Nom : .... **Prénom :** ....

INFO I31, partiel 2012

Pour chaque question, répondez directement sur la feuille. Quand vous avez le choix, il n'y a qu'une seule bonne réponse par question. Lisez et comprenez les questions avant d'y répondre !

Question 1 – Déroulez l'algorithme d'Euclide pour calculer le PGCD de 156 et 180 :

Question 2 – Déroulez l'algorithme d'Euclide étendu (ou Bezout) pour $a = 180$ et $b = 156$. u et v sont tels que $au + bv = g = PGCD(a, b)$; $r = a \mod b$, et $q = \lfloor \frac{a}{b} \rfloor$ est le quotient de a par b :

a	b	r	q	g	u	v
180	156					

Question 3 – Citez les noms de 3 algorithmes de tri :

Question 4 – Citez les noms de 3 algorithmes calculant les plus courts chemins dans un graphe :

Question 5 – Un étudiant programme une méthode de tri. Son programme met 1 seconde pour trier 100 mille éléments, et 4 secondes pour trier 200 mille éléments. La complexité de son algorithme est :

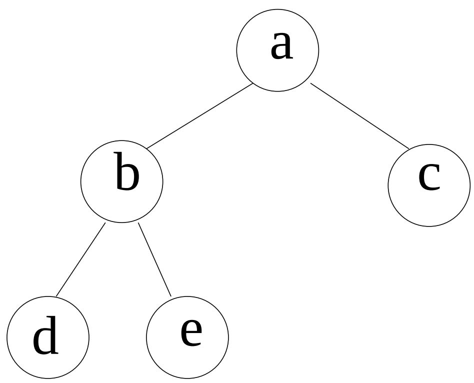
- ☐ $O(n^2)$, où n est le nombre d'éléments à trier
- ☐ $O(n)$, où n est le nombre d'éléments à trier
- ☐ $O(n \log n)$, où n est le nombre d'éléments à trier
- ☐ aucune des réponses précédentes n'est correcte.

Question 6 – Un étudiant programme une méthode de calcul de l'enveloppe convexe de n points en 2D. Son programme met 1 seconde pour $n = 100$ mille points, et 8 secondes pour $n = 200$ mille points. Son algorithme est en :

- ☐ $O(n^3)$; il s'agit vraisemblablement de la méthode naïve qui teste toutes les paires de points pour décider si elle donne une arête de l'enveloppe convexe.
- ☐ $O(n^2)$
- ☐ $O(n \log n)$
- ☐ aucune des réponses précédentes n'est correcte.

Question 7 – Donnez les formules nécessaires pour le calcul récursif de a^n ($a \in \mathbb{R}, n \in \mathbb{N}$). N'oubliez pas le ou les cas terminaux :

Question 8 – Le tri bulle (bubble sort) échange deux éléments consécutifs qui ne sont pas dans le bon ordre, tant que c'est possible. Donner l'ordre le plus favorable et l'ordre le plus défavorable pour cette méthode de tri. Fournir le nombre de permutations réalisées dans chacun des deux cas :



Question 9 – Donnez un ordre possible de visite des noeuds (feuilles incluses) de cet arbre binaire pour la traversée en largeur :

Question 10 – Donnez un ordre possible de visite des noeuds (feuilles incluses) de cet arbre binaire pour la traversée en profondeur :

Question 11 – Un de ces problèmes est difficile. Lequel :

- ☐ trouver un chemin qui passe par tous les sommets d'un graphe
- ☐ trouver un chemin qui passe par toutes les arêtes d'un graphe

Question 12 – Quand un problème est dans NP, alors :

- ☐ il est difficile de décider si une proposition de solution est une solution
- ☐ il est facile de décider si une proposition de solution est une solution

Question 13 – Y a t-il des problèmes impossibles à résoudre en informatique ? :

- ☐ oui, en voici deux :
- ☐ non, il n'y a pas de problème insoluble, seulement des problèmes difficiles, qui exigent beaucoup de temps pour être résolus

Question 14 – Un algorithme optimal de tri, n'utilisant que des comparaisons entre 2 éléments, nécessite :

- ☐ $O(n \log n)$ comparaisons pour trier n éléments
- ☐ $O(n^2)$ comparaisons
- ☐ $O(n)$ comparaisons
- ☐ un nombre exponentiel de comparaisons

Question 15 – Un tableau contient n entiers triés par ordre croissant ; la méthode optimale pour décider si le tableau contient un entier donné nécessite :

- ☐ $O(\log n)$ comparaisons (elle utilise la dichotomie)
- ☐ $O(n)$ comparaisons
- ☐ $O(n \log n)$ comparaisons

Question 16 – Quelle structure de données utiliser pour répondre en temps constant à la requête précédente :

- ☐ une table de hachage ($hashtable$)
- ☐ un arbre binaire
- ☐ un tas ($heap$)
- ☐ une liste
- ☐ ce n'est pas possible

Question 17 – Un tableau, trié, contient n entiers ; la méthode optimale pour trouver le nombre maximum dans le tableau qui est inférieur (strictement) à un nombre donné, ou détecter que ce nombre n'existe pas, nécessite :

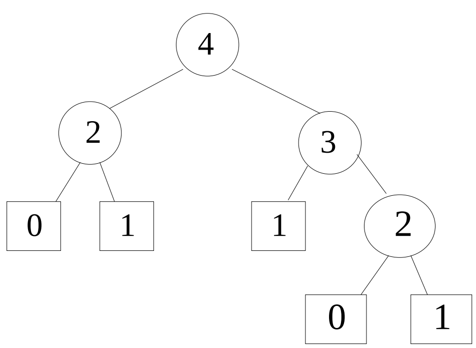
- ☐ $O(\log n)$ comparaisons (elle utilise la dichotomie)

- ☐ $O(n)$ comparaisons
- ☐ $O(1)$ comparaisons

Question 18 – La méthode la plus rapide pour calculer le n ième terme de la suite de Fibonacci, F_n (rappel : $F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-1}$ quand $n > 1$) nécessite :

- ☐ $O(\log n)$ opérations sur des entiers
- ☐ $O(n)$ opérations sur des entiers
- ☐ $O(F_n)$ opérations sur des entiers

Question 19 – Proposez une méthode rapide pour calculer K_n, où K_0, K_1 sont donnés, et $K_n = a_1 K_{n-1} + a_2 K_{n-2}$ quand $n > 1$:



Question 20 – L'arbre de Fibonacci T_4 est dessiné ci-dessus. T_0 est une feuille portant l'étiquette 0, T_1 est une feuille portant l'étiquette 1 ; pour $n > 1$, T_n est un noeud binaire, dont le fils gauche est T_{n-2} et dont le fils droit est T_{n-1}. Donnez une formule récursive pour le nombre d'éléments (feuilles ou sommets), noté $|T_n|$, de T_n, pour $n > 1$:

Question 21 – Définissez U_n le nombre de feuilles étiquetées 1 de T_n. Que constatez-vous :

Question 22 – Définissez Z_n le nombre de feuilles étiquetées 0 de T_n :

Question 23 – Remplissez le tableau suivant, où U_n est le nombre de feuilles étiquetées 1 de T_n, Z_n le nombre de feuilles étiquetées 0 de T_n :

n	0	1	2	3	4	5	6	7	8	9
$\|T_n\|$										
U_n										
Z_n										

Question 24 – Prolongez de façon logique la suite de Fibonacci aux nombres négatifs :

i	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5
F_i							0	1	1	2	3	5

Question 25 – Vous constatez que, pour $n \in \mathbb{N}$, F_{-2n} est égal à :

Question 26 – Vous constatez que, pour $n \in \mathbb{N}$, F_{-2n-1} est égal à :

Question 27 – Soient $\phi = \frac{1}{2}(1 + \sqrt{5}) \approx 1.618$ et $\phi' = \frac{1}{2}(1 - \sqrt{5}) \approx -0.618$; ce sont les deux racines de $x^2 - x - 1 = 0$. Définissons $f(n) = \frac{1}{\sqrt{5}}(\phi^n - \phi'^n)$. Donnez la valeur de $f(0), f(1), f(2), f(3)$:

Question 28 – Par récurrence, prouvez que $F_n = f(n)$:

Question 29 – Le temps d'exécution d'un algorithme est $T(n)$ pour une donnée de taille n, où $T(1) = 1$ et $T(n) = 3T(n/2) + n$. Prouver par récurrence que $T(2^k) = 3^{k+1} - 2^{k+1}$:

Question 30 – Prouvez que $3^{\log_2 n}$ est en $O(n^{log_3 2})$ (Remarque : $\log_3 2 \approx 1.5849625$) :

Question 31 – Calculer le produit scalaire entre 2 vecteurs de n nombres flottants $u = (u_i)$ et $v = (v_i)$ nécessite :

- ☐ $O(\log n)$ opérations flottantes
- ☐ $O(n)$ opérations flottantes
- ☐ $O(n \log n)$ opérations flottantes
- ☐ $O(n^2)$ opérations flottantes

Question 32 – Calculer le produit entre une matrice carrée quelconque de taille $n \times n$, contenant n^2 nombres flottants, et un vecteur colonne de n éléments (n nombres flottants) nécessite :

- ☐ $O(n \log n)$ opérations flottantes
- ☐ $O(n)$ opérations flottantes
- ☐ $O(n^2)$ opérations flottantes
- ☐ ce n'est pas toujours possible

Question 33 – Calculer le produit entre deux matrices carrées de taille $n \times n$, en appliquant la formule : $C_{lc} = \sum_{k=1}^{n} A_{lk} \times B_{kc}$, nécessite :

- ☐ $O(n \log n)$ opérations flottantes
- ☐ $O(n)$ opérations flottantes
- ☐ $O(n^2)$ opérations flottantes
- ☐ $O(n^3)$ opérations flottantes
- ☐ ce n'est pas toujours possible (la matrice doit être inversible, par exemple).

Question 34 – La méthode de puissance rapide pour calculer M^k (M est une matrice carrée de taille $n \times n$, et k est un entier naturel) nécessite :

- ☐ inévitablement $k - 1$ (donc $O(k)$) multiplications de matrices carrées de taille $n \times n$
- ☐ $O(\log k)$ multiplications de matrices carrées de taille $n \times n$
- ☐ $O(k \log k)$ multiplications de matrices carrées de taille $n \times n$
- ☐ ce n'est pas toujours possible (la matrice doit être inversible, par exemple).

Question 35 – Trouver l'élément le plus petit dans une liste non ordonnée de taille $n > 0$ nécessite :

- ☐ $O(1)$ comparaisons
- ☐ $O(n^2)$ comparaisons
- ☐ $O(n)$ comparaisons
- ☐ $O(\log n)$ comparaisons

Question 36 – Trouver l'élément le plus petit dans une liste triée par ordre croissant et de taille $n > 0$ nécessite :

- ☐ $O(1)$ comparaisons
- ☐ $O(n^2)$ comparaisons
- ☐ $O(n)$ comparaisons
- ☐ $O(\log n)$ comparaisons

Question 37 – Un étudiant programme le tri rapide ("quicksort") d'une liste L ainsi : si L contient moins de 2 éléments, alors elle est déjà triée. Sinon, l'étudiant choisit un élément p dans L. Il partitionne L en 3 sous listes, la liste L_1 des éléments dans L inférieurs à p, la liste L_2 des éléments dans L égaux à p, la liste L_3 des éléments dans L supérieurs à p. Il trie récursivement les 3 listes, puis concatène les résultats. Qu'en pensez-vous :

- ☐ la méthode est correcte mais la complexité est modifiée
- ☐ la méthode est correcte et la complexité est inchangée
- ☐ la méthode est incorrecte ; elle boucle quand L contient deux (ou davantage) éléments égaux à p.
- ☐ la méthode est incorrecte car L_2 n'est jamais vide, puisqu'elle contient p.

Question 38 – Pour le calcul de l'arbre couvrant minimum d'un graphe connexe, un étudiant propose l'algorithme suivant : si le graphe est un arbre, alors insérer cet arbre dans l'arbre couvrant minimum ; sinon décomposer le graphe G en 2 graphes de taille à peu près moitié, G_1, et G_2, tous deux connexes ; calculer l'arbre couvrant minimum de G_1 ; calculer l'arbre couvrant minimum de G_2 ; joindre ces deux arbres par l'arête de coût minimum entre G_1 et G_2 (cette arête a un sommet dans G_1 et un sommet dans G_2). Que pensez-vous de cet algorithme :

- ☐ il est correct
- ☐ il est difficile de partitionner un graphe connexe en deux sous graphes connexes ayant (à peu près) deux fois moins de sommets ; c'est pourquoi cet algorithme n'est pas utilisé
- ☐ il est incorrect, et voici un contre exemple simple (au plus 4 sommets !) :

Question 39 – Une matrice de Vandermonde a la structure suivante :

$$M = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w & w^2 & \dots & w^{n-1} \\ 1 & w^2 & w^4 & \dots & w^{2(n-1)} \\ \dots & \dots & \dots & \dots & \\ 1 & w^{n-1} & w^{2n-2} & \dots & w^{(n-1)^2} \end{pmatrix}$$

où n est une puissance de 2. Si de plus w est une racine n ième de l'unité, alors le produit avec un vecteur colonne quelconque de taille n :

- ☐ peut se faire en $O(n \log n)$ (avec l'algorithme de la transformée rapide de Fourier)
- ☐ ne peut pas se faire en moins que $O(n)$
- ☐ ne peut pas se faire en moins que $O(n^2)$

Question 40 – Pour $n = 8$ (donc $w^8 = 1$), écrire la ligne de la matrice de Vandermonde commençant par 1 puis w^3, en simplifiant (c'est à dire en éliminant les puissances plus grandes que 7) :

Question 41 – Idem, pour $n = 8$, écrire la ligne de la matrice de Vandermonde commençant par 1 puis w^6 :

Question 42 – Dans le problème de la somme, un ensemble E de n entiers positifs $e_1 \geq e_2 \geq \dots \geq e_n$

est donné, ainsi qu'un entier $0 < S < \sum_1^n e_i$. Il faut trouver le sous ensemble X de E dont la somme des éléments est maximum, mais inférieure ou égale à S. Pour tout ensemble K, on note $\sigma(K)$ la somme des éléments de K. L'algorithme glouton calcule $X_0 = \emptyset$ (donc $s(X_0) = 0$) ; puis, pour i de 1 à n, il calcule $X_i = X_{i-1} \cup \{e_i\}$ si $e_i + \sigma(X_{i-1}) \leq S$, et $X_i = X_{i-1}$ sinon. Avec $E = 100, 25, 20, 10, 1, 1, 1$ et $S = 40$, donnez les valeurs de $X_0, X_1, \ldots X_7$, et les $\sigma(X_i)$ correspondants :

Question 43 – Cet algorithme donne-t-il un sous ensemble X_n :

- ☐ toujours correct ($s(X_n) \leq S$), mais pas forcément optimal : donnez un exemple simple ($n < 5$) où X_n n'est pas optimal
- ☐ toujours correct, et toujours optimal
- ☐ pas toujours correct, mais optimal : donnez un exemple simple ($n < 5$)
- ☐ souvent correct, et souvent optimal
- ☐ ni correct ni optimal : donnez un exemple simple ($n < 5$)
- ☐ toutes les réponses précédentes sont fausses

Question 44 – Arthur conjecture que si, dans un graphe non orienté, tous les sommets distincts soit sont voisins, soit ont un voisin en commun, alors il existe au moins un sommet qui est voisin de tous les autres. Que pensez vous de cette conjecture :

- ☐ elle est vraie
- ☐ elle est fausse et vous en dessinez un contre exemple simple