n) :
Si n ≤ 1 : Renvoyer 17
Pour t = 0 à n-1 :
    ALGO4(n-1)
```

```
ALGO5(n) :
Tant que n ≥ 0 :
    <op_elem>
    n ← n-3
<op_elem>
```

```
ALGO6(n) :
Si n ≤ 1 : Renvoyer 4
ALGO6(⌊n/2⌋)
<op_elem>
```

```
ALGO7(n) :
Si n ≤ 1 : Renvoyer 51
ALGO7(⌈n/2⌉)
Pour i = 0 à n : <op_elem>
ALGO7(⌈n/2⌉)
<op_elem>
```

```
ALGO8(n) :
Si n ≤ 1 : Renvoyer 0
<op_elem>
ALGO8(⌈n/3⌉)
ALGO8(⌈n/3⌉)
Pour i = 0 à n : <op_elem>
```

```
ALGO9(n) :
Si n ≤ 1 : Renvoyer 911
ALGO9(⌈n/2⌉)
ALGO9(⌈n/2⌉)
ALGO9(⌈n/2⌉)
Tant que n > 2 : n ← n-15
```

Exercice 2. *Probabilités discrètes*

1. On tire deux dés à six faces, équilibrés. Calculer la probabilité des évènements suivants :
   a. Les deux dés ont la même valeur.
   b. Le deuxième dé vaut strictement plus de points que le premier.
   c. La somme des deux dés est un nombre pair.
2. On tire dix fois une pièce équilibrée. Calculer la probabilité des évènements suivants.
   a. On n'obtient que des PILE.
   b. On obtient au moins une fois PILE.
   c. On obtient autant de PILE que de FACE.
   d. *(Bonus)* On obtient plus de PILE que de FACE.

Exercice 3. *Probabilités discrètes, suite...*

1. On jette un dé équilibré à k faces, numérotées de 1 à k. Soit X la variable aléatoire décrivant la valeur observée. Calculer $\mathrm{E}[X]$.
2. On tire dix fois de suite une pièce équilibrée. On note X la variable aléatoire décrivant le nombre de PILE obtenus. Calculer $\mathrm{E}[X]$.
3. Un singe tape sur un clavier à 26 lettres (toutes en minuscules) de manière aléatoire : chaque lettre tapée est choisie uniformément et indépendamment parmi les 26 lettres possibles. Le singe tape un texte d'un million de lettres.
   1. Quelle est l'espérance du nombre de fois que la lettre '`x`' apparaît dans le texte ?
   2. Quelle est l'espérance du nombre de fois que le mot '`algo`' apparaît dans le texte.

Exercice 4. *Un algorithme probabiliste*

Dans cet exercice, T est un tableau de n entiers.

1. Soit x un entier qui apparaît (une seule fois) dans T. On veut connaître l'indice i tel que $T_{[i]} = x$. Donner un algorithme naïf déterministe pour ce problème et analyser sa complexité.

2. On décrit maintenant un algorithme probabiliste simple : on tire un indice i aléatoirement entre 0 et $n-1$ et on teste si $T_{[i]} = x$; on renvoie i si c'est le cas et on recommence sinon.

1. Quelle est la probabilité p de trouver le bon indice au premier essai ?
2. On note X la variable aléatoire correspondant au nombre d'essais nécessaires avant de trouver x. On note E_n l'espérance de X. En utilisant la formule de l'espérance totale, conditionnée au fait de trouver x au premier tirage ou non, montrer que E_n vérifie $E_n = 1 + (1-p)E_n$.
3. En déduire que $E_n = n$. Cet algorithme est-il intéressant ?

3. On dit qu'un élément x est *majoritaire* dans T si au moins la moitié des éléments de T sont égaux à x : formellement, $\#\{i : T_{[i]} = x\} \geq n/2$.

1. On considère le problème suivant : en supposant que T contient un élément majoritaire, on veut renvoyer cet élément.
   - i. Donner un algorithme de complexité quadratique pour le problème.
   - ii. On tire i aléatoirement entre 0 et $n-1$. Quelle est la probabilité que $T_{[i]}$ soit l'élément majoritaire ?
   - iii. En déduire un algorithme probabiliste qui trouve x, et dont l'espérance du temps de calcul est linéaire en n. *Pour le calcul de l'espérance, on peut réutiliser la technique de la question **1.iii**.*
2. On considère maintenant le problème suivant : étant donné T, on cherche à déterminer s'il existe un élément majoritaire. *On ne suppose donc plus que T possède un élément majoritaire.*
   - i. Donner un algorithme de complexité quadratique pour le problème.

   On applique l'algorithme suivant : on tire k indices $i_1, \ldots, i_k$ aléatoirement et indépendamment entre 0 et $n-1$; on renvoie VRAI si $T_{[i_j]}$ est majoritaire pour (au moins) l'un des indices i_j et FAUX sinon.

   - ii. Détailler l'algorithme et calculer sa complexité.
   - iii. S'il n'existe pas d'élément majoritaire dans T, montrer que l'algorithme renvoie FAUX quelque soit k.
   - iv. S'il existe un élément majoritaire dans T, quelle est la probabilité que l'algorithme renvoie FAUX ?
   - v. Quelle est la complexité de l'algorithme si on veut que le résultat soit correct avec probabilité $\geq 999/1000$? Et avec probabilité $\geq 1 - 1/n$?

Exercice 5. *Simulations*

1. Soit BITALÉATOIRE() un générateur aléatoire qui renvoie 0 avec probabilité $\frac{1}{2}$ et 1 avec probabilité $\frac{1}{2}$.

1. Écrire un algorithme ENTIERALÉATOIRE(k) qui renvoie un entier aléatoire entre 0 et 2^k-1. Montrer que la distribution obtenue est uniforme : chaque entier entre 0 et $2^k - 1$ est obtenu avec probabilité $1/2^k$.
2. On souhaite maintenant tirer un entier aléatoire entre 0 et $N-1$, où N est un entier de k bits. On propose la solution suivante : on tire un entier n avec ENTIERALÉATOIRE(k) puis on renvoie $n \bmod N$. La distribution obtenue est-elle toujours uniforme ? *On pourra calculer la probabilité d'obtenir 0 et celle d'obtenir $N-1$.*
3. On veut toujours tirer un entier aléatoire entre 0 et $N-1$. On opte pour une nouvelle stratégie : on tire toujours n avec ENTIERALÉATOIRE(k) ; si $n < N$ on renvoie n ; sinon on recommence. Montrer que cette stratégie produit bien une distribution uniforme et borner l'espérance du nombre d'appels à ENTIERALÉATOIRE.
4. *(bonus)* Supposons qu'on dispose d'une fonction ENTIERALÉATOIRE() qui renvoie un entier entre 0 et $2^K - 1$ pour une certaine valeur K qu'on ne choisit pas. Proposer une méthode pour tirer un entier aléatoire entre 0 et $N-1$, pour n'importe quel $N \leq 2^K$, qui produit bien une distribution uniforme et dont l'espérance du nombre d'appels à ENTIERALÉATOIRE est au plus 2.

2. On suppose maintenant disposer d'un générateur RÉELALEATOIRE() qui produit un réel aléatoire x entre 0 et 1. Soit $V = \{v_1, \ldots, v_k\}$ un ensemble de valeurs entières, et $P = \{p_1, \ldots, p_k\}$ des probabilités, telles que $\sum_i p_i = 1$. On souhaite tirer v_i avec probabilité p_i. Pour cela, on partitionne l'intervalle $[0,1]$ en k intervalles de longueurs $p_1, \ldots, p_k$ respectivement. On tire un réel r entre 0 et 1, et on renvoie v_i si r appartient à l'intervalle correspondant à v_i.

1. Justifier que l'algorithme esquissé ci-dessus renvoie bien un entier avec la distribution souhaitée.
2. Écrire formellement l'algorithme.