EXAMEN D'ALGORITHMIQUE ET COMPLEXITE

Master Informatique, première année

Janvier 2008

Documents autorisés: une page A4 recto verso.

TOUTES LES QUESTIONS ONT ETE VUES EN COURS, TD OU TP.

Répondez aux questions, dans l'ordre, en donnant le numéro des questions. Ecrivez lisiblement. Répondez en une ligne à chaque question, sauf mention contraire de l'énoncé.

Barême : les 10 premières questions sont notées sur 5. Les 15 autres sont notées sur 15, à raison d'un point par question.

1. Citer un algorithme en $O(\log n)$.
2. Citer un algorithme en $O(n)$.
3. Citer un algorithme en $O(n^3)$.
4. Citer un algorithme en $O(n \log n)$.
5. Citer un algorithme en $O(n^2)$.
6. Un algorithme en $O(n^3)$ met T secondes pour n données; combien de secondes met il pour $10n$ données?
7. Citer un algorithme en $O(2^n)$.
8. Citer 3 algorithmes polynomiaux pour calculer les plus courts chemins dans un graphe.
9. Citer un algorithme de tri d'entiers, plus rapide que $O(n \log n)$.
10. Définissez le principe de l'attribut-fonction (appelez la fonction $f(x)$ si nécessaire). Quel est l'intérêt? Donnez un exemple. Répondre en 5 lignes au plus.
11. Donnez en 5 lignes maximum le principe des tables de hachage.
12. Définissez la fermeture transitive d'un sommet S dans un graphe orienté, en 2 lignes.

13. Quel est l'intérêt d'un arbre équilibré de recherche sur un arbre non équilibré (une ligne au maximum).

14. Un exemple de programmation dynamique a été programmé en ocaml. Lequel?

15. Donner 2 exemples de problèmes solubles par "backtrack" (exploration arborescente).

16. Citer 2 méthodes de tri en $O(n \log n)$. Attention: le tri rapide (quicksort) est parfois quadratique.

17. Un problème est dans NP s'il est possible de contrôler une solution en temps polynomial. Qu'est ce qu'un problème NP-complet? (2 lignes)

18. Citer un exemple de problème NP-complet.

19. Définissez le tri topologique (ou extension linéaire) d'un graphe orienté. Quelle est la condition d'existence ? Quelle commande unix utilise cette notion? (3 lignes)

20. L'ensemble des sommets d'un graphe biparti peut être partitionné en 2 sous ensembles: aucune arête du graphe ne lie deux sommets dans le même sous ensemble. Qu'est ce qu'un couplage maximal dans un graphe biparti? A quel problème la recherche d'un couplage maximal est elle équivalente? Donnez une application. 4 lignes.

21. Citer 2 méthodes pour résoudre les problèmes de programmation linéaire.

22. Donner en *français* un algorithme itératif d'énumération d'arbre de recherche, ou arbre de décision, en une douzaine de lignes. Vous pouvez supposer qu'il faut trouver les valeurs de $p_1 \in E_1, p_2 \in E_2, \ldots p_n \in E_n$, où n est fixé, et les ensembles E_i sont finis et donnés. Un prédicat (nommé accepte) décide si $P_1 \in E_1, P_2 \in E_2, \ldots P_n \in E_n$ vérifie ou non les contraintes du problème. Ce prédicat accepte en argument une assignation partielle (certains champs P_i ne sont pas fixés).

23. Permutation de n objets: soit un ensemble de n objets numérotés de 1 à n. Il existe $n!$ permutations de ces objets, qui peuvent être représentées chacune pa un n-uplet (une liste, ou un tableau). Donner un algorithme en *français* qui imprime successivement et sans répétition toutes les permutations de n objets, en une douzaine de lignes.

24. Donner la définition d' un CSP (Problème de Satisfaction de Contraintes) avec un exemple simple. 5 lignes.

25. Citer 2 méthodes de résolution de CSP

1. Citer un algorithme en $O(\log n)$. Réponse : Chercher un élément dans un tableau trié de n éléments, par la méthode dichotomique.

2. Citer un algorithme en $O(n)$. Réponse : Lire les données. Produit scalaire de 2 vecteurs de taille n. Fusion de deux listes triées de longueur n.

3. Citer un algorithme en $O(n^3)$. Réponse : Produit de 2 matrices carrées de taille $n \times n$. Résolution de systèmes linéaires par la méthode de Gauss ou une variante (LUP).

4. Citer un algorithme en $O(n \log n)$. Réponse : Tri par fusion de n éléments.

5. Citer un algorithme en $O(n^2)$. Réponse : Somme de 2 matrices carrées de taille $n \times n$. Tri naïf.

6. Un algorithme en $O(n^3)$ met T secondes pour n données; combien de secondes met il pour $10n$ données ? Réponse : $10^3T = 1000T$

7. Citer un algorithme en $O(2^n)$. Réponse : Tester tous les possibles pour satisfaire une formule logique $F(x_1, \ldots x_n)$.

8. Citer 3 algorithmes polynomiaux pour calculer les plus courts chemins dans un graphe. Réponse : Dijkstra, Ford (ou Bellman-Ford), Dantzig, Floyd (ou Floyd-Warshall), Johnson-Dijkstra

9. Citer un algorithme de tri d'entiers, plus rapide que $O(n \log n)$. Réponse : Le "radix sort", ou tri par base.

10. Définissez le principe de l'attribut-fonction (appelez la fonction $f(x)$ si nécessaire). Quel est l'intérêt? Donnez un exemple. Répondre en 5 lignes au plus. Réponse : La valeur de la fonction est calculée et stockée quand elle est demandée pour la première fois. Une même valeur n'est donc calculée qu'une fois. Ainsi le calcul naïf (il utilise la formule récursive) de Fibonacci(n) devient en $O(n)$. Avantages: seules les valeurs nécessaires sont calculées; et le programmeur n'a pas à déterminer lui même dans quel ordre calculer les valeurs de la fonction, dans les cas compliqués de récursion (ex: sac à dos par programmation dynamique).

11. Donnez en 5 lignes maximum le principe des tables de hachage. Réponse : Elles permettent typiquement d'attacher des informations (appelées attributs) à des chaînes de caractères (noms de personnes; noms de variables ou de fonctions, pour un compilateur). Un entier, appelé clef de hachage, doit pouvoir être calculé à partir de la chaîne; c'est le numéro du "tiroir" dans lequel est rangé le couple: (chaîne, attribut); les tiroirs sont, typiquement, un tableau de liste de couples (chaîne, attribut). On espère que les chaînes différentes ont des clefs différentes.

12. Définissez la fermeture transitive d'un sommet S dans un graphe orienté, en 2 lignes. Réponse : La fermeture transitive de S (ou d'un sous ensemble S de sommets $s \in S$) est l'ensemble des sommets t tels qu'il existe un chemin depuis $s \in S$ vers t dans le graphe.

13. Quel est l'intérêt d'un arbre équilibré de recherche sur un arbre non équilibré (une ligne au maximum). Réponse : Les opérations élémentaires (insertion, suppression, recherche) sont garanties être en temps $O(\log n)$.

14. Un exemple de programmation dynamique a été programmé en ocaml. Lequel? Réponse : Parenthésage optimal de produits de matrices; une variante du sac à dos avec poids entiers).

15. Donner 2 exemples de problèmes solubles par "backtrack" (exploration arborescente). Réponse : Coloriage des sommets d'un graphe (sudoku), le compte est bon, chemin ou circuit hamiltonien, 3-SAT.

16. Citer 2 méthodes de tri en $O(n \log n)$. Attention: le tri rapide (quicksort) est parfois quadratique. Réponse : Le tri fusion (mergesort), le tri par tas (heapsort), le tri par utilisation d'un arbre équilibré (AVL, rouge-noir, etc).

17. Un problème est dans NP s'il est possible de contrôler une solution en temps polynomial. Qu'est ce qu'un problème NP-complet? (2 lignes) Réponse : Un problème est NP-complet ssi tout problème de NP s'y réduit, en temps polynomial. Donc trouver une méthode polynomiale pour un problème NP-complet permettrait de résoudre tous les problèmes de NP en temps polynomial.

18. Citer un exemple de problème NP-complet. Réponse : 3-SAT. Coloriabilité et coloriage des sommets d'un graphe en 3 couleurs. Chemin ou circuit hamiltonien.

19. Définissez le tri topologique (ou extension linéaire) d'un graphe orienté. Quelle est la condition d'existence ? Quelle commande unix utilise cette notion? (3 lignes). Réponse : C'est un ordre sur les sommets tel que pour tout arc $a \rightarrow b$ du graphe, $a > b$. Le graphe ne doit pas contenir de circuit. La commande make utilise cette notion pour gérer les dépendances entre tâches (compilations, éditions de liens).

20. L'ensemble des sommets d'un graphe biparti peut être partitionné en 2 sous ensembles: aucune arête du graphe ne lie deux sommets dans le même sous ensemble. Qu'est ce qu'un couplage maximal dans un graphe biparti? A quel problème la recherche d'un couplage maximal est elle équivalente? Donnez une application. 4 lignes. Réponse : Un couplage est un sous ensemble des arêtes, tel qu'un sommet appartient à au plus 1 arête du couplage. Trouver un couplage maximum (en nombre d'arêtes) est un problème de flot maximum. Application: affectation optimale de ressources non partageables; détection des sous- ou sur-contraintes dans des systèmes d'équations.

21. Citer 2 méthodes pour résoudre les problèmes de programmation linéaire. Réponse : Simplexe dû à Dantzig. Méthode de l'ellipsoïde. Méthode du simplexe englobant (l'ellipsoïde englobant est remplacé par un simplexe). Méthodes par l'intérieur.

22. Donner en *français* un algorithme itératif d'énumération d'arbre de recherche, ou arbre de décision, en une douzaine de lignes. Vous pouvez supposer qu'il faut trouver les valeurs de $p_1 \in E_1, p_2 \in E_2, \ldots p_n \in E_n$, où n est fixé, et les ensembles E_i sont finis et donnés. Un prédicat (nommé accepte) décide si $P_1 \in E_1, P_2 \in E_2, \ldots P_n \in E_n$ vérifie ou non les contraintes du problème. Ce prédicat accepte en argument une assignation partielle (certains champs P_i ne sont pas fixés). Réponse :

On appelle état une instanciation partielle de $P_1, P_2, \ldots P_k$. L'état vide est celui où aucun champ n'a de valeur. Empiler l'état vide. Tant que la pile est non vide, dépiler l'état en sommet de pile. S'il est solution (tous les champs sont affectés et l'état est accepté par le prédicat), l'ajouter à l'ensemble des solutions. Sinon, déterminer quel champ fixer (par exemple: le premier champ non fixé, ou bien celui qui a le moins de valeurs possibles) et la liste des valeurs

possibles pour ce champ. Empiler tous les états fils correspondants; seuls sont empilés des états acceptés par le prédicat; il est donc possible qu'aucun état fils ne soit empilé.

L'algorithme précédent effectue un parcours en profondeur sur l'arbre de recherche. Remplacer la pile par une file donne un parcours en largeur.

Pour générer les fils et parcourir l'arbre en profondeur, on peut aussi utiliser deux fonctions, donnant le fils et le frère. Cela revient à binariser l'arbre de recherche. Avantage : la pile devient inutile. Voir le programme C de la question 23.

23. Permutation de n objets: soit un ensemble de n objets numérotés de 1 à n. Il existe $n!$ permutations de ces objets, qui peuvent être représentées chacune pa un n-uplet (une liste, ou un tableau). Donner un algorithme en *français* qui imprime successivement et sans répétition toutes les permutations de n objets, en une douzaine de lignes. Réponse :

Une permutation partielle est un suffixe (une fin) d'une permutation. Empiler la permutation vide. Tant que la pile est non vide, dépiler une permutation partielle p. Si elle est complète, l'insérer dans l'ensemble des solutions. Sinon, pour chaque élément e absent de la permutation partielle p, empiler la concaténation de e et de p.

Variante: gérer une pile de couples (l, p) où p est une permutation partielle, et l la liste des éléments absents de p. Le programme en ocaml (il n'était pas demandé):

```
let permutations liste =
   let rec working known_solutions stack = match stack with
   | [] -> known_solutions
   | (l,p)::qstack -> match l with
     (* l est la liste des absents de p, p est la permutation partielle *)
        | [] -> working (p::known_solutions) qstack
        | _ -> working known_solutions
                       ((List.map (function e->
                        let l_moins_e =  List.filter (function x-> x<>e) l in
                        (l_moins_e, e::p) ) l) @ qstack)
in let initial_stack= [ (liste, []) ] in working [] initial_stack;;

# permutations [1;2;3];;
- : int list list =
[[1; 2; 3]; [2; 1; 3]; [1; 3; 2]; [3; 1; 2]; [2; 3; 1]; [3; 2; 1]]
```

On rappelle l'autre solution vue en TP d'ocaml.

```
(* insert e [a; b; c]: rend la liste [ [e;a;b;c]; [a;e;b;c]; [a;b;e;c]; [a;b;c;e]] *)
let rec insert e liste = match liste with
| [] -> [[ e]]
| t::q -> (e::liste)::(List.map (function y -> t::y) (insert e q));;

let rec permutations l = match l with [] -> [[]]
| t::q -> List.flatten (List.map (function permu_sans_t-> insert t permu_sans_t)
                                 (permutations q));;
```

Finalement, voici une solution procédurale en C avec fils ainé et frère, sans pile (elle n'était pas demandée) :

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
struct etat { int P[10]; int k; int n; /* permutation de 1...n; P[1..k] fixes */ };
void initialize( etat &e, int n ) { e.n=n; e.k=0; }
void etat_fils( etat & e) { assert( e.k<e.n); e.k++; e.P[ e.k]=1; }
void etat_frere( etat & e) { for (; e.k!= 0 && e.P[e.k] == e.n; --e.k );
                 if (0 != e.k) ++e.P[ e.k ] ; }
int accepte_etat( etat &e) /* P est correct ssi P[i<k] <> P[k] */
{   for (int i=1; i<e.k; i++) if (e.P[i]==e.P[e.k]) return 0;
    return 1; }
int etat_terminal( etat &e) { return e.k==e.n; }
int terminaison( etat &e) { return 0==e.k; }
void print_etat( etat &e)  { for( int k=1; k<=e.k; k++) printf("%d ", e.P[k]); printf("\n"); }
void new_solution( etat &e) {   printf( "solution:");  print_etat( e); }

void permutations( int n)
{   etat e; initialize( e, n);
    do{ printf("etat="); print_etat( e);
        int termine= etat_terminal( e);
        int accepte= accepte_etat( e);
        if (accepte) if  (termine) new_solution( e);
                     else etat_fils( e);
        if (termine || !accepte) etat_frere( e);
    } while (!terminaison( e)); }

int main( int argc, char **argv)
{   int n; int P[20];
    if (argc==2) n=atoi( argv[1]); else n=4;
    permutations( n);
    return 0; }
```

24. Donner la définition d' un CSP (Problème de Satisfaction de Contraintes) avec un exemple simple. 5 lignes. Réponse : Trouver les valeurs x_i satisfaisant des contraintes données, et/ou optimisant un critère donné. Les domaines des x_i peuvent être continus ou discrets (entiers, booléens). Exemple: trouver un couplage optimal; résoudre un puzzle; trouver les permutations vérifiant certaines contraintes (ex: tous les éléments doivent être déplacés); trouver le circuit hamiltonien le plus court...

25. Citer 2 méthodes de résolution de CSP. Réponse : Backtrack. Métaheuristiques: recherche locale, recuit simulé, tabou, algorithmes évolutionnaires.