
Question 1 (2pts): binaire, octal et hexadécimal (sans limitation de capacité)

Soit X et Y deux nombres entiers positifs (non-signés) dont la valeur est donnée en **octal** :
$X = 2351_8$ et $Y = 4017_8$. Quelle est la valeur du résultat de l'addition X + Y en **hexadécimal** ?

A	$CF8_{16}$
B	6370_{16}
C	6368_{16}
D	$19F0_{16}$

Question 2 (2 pts) : exécution d'un algorithme

Algo1
entrée : entier **N** > 0 sortie : aucune (affiche des valeurs)

i ← N
r ← 0
Tant que i > 0
  Pour j **de** 1 **à** i
  r ← r + 2
  i ← i / 2 // division entière
 	
Afficher r

Quelle valeur est affichée pour N valant 4 :

A	0
B	8
C	14
D	Aucune des autres valeurs

Question 3 (2 pts) : ordre de complexité

Estimer le nombre d'instructions exécutées par l'algorithme Algo1 en fonction de N lorsque N tend vers l'infini.

Simplifier l'expression obtenue en sachant que $\sum_{k=0}^{\infty} \frac{1}{2^k} = 2$

En déduire l'ordre de complexité de Algo1 :

A	O(Nlog(N)) mais pas O(N)
B	$O(N^2)$ mais pas O(Nlog(N))
C	O(log(N)) mais pas O(1)
D	O(N) mais pas O(log(N))

Question 4++ (3pts): algorithme avec 3 variables en virgule flottante IEEE 754 simple précision ayant une précision $\varepsilon = 2^{-23}$

Algo2
entrée : 3 nombres en **virgule flottante IEEE 754 simple précision** : **start, stop, step** tels que : stop > start > 0 et 0 < step < (stop – start) sortie : *aucune,* réalise des actions d'affichage

x ← start
Tant que x < stop
  Afficher x
  Afficher passage à la ligne
  x ← x + step

Indiquer la réponse correcte pour l'appel **Algo2(2.^{23}, 2.^{25}, 1.)** :

A	Cet appel affiche x jusqu'au moment où x est égal à stop ce qui fait quitter la boucle et terminer l'algorithme
B	Cet appel affiche x jusqu'au moment où x > stop
C	Cet appel n'entre jamais dans la boucle et termine l'algorithme sans affichage
D	L'algorithme ne se termine pas car il entre dans une boucle infinie pour cet appel

⇨ **Complément exigé pour valider votre réponse ci-dessus : Justifier votre réponse**

Lorsque x atteint la valeur 2^{24} la somme x+1 donne x car l'erreur absolue vaut 2 à partir de 2^{24}

Comme la somme x+1 reste constante on crée ainsi une boucle infinie et le programme ne se termine pas.

Question 5 (1 pt): représentation des données

Un système de représentation des couleurs choisit de représenter une couleur en indiquant une quantité pour chacune des trois couleurs suivantes : Rouge, Vert et Bleu.

Sachant que la quantité de chacune des composantes (rouge, verte et bleue) est représentée avec deux octets, combien de couleurs distinctes est-il possible de définir avec cette représentation ?

A	6^{16}
B	256^6
C	48^2
D	Aucune des autres réponses

Question 6 (1 pt): Quelle est la complexité de **Algo3** en fonction de n ?

Algo3
entrée : entier **n** strictement positif sortie : *non précisée*
Si n=1   ***Sortir*** 1 ***Sortir*** n*__Algo3__(n-1)

A	O(n!) mais pas O(2^n)
B	O(log(n)) mais pas O(1)
C	O(n) mais pas O(log(n))
D	O(n^2) mais pas O(n)

Question 7 (1pt) : Représentation des entiers en **complément à 2** sur une capacité de **4 bits**

A	Le domaine couvert est [-7 , 8]
B	Le domaine couvert est [-8 , 7]
C	Le domaine couvert est [-8 , 8]
D	Le domaine couvert est [-7 , 7]

Question 8 (2 pts):
Dans la représentation de la question précédente, le résultat de l'addition : $1011_2 + 1111_2$

A	produit une violation du domaine couvert
B	donne -6
C	donne 6
D	donne une autre valeur dans le domaine couvert (différente de 6 et de -6)

Question 9++ (2 pts) : Quelle affirmation est VRAIE

A	Le problème de l'arrêt appartient à la classe NP
B	Le problème du sac à dos est dans P grâce au tri des objets selon le critère V/P
C	Le problème du tri appartient à la classe NP
D	Il n'existe pas de surjection de l'ensemble des entiers vers l'ensemble des rationnels

⇨ **Complément exigé pour valider votre réponse ci-dessus : Justifier votre réponse**

Le problème du tri appartient à P car sa complexité est O(Nlog(N)).
Or P est inclus dans NP, donc le problème du tri appartient à la classe NP

Question Ouverte 1 : (5 pts) Soit une liste **L** contenant N entiers avec N>0

1.1) Ecrire le pseudocode de l'algorithme **Tous_Differents** qui renvoie **VRAI** si ***tous les éléments de L sont différents entre eux***.

Par exemple l'algorithme renvoie VRAI pour L={4, 9, 12} tandis qu'il renvoie FAUX pour L={4, 9, 12, 11, 9, 22}. L'algorithme renvoie VRAi s'il n'y a qu'un seul élément dans L. Formulé d'une autre manière, on peut dire que cet algorithme doit renvoyer FAUX dès qu'il trouve deux éléments égaux.

On demande une version itérative, sans récursivité, pour cette question

Tous_Differents *(faisable en 5 lignes)*
entrée : entier **N** > 0 , Liste **L** contenant N entiers **sortie :** booléen VRAI si tous les éléments sont différents entre eux (sinon renvoie FAUX)

Pour i de 1 à N-1 par pas de 1
  Pour j de (i+1) à N
  Si L(i) = L(j)
  Sortir Faux
 	
Sortir Vrai

1.2) Quelle est la complexité de votre algorithme en fonction de N ? **$O(N^2)$**

Justifier votre réponse :

Le nombre de passages dans la boucle interne est : (N-1) + (N-2) + … + 2 + 1
En regroupant les termes au début et à la fin, on obtient (N-1)/2 paires de valeur N

Nb passages = N(N-1)/2 dont le terme dominant est en N^2

1.3) fournir une ***version récursive de votre algorithme*** (qu'on appellera **TDR**). Faire comme dans le cours sur le tri fusion pour la syntaxe du passage d'une sous-liste lors d'un appel récursif: après la taille de la sous-liste, indiquer le nom de la liste **L** avec, entre parenthèses, les indices du premier et du dernier élément transmis pour la sous-liste.

Exemple : voici un appel récursif qui transmet la sous-liste contenant les deux derniers éléments de L : **TDR(2, L(N-1, N))**

TDR *(faisable en 8 lignes)*
entrée : entier **N** > 0 , Liste **L** contenant N entiers **sortie :** booléen VRAI si tous les éléments sont différents entre eux (sinon renvoie FAUX)

Si N = 1
  Sortir Vrai
 	
Si N = 2
  Sortir L(1) ≠ L(2)
 	
Pour j de 2 à N
  Si L(1) = L(j)
  Sortir Faux
 	
Sortir TDR(N-1, L(2, N))

1.4) Quelle est la complexité de votre algorithme en fonction de N ? $O(N^2)$

Justifier votre réponse :

la boucle visible ici traite seulement le premier élément de la liste en le comparant avec tous les autres. L'appel récursif traite la sous-liste de taille N-1 qui commence à partir du second élément.

On retrouve les même nombres de passages dans la boucle sur j : (N-1) au premier appel récursif puis (N-2) au second, etc… le bilan est du même ordre que le cas itératif avec un terme dominant en N^2.

Question Ouverte 2 : (4 pts)

Soit l'algorithme mystère **Algo4**

Algo4
entrée : entier **N** > 0, Liste **L** contenant N valeurs entières 0 ou 1, et entier **P** tel que 0<P≤N **sortie :** *à déterminer*
i ← 1 k ← P – 1 ***Sortir*** **Algo5**(N, L, i, k)

Avec **Algo5** :

Algo5
entrée : **N** > 0 , Liste **L** contenant N valeurs entières 0 ou 1, entier **i** >0, entier **k** **sortie :** *à déterminer*
Si i = N ***Sortir*** L(i)* 2^k ***Sortir*** L(i)* 2^k + **Algo5**(N, L, i+1, k-1)

2.1) Indiquer la valeur de la **sortie de Algo4** lorsque que cet algorithme est appelé avec N valant 4, L = {1, 0, 1, 1} et P valant 3. Préciser chacun des appels (récursifs) de **Algo5**, en particulier la valeur renvoyée à chaque appel.

Remarque importante pour cet exercice: l'accès aux éléments de L se fait de gauche à droite, c'est-à-dire que L(1) vaut 1, L(2) vaut 0, L(3) vaut 1 et L(4) vaut 1 .

Algo4 appelle **Algo5**(4, L, 1, 2)
Algo5(4, L, 1, 2) sort L(1)*2^2 + **Algo5**(4, L, 2, 1)
c'est-à-dire 1*4 + **Algo5**(4, L, 2, 1)

Algo5(4, L, 2, 1) sort L(2)*2^1 + **Algo5**(4, L, 3, 0)
C'est-à-dire 0*2 + **Algo5**(4, L, 3, 0)

Algo5(4, L, 3, 0) sort L(3)*2^0 + **Algo5**(4, L, 4, -1)
C'est-à-dire 1*1 + **Algo5**(4, L, 4, -1)

Algo5(4, L, 4, -1) sort L(4)*2^{-1} c'est-à-dire 1*0.5

La valeur renvoyée est : 4 + 1 + 0.5 = 5.5

2.2) Quel type de calcul est réalisé par ces deux algorithmes, c'est-à-dire : quelle est la nature du résultat en fonctions des paramètres ?

Évaluation d'un nombre en virgule fixe avec (N-P) bits de partie fractionnaire pour le motif binaire présent dans L avec les poids forts au début de la liste.

2.3) En déduire la valeur renvoyée par **Algo4** pour les mêmes paramètres sauf P valant 2 puis 1.

- ⇨ Pour P valant 2 : on évalue le motif binaire 10.11 qui est la moitié de la réponse 1) = 2.75
- ⇨ Pour P valant 1 : on évalue le motif binaire 1.011 qui est la moitié du précédent = 1.375

2.4) proposer un algorithme unique itératif sortant le même résultat que **Algo4** sans appeler cet algorithme.

Algo6 *(faisable en 6 lignes)*
entrée : entier **N** > 0, Liste **L** contenant N valeurs entières 0 ou 1, et entier **P** tel que 0<P≤N **sortie :** *la même que Algo4*
total ← 0 Pour i de 1 à N   total ← total + L(i)* 2^(P-i) ***Sortir*** total