CAPES MATHS OPTION INFORMATIQUE

ALGORITHMIQUES CLASSIQUES

Recherche dans un tableau

a. Ecrire en Python la fonction qui recherche un élément e dans un tableau tab. La fonction retournera l'indice de l'élément s'il est présent, -1 sinon.

b. Ecrire en Python la fonction récurvive qui recherche un élément dans un tableau trié de façon dichotomique.

c. Quel est le coût d'une telle recherche ?

d. Donner une version de l'algorithme de recherche dichotomique qui est plus performante en termes d'espace mémoire.

Recherche d'un mot dans une chaîne

a. Ecrire une fonction en Python qui prend en paramètres un mot et une chaîne de caractères et détermine si le mot est dans la chaîne (sans utiliser la fonction in déjà existante en Python).

b. Estimer la complexité de cet algorithme dans le pire des cas.

Insertion dans liste triée

a. Ecrire en Python la fonction qui permet d'ajouter un élément dans une liste triée. Cette fonction retournera la nouvelle liste. Vous ne pouvez pas utiliser la procédure insert de Python.

b. Donner le coût de cette opération d'insertion.

c. Si l'élément était d'abord recherché par dichotomie aurait-on une meilleure complexité ?

Conversion décimal-binaire

L'objectif est de créer un programme qui code un nombre entier naturel, donné en écriture décimale, en binaire. Les nombres binaires seront représentés par des chaînes de caractères. Pour plus de lisibilité, on insérera un espace tous les quatres caractères en partant du dernier. Ainsi, le nombre décimal 525 sera traduit par la chaîne "10 0000 1101" en binaire.

a. Ecrire une fonction conversion en Python, qui prend en paramètre un nombre entier naturel en décimal et renvoie la chaîne de caractères composée uniquement de 0 et de 1 correspondant à son écriture binaire. On ne cherchera pas encore à insérer d'espaces.

b. Ecrire une fonction prenant en paramètre une chaîne de caractères et réalisant l'insertion d'un espace tous les quatres caractères en partant du dernier.

c. Modifier la fonction conversion pour prendre en compte l'insertion des espaces

Exponentielle rapide

On s'intéresse à calculer une exponentielle de la manière la plus efficace possible étant donné un réel n et un exposant entier positif a : n^a.

a. Ecrire une première version de la fonction exponentielle, la plus simple et intuitive possible.

b. Indiquer la complexité de cette fonction.

Dans une deuxième version, on propose d'utiliser le principe d'exponentiation rapide reposant sur le constat que :

$$n^a = \begin{cases} (n^2)^{\frac{a}{2}} & \text{si a est pair} \\ n \times n^{a-1} & \text{si a est impair} \end{cases}$$

c. Ecrire en Python la fonction `exponentielleRapide` qui retourne la valeur de n^a suivant le principe ci-dessus.

d. Donner la complexité de cette fonction.

Crible d'Eratosthène

On se propose de calculer tous les nombres premiers plus petits qu'un entier n donné. La méthode consiste à calculer pas à pas ces nombres en utilisant la règle suivante : si un entier k n'est divisible par aucun nombre premier plus petit que k alors il est lui-même premier. Ecrire en Python la fonction `eratosthene` qui retourne le tableau des nombres premiers plus petits que n passé en paramètre.

Analyse de la complexité d'un algorithme

On considère le pseudo-code suivant, comportant deux « tant que » imbriqués. On cherche à mesurer la complexité de cette imbrication en fonction de n. Pour cela, on utilise la variable « compteur », qui est incrémentée à chaque passage dans le « tant que » interne.

```
Variables :
  n : entier
  compteur : entier
  i, j : entiers
Début
  Afficher(« Quelle est la valeur de n ? »)
  Saisir(n)
  compteur ← 0
  i ← 1
  Tant que (i < n) Faire
      j ← i + 1
      Tant que (j ≤ n) Faire
          compteur ← compteur + 1
          j ← j + 1
      Fin tantque
      i ← i * 2
  Fin tantque
  Afficher(compteur)
Fin
```

a. Quelle est la valeur finale du compteur dans le cas où n = 16 ?

b. Considérons le cas particulier où n est une puissance de 2 : on suppose que $n = 2^p$ avec p connu. Quelle est la valeur finale du compteur en fonction de p ? Justifiez votre réponse.

c. Réexprimez le résultat précédent en fonction de n.

Calcul d'un polynôme en un point

Soit un polynôme *P* tel que $P(x) = \sum_{k=0}^{n} a_k x^k$.

On veut écrire la fonction qui retourne la valeur de $P(x)$ pour une valeur de *x* passée en paramètre, les coefficients du polynôme étant également passés en paramètre dans un tableau. Pour cela, une méthode efficace est la méthode de Horner, qui utilise la réécriture suivante de *P*(*x*) :

$$P(x) = \Big(\big(\ldots\big((a_n x + a_{n-1})x + a_{n-2}\big)x + \cdots\big)x + a_1\Big)x + a_0$$

La méthode consiste donc à multiplier le coefficient de plus haut degré par *x* et à lui ajouter le coefficient suivant. On multiplie alors le nombre obtenu par *x* et on lui ajoute le troisième coefficient, etc., jusqu'à avoir ajouté le coefficient constant.

Exemple : calcul de $4x^3 - 7x^2 + 3x - 5 = \big((4x - 7)x + 3\big)x - 5$
pour *x* = 2 :

- première étape : 4*2 – 7 = 1
- deuxième étape : 1*2 + 3 = 5
- troisième étape : 5*2 – 5 = 5

Voici le code Python de la fonction qui calcule $P(x)$ selon cette méthode :

```
def valeur_polynome (x, coefs, n) :
# Précondition : coefs contient les (n+1) coefficients du polynôme, ainsi coef[0]
#                contient a0, coef[1] contient a1, etc.
# Résultat : retourne P(x), la valeur du polynôme au point x
1   y = 0.0
2   k = n
3   while k >= 0 :
4       y = y * x + coefs[k]
5       k = k - 1
6   return y
```

a. Combien de fois passe-t-on par les lignes 4 et 5 si le polynôme est de degré *n* ?

b. Faire un tableau comptant le nombre d'opérations faites à chaque ligne (affectation, addition/soustraction, multiplication, comparaison).

c. Déduisez-en le temps d'exécution *T*(*n*) de la fonction en microsecondes, en supposant qu'une affectation prend t_{aff} microsecondes, une addition ou une soustraction t_{add} microsecondes, une multiplication t_{mult} microsecondes, et une comparaison t_{comp} microsecondes.

d. De quelle fonction mathématique de n ce temps d'exécution *T*(*n*) est-il « grand *O* » ?

e. Complétez l'invariant de boucle de l'algorithme de Horner : « Au début de chaque itération de la boucle while (juste avant d'exécuter la ligne 4), on sait que (complétez les pointillés) : $y = \sum_{i=0}^{n-(k+1)} a_{\ldots\ldots} x^i$ ».

f. A l'aide la propriété de terminaison de cet invariant, montrez que l'algorithme de Horner permet bien d'obtenir la valeur du polynôme au point *x*.

g. Ecrivez en langage Python la version « naïve » de la fonction valeur_polynome, qui calcule *P*(*x*) selon la formule $P(x) = \sum_{k=0}^{n} a_k x^k$, c'est-à-dire en calculant pour chaque terme la puissance de *x* via autant de multiplications que nécessaire (on n'utilisera donc pas la fonction `pow()` ou l'opérateur `**`).

h. Combien de multiplications fait-on dans cette version naïve, si le polynôme est de degré *n* ?

Points fixes

Soit $n \in \mathbb{N}^*$. On s'intéresse aux points fixes de fonctions $f : E_n \to E_n$, où $E_n = [0; n-1]$. On représente une fonction par une liste L de taille n telle que pour tout x, $L[x] = f(x)$. Pour tout k entier naturel, on note f^k l'itérée k de f, c'est-à-dire $f^k = f \circ f \circ \ldots \circ f$.

a. Ecrire une fonction admetPointFixe qui prend en argument une liste L de taille n et renvoie vrai si la fonction f représentée par L admet un point fixe, et faux sinon.

b. Ecrire une fonction nbPointFixe qui prend en argument une liste L de taille n et renvoie le nombre de points fixes de la fonction f représentée par L.

c. Ecrire une fonction itere qui prend en premier argument une liste L de taille n représentant une fonction f, en deuxième et troisième arguments un entier x et un entier k, et renvoie $f^k(x)$.

d. Ecrire une fonction nbPointFixeItere qui prend en premier argument un liste L de taille n représentant une fonction f, en deuxième argument un entier naturel k et renvoie le nombre de points fixes de f^k.

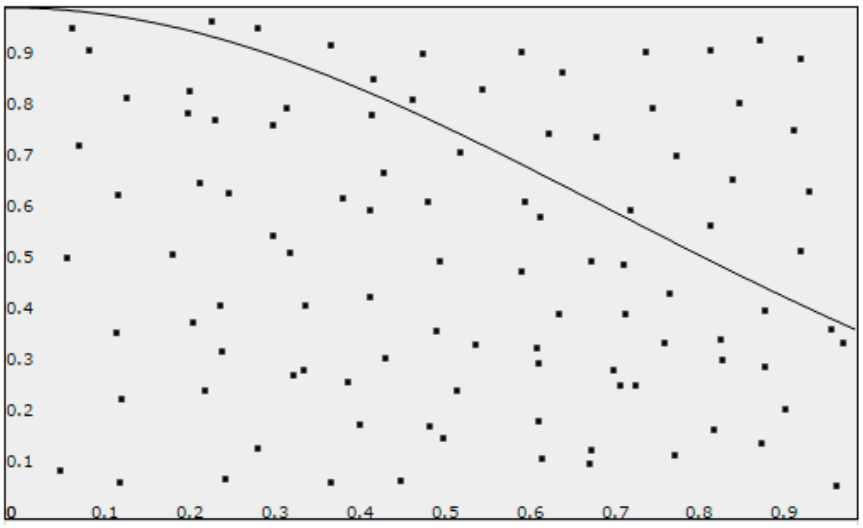
Calcul d'intégrale par la méthode de Monte-Carlo

On souhaite calculer une valeur approchée de $I = \int_0^1 e^{-x^2}\, dx$ c'est-à-dire de l'aire sous la courbe de la fonction positive f définie par $f(x) = e^{-x^2}$ entre $x = 0$ et $x = 1$.
Cette fonction f est continue, positive et strictement décroissante sur [0 ;1]. La valeur maximale prise par f sur [0 ;1] est $f(0) = 1$. Soit $(0; \vec{\imath}, \vec{\jmath})$ un repère orthonormal. On considère les points : $A(0,1), B(1,1), C(1,0), D(0,0)$. On note D le domaine défini par la courbe représentative de f, la droite d'équation $x = 0$, celle d'équation $x = 1$ et l'axe des abscisses.
Soit l'expérience aléatoire suivante : on tire au hasard un point de coordonnées (x,y) dans le rectangle ABCD. On associe à cette expérience aléatoire la variable aléatoire X qui prend la valeur 1 en cas de succès (appartenance du point au domaine D) et 0 en cas d'échec. On est donc en présence d'une épreuve de Bernoulli. On note p la probabilité de succès de cette expérience aléatoire.

a. Exprimer p en fonction de I et de l'aire du rectangle ABCD.

On cherche à estimer au mieux p à partir de la répétition de l'expérience aléatoire. On répète donc n fois cette expérience et on note N_n la variable aléatoire qui prend pour valeur le nombre de succès obtenus lors de cette répétition de n expériences indépendantes et identiques.

Ci-dessous est représenté le tirage aléatoire, de manière indépendante, de 100 points dans le rectangle ABCD.



b. Par lecture graphique, déterminer la valeur prise par la variable aléatoire N_n à l'issue de ces 100 tirages. En déduire une estimation de p et de I, et comparer ces résultats avec la valeur exacte de I, en admettant que $I \approx 0.746824132812$.

On généralise la méthode précédente, appelée méthode de Monte-Carlo, à toute fonction f continue, positive et décroissante sur un intervalle [a;b]. On note D le domaine défini par la courbe représentative de f, la droite d'équation $x = a$, celle d'équation $x = b$, et l'axe des abscisses.

c. Quelles sont les coordonnées des points A, B, C et D dans ce cas ? Quelle est l'aire du rectangle ABCD ?

d. Comment caractériser l'appartenance d'un point de coordonnées (x,y) au domaine D ?

e. On rappelle que l'on dispose de la fonction random du module random pour générer un nombre aléatoire pris entre 0 et 1. Quelle instruction permet d'obtenir un nombre aléatoire pris entre a et b ?

f. Ecrire la fonction monteCarlo qui prend en paramètres la fonction continue, décroissante et positive f, les nombres réels a et b, et l'entier n non nul, et qui renvoie une valeur approchée de l'intégrale I par la méthode de Monte-Carlo.

g. Donner des valeurs approchées de I obtenues pour $n = 100, n = 1000, n = 10000, n = 100000, n = 1000000$.

Contrairement à d'autres méthodes d'intégration numérique, cette méthode est stochastique (aléatoire). Elle retourne un résultat différent à chaque appel de la fonction monteCarlo.

h. Comment modifier l'algorithme si f est continue, positive et croissante ?