

Solution QUIZZ et questions ouvertes MT-EL 27/10/2017

Remarque : l'ordre des réponses était différent selon les variantes. Donc ne faites pas attention à la lettre correspondant à la réponse correcte mais seulement à la **réponse correcte elle-même qui est surlignée en jaune**. **Quelques explications brèves sont ajoutées pour les ordres de complexité**.

Notation: on utilise dans ce quizz la virgule pour séparer les puissances positives des puissances négatives de la base dans la notation positionnelle des nombres.

Question 1: Quelle est la représentation binaire en complément à deux sur 7 bits de -10_{10}

- A 0001010
- B 1001010
- C **1110110**
- D 1000101

Question 2: Sachant qu'une image RGB est codée avec 256 niveaux d'intensité par couleur et a 3 canaux de couleur par pixel, combien faut-il d'octets pour encoder une image de taille 100x100 ?

- A 7 680 000
- B 240 000
- C **30 000** // un octet par canal de couleur x 3 canaux x 10'000 pixels
- D 5 760 000

Question 3: Quelle est le meilleur ordre de complexité pour un algorithme de recherche de **N éléments quelconques** dans une liste **non-triée** de taille **N**.

- A **$O(N^2)$ mais pas $O(N \log(N))$** // (sans tri) N fois une recherche séquentielle sur N éléments
- B **$O(N \log(N))$ mais pas $O(N)$** // avec tri en $N \log N$ puis N recherches dichotomiques en $\log N$
- C $O(N)$ mais pas $O(\log(N))$
- D $O(\log(N))$ mais pas $O(1)$

Question 5: Supposons connu un algorithme de complexité en $O(N \log(N))$ qui permet de trouver la solution d'un problème de décision appelé « TRUC » qui travaille sur des données de taille N. Que peut-on en déduire ?

- A « TRUC » n'est pas dans NP
- B « TRUC » n'est pas dans P
- C **« TRUC » est dans NP** // P est inclus dans NP
- D rien du tout

Question 6: Pour deux entiers strictement positifs a et b, l'algorithme de la multiplication égyptienne de a par b revient à :

- A Décomposer les deux opérandes en produit de facteurs premiers pour recombinaison des puissances des facteurs premiers et effectuer les produits de ces termes
- B **Effectuer une décomposition d'un opérande en une somme de puissances de deux pour réduire le nombre d'additions à calculer sur des multiples de deux de l'autre opérande** // cf séries
- C Effectuer une recherche dichotomique du plus petit opérande parmi les diviseurs du plus grand opérande
- D D'abord effectuer une recherche du plus grand diviseur commun avant de l'élever au carré et de poursuivre le produit avec les autres facteurs plus simples

Question 7: L'algorithme suivant doit calculer le PGCD de deux entiers naturel non nuls x et y fournis en entrée. Choisir la combinaison correcte d'instructions manquantes pour INSTRUCTION1 et INSTRUCTION2:

PGCD
Entrée : deux entiers strictement positifs x et y Sortie : leur PGCD
Tant que $x \neq y$ Si $x > y$ INSTRUCTION1 Sinon INSTRUCTION2 sortir: x

Le caractère / désigne la division entière.

- A INSTRUCTION1= $y \leftarrow y - x$ et INSTRUCTION2= $x \leftarrow x - y$
- B INSTRUCTION1= $x \leftarrow x / y$ et INSTRUCTION2= $y \leftarrow y / x$
- C **INSTRUCTION1= $x \leftarrow x - y$ et INSTRUCTION2= $y \leftarrow y - x$ // vu en série sem2**
- D INSTRUCTION1= $y \leftarrow x - y$ et INSTRUCTION2= $x \leftarrow y - x$

Question 8: soit l'algorithme suivant qui doit chercher la valeur maximum contenue dans une matrice A de N lignes et M colonnes (dans le cas général $M \neq N$). L'accès à un élément d'une matrice A se fait comme en math à l'aide de deux indices qui commencent à 1 et varient jusqu'au nombre de ligne/ de colonne. On indique **d'abord l'indice de ligne suivi par l'indice de colonne** : $A(1,2)$ désigne l'élément de la première ligne et qui est dans la deuxième colonne de cette ligne.

MatMax
entrées : deux entiers strictement positif N et M une matrice A de N lignes et M colonnes contenant des entiers signés sortie : la valeur entière maximum contenue dans la matrice A
$\max \leftarrow A(1,1)$ Pour i de 1 à N Pour j de 1 à M Si $A(i, j) > \max$ $\max \leftarrow A(i,j)$ sortir: \max

Quel est l'ordre de complexité de cet algorithme avec la notation de Landau ?

- A $O(N^M)$
- B $O(N^2)$
- C **$O(NM)$ // exemple en série sem4**
- D $O(N+M)$

Question 9: Si on suppose maintenant que les valeurs de la matrice A fournie à l'algorithme **MatMax** sont triées dans l'ordre croissant sur chaque ligne mais qu'il n'y a pas d'ordre particulier entre les lignes. Quelle est la proposition correcte ?

- A L'algorithme MatMax ci-dessus s'exécute avec une complexité de $O(N)$
- B L'algorithme MatMax ci-dessus s'exécute avec une complexité de $O(M)$
- C **L'algorithme MatMax peut être modifié pour obtenir une complexité qui ne dépend plus de M**
- D L'algorithme MatMax peut être modifié pour obtenir une complexité qui ne dépend plus de N

Question 10: Soit une représentation en virgule flottante en binaire utilisant 5 bits pour la mantisse et 3 bits pour l'exposant. On suppose qu'on utilise toujours la forme normalisée de cette représentation. Que peut-on dire sur l'erreur relative ?

- A Elle vaut au maximum 0.5 sur le domaine couvert
- B Elle vaut au maximum 1/5 sur le domaine couvert
- C Elle vaut au maximum 1/32 sur le domaine couvert // poids faible de la mantisse : 2⁻⁵
- D Elle vaut 1/32 pour la valeur min de l'exposant puis double pour chaque incrémentation de l'exposant

Question 11 : Soit Algo_X un algorithme recevant en entrée un entier strictement positif N et une liste L de taille N contenant des entiers positifs. Le caractère / désigne la division entière.

<p>Algo_X</p> <p>entrée : entier strictement positif N, liste L de taille N contenant des entiers positifs</p> <p>sortie : ???</p> <pre> z ← 0 Pour i de 1 à N a ← 0 x ← N Tant que (x > 0) a ← L[x] * 2 x ← x / 2 z ← z + a sortir: z </pre>
--

Quel est l'ordre de complexité de cet algorithme avec la notation de Landau ?

- A O(N log N) mais pas O(N) // boucle principale : N passages * boucle interne : Log2(N) passages
- B O(N) mais pas O(log(N))
- C O(log N) mais pas O(1)
- D O(N²) mais pas O(Nlog(N))

Question 12 : Quel est le résultat de Algo_X ci-dessus ?

- A le double de la somme des éléments de la liste L
- B la somme des éléments de la liste L élevés à la puissance N
- C la somme des éléments de la liste L multipliés par 2^N
- D le double du premier élément de la liste multiplié par N

Questions Ouvertes

Question 1: Liste des sommes partielles d'une liste L de N éléments

Soit L une liste non-vide de N entiers. On définit une liste P des sommes partielles, de même taille N que la liste L, de la manière suivante : chaque élément P(i) est égal à la somme des éléments de la liste L compris entre 1 et i (1 et i étant inclus dans cette somme):

$$P(i) = \sum_{j=1}^i L(j) , \text{ pour } i \text{ de } 1 \text{ à } N.$$

Par exemple, si L = {4, 10, 9, 2}, alors P = {4, 14, 23, 25}.

1) Proposez un algorithme qui calcule les sommes partielles.

Somme_partielle
entrée : entier $N > 0$, liste L d'entiers de taille N, Liste P de taille N contenant des zéros sortie : liste P des sommes partielles de L
// voici une solution parmi plusieurs variantes possibles
somme $\leftarrow 0$
Pour i de 1 à N
somme \leftarrow somme + L(i)
P(i) \leftarrow somme
Sortir : P

2) Justifier l'ordre de complexité de votre algorithme en fonction de N avec la notation de Landau en $O(\dots)$.

$O(N)$ car il y a une seule boucle parcourue N fois avec un nombre constant d'opération à chaque passage

3) on ajoute à l'algorithme un paramètre supplémentaire de largeur (Width) sous forme d'un entier strictement positif W, inférieur ou égal à N. Ce paramètre W indique le nombre maximum de termes à utiliser dans la somme partielle, à partir du terme d'indice i et en remontant vers le début de la liste L. On obtient ainsi une liste S dont l'élément S(i) est la somme des éléments de L compris entre L(i) et L(i-W+1). Par exemple, si W vaut 1, S(i) = L(i). Si W vaut 2, S(3)=L(3)+L(2). Si W est plus grand que 1 l'algorithme doit s'assurer que les indices utilisés sont toujours strictement positifs. Par exemple, si W vaut 5, alors on limite les termes pour S(3) avec S(3)= L(3)+L(2)+L(1).

Proposez un algorithme qui calcule les sommes partielles avec largeur W. Vous pouvez ré-utiliser votre algorithme de la question 1 mais cela n'est pas obligatoire.

Somme_partielle_W
entrée : entiers $N > 0$ et $0 < W \leq N$, liste L d'entiers de taille N, Liste S de taille N contenant des zéros sortie : liste S des sommes partielles de L avec une largeur de W
// voici une solution parmi plusieurs variantes possibles
somme $\leftarrow 0$
Pour i de 1 à N
somme \leftarrow somme + L(i)
Si $i > W$
somme \leftarrow somme - L(i-W)
S(i) \leftarrow somme
Sortir : S

Question 2: Fusion de deux listes triées

Soit un algorithme FusionListes qui reçoit en entrée un entier strictement positif N , deux listes A et B de N éléments triés dans l'ordre croissant et une liste C de taille $2N$ contenant des zéros. Le but de l'algorithme est fusionner les éléments des listes A et B dans la liste C de telle façon que la liste C soit dans l'ordre croissant. On conserve les doublons, c'est-à-dire que si un élément de la liste A se trouve aussi dans la liste B alors il apparaîtra deux fois dans la liste C .

1) Soit les listes suivantes pour A et B , indiquer quelle est la liste C attendue en sortie :

1.1) $A=\{-2, 5, 11, 49\}$, $B=\{8, 11, 20, 30\}$, $C= \{-2, 5, 8, 11, 11, 20, 30, 49\}$

1.2) $A=\{66, 70, 75, 88\}$, $B=\{3, 20, 33, 60\}$, $C= \{3, 20, 33, 60, 66, 70, 75, 88\}$

2) proposez un algorithme pour résoudre ce problème

FusionListes
entrée : entier $N>0$, listes A et B de taille N contenant des valeurs entières triées dans l'ordre croissant, liste C de taille $2N$ contenant des zéros
sortie : liste C contenant la fusion de A et B dans l'ordre croissant
<pre>// voici une solution possible parmi un grand nombre de variantes // un point important à garantir est de ne pas sortir de l'intervalle [1,N] pour accéder à A ou à B i ← 1 j ← 1 Tant que i ≤ N ET j ≤ N Si A(i) ≤ B(j) C(i+j-1) ← A(i) i ← i + 1 Sinon C(i+j-1) ← B(j) j ← j + 1 // on sort de la première boucle lorsqu'on a atteint la fin de A ou de B // ici on suppose qu'on a atteint la fin de A ; il suffit de recopier le reste de B dans C Tant que (j ≤ N) C(j + N) ← B(j) j ← j + 1 // ici on suppose qu'on a atteint la fin de B ; il suffit de recopier le reste de A dans C Tant que (i ≤ N) C(i + N) ← A(i) i ← i + 1 Sortir : C</pre>

3) Quel est l'ordre de complexité de votre algorithme avec la notation de Landau ? Argumentez votre choix.

$O(N)$ car il y a un seul parcours de chaque boucle avec $2N$ passages au maximum cumulés sur l'ensemble des boucles. le nombre d'opérations maximum par passage est constant et indépendant de N .