Ecole Normale Supérieure de Lyon (H')

Epreuve d'algorithmique – Mathématiques appliquées

(Durée : 3 heures)

Important

Il est recommandé de lire l'ensemble du sujet avant de commencer la rédaction. Bien que traitant du même thème, les différentes parties sont largement indépendantes. Il est donc souhaitable de répartir son temps équitablement entre elles.

Les questions concernant la ***mise en œuvre des algorithmes*** *en* PASCAL *ne seront prises en considération que si la description algorithmique de la méthode est correcte et suffisamment complète : sachez expliquer vos programmes (un dessin, un schéma ou un diagramme sont aussi des moyens d'expliquer vos algorithmes).*

On demandera souvent d'exprimer le ***coût d'un algorithme*** *écrit en* PASCAL *: il s'agira toujours du nombre d'opérations élémentaires* $(+, -, \times, \div,$ *test, accès à un tableau,...)* *exprimé en ordre de grandeur et sans distinction des opérations. Exemple : le produit scalaire de deux vecteurs de taille* N *est de* $O(N)$*, c'est-à-dire qu'il suffit de l'ordre de* N *opérations élémentaires pour effectuer un produit scalaire de deux vecteurs de taille* N*. Similairement, le coût de la multiplication usuelle de deux matrices* $N \times N$ *est de* $O(N^3)$*. Rappel: si* f *et* g *sont deux fonctions de* $\mathbb{N}$ *dans* $\mathbb{R}^+$*, on note* $f = O(g)$ *s'il existe* $c \in \mathbb{R}^+$ *et* $n_0 \in \mathbb{N}$ *tel que* $f(n) \leq cg(n)$ *pour tout* $n \geq n_0$*.*

Etant donnée une matrice M*,* $M_{i,j}$ *désigne l'élément de* M *situé sur la ligne* i *et la colonne* j*.*

Introduction

Ce sujet traite des réseaux d'ordinateurs et de l'aptitude de ceux-ci à supporter des communications rapides. Un réseau est modélisé par un *graphe* dont les *sommets* représentent les ordinateurs et les *arêtes* représentent les liens de communication entre les ordinateurs. Par analogie, un réseau de métro peut se modéliser par un graphe dont les sommets sont les stations et les arêtes sont les voies reliant ces stations. Nous concentrerons en particulier notre attention sur les réseaux en *arbre*, c'est-à-dire les réseaux dans lesquels il n'y a qu'un seul *chemin* entre toute paire de sommets.

Il est impossible de connecter directement toutes les stations d'un réseau métropolitain. Ainsi, voyager d'une station à une autre nécessite de traverser d'autres stations intermédiaires. De même, il est impossible de connecter directement tous les ordinateurs d'un réseau, ainsi un message transitant d'un ordinateur à un autre devra traverser des ordinateurs intermédiaires qui joueront le rôle de relais pour ce message.

Tout comme un voyageur se rendant d'une station de métro à une autre estime son temps de parcours par le nombre de stations intermédiaires traversées, le temps de transfert d'un message d'un ordinateur à un autre peut s'estimer en fonction du nombre d'ordinateurs relais traversés. C'est une approximation satisfaisante dans un réseau d'ordinateurs car le temps de traversée d'un relais ne dépend ni du lien par lequel le message est arrivé ni de celui par lequel il va repartir (au contraire du métro où il existe une notion de "changement"). C'est cette notion de temps qui va nous servir à calculer le coût des communications dans les réseaux d'ordinateurs.

Nous nous intéresserons également à la structure du réseau en considérant des paramètres tels que son *degré* (le nombre de liens de communication partant et arrivant en chaque sommet) et son *diamètre* (le plus long chemin entre deux sommets du réseau).

Les termes précédemment utilisés sont précisément définis ci-après.

Terminologie et notations

Un *graphe* G est défini par un ensemble fini quelconque S dont les éléments sont appelés les *sommets*, et par un sous-ensemble A de $S \times S$ dont les éléments sont appelés les *arêtes*. (Les couples définissant les arêtes ne sont pas ordonnés, c'est-à-dire $(x, y) = (y, x)$.) La cardinalité de S est appelé l'*ordre* du graphe.

Etant donnés un graphe $G = (S, A)$ et un sommet x de G, un *voisin* de x est un sommet y vérifiant $(x, y) \in A$. On dit aussi que x est *connecté* à y. On note $\Gamma(x)$ l'ensemble des voisins de x. Le *degré* de x est noté $\deg(x)$ et satisfait $\deg(x) = |\Gamma(x)|$. Le *degré maximum* de G est noté Δ et satisfait $\Delta = \max_{x \in S} \deg(x)$.

Un *chemin* dans G est une suite finie $x_0, e_1, x_1, e_2, x_2, \ldots, e_k, x_k$, $k \geq 0$, telle que $x_i \in S$, pour tout $i = 0, \ldots, k$ et, si $k > 0$, $e_i = (x_{i-1}, x_i) \in A$ pour tout $i = 1, \ldots, k$. Le paramètre k est la *longueur* du chemin. Pour toute paire de sommets x et y de G, la distance de x à y est la longueur d'un plus court chemin entre x et y. Elle est notée $d(x, y)$. S'il n'y a pas de chemin dans G entre deux sommets x et y, alors on fixe $d(x, y) = +\infty$. L'*excentricité* d'un sommet x est notée $\text{exc}(x)$ et satisfait $\text{exc}(x) = \max_{y \in S} d(x, y)$. Le *diamètre* de G est noté D et satisfait $D = \max_{x \in S} \text{exc}(x)$. Si $D \neq +\infty$, alors G est dit *connexe*.

Dans la suite, les graphes seront toujours supposés connexes. Ils seront également toujours sans *boucle*, c'est-à-dire : $\forall x \in S, (x, x) \notin A$.

Partie 1. Matrice d'adjacence

Motivation : comprendre la structure d'un réseau.

Une façon naturelle de représenter un graphe $G = (S, A)$ d'ordre N est de considérer sa *matrice d'adjacence*. Cette matrice M de taille $N \times N$ est obtenue en numérotant arbitrairement les sommets de 1 à N et vérifie $M_{i,j} = 1$ si et seulement si les sommets i et j sont voisins, $M_{i,j} = 0$ sinon. Nous avons donc la structure de données suivante :

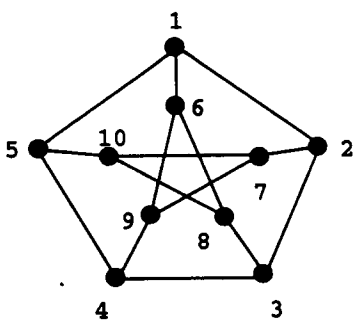
```
cons Nmax = 1024 ; (* par exemple *)
type graphe = array [1..Nmax,1..Nmax] of integer ;
var G : graphe ;
```

Question 1.1.

a) Tracer le graphe de matrice d'adjacence

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

b) Donner une matrice d'adjacence du graphe suivant :



c) Ecrire une fonction PASCAL

function degre_max (G : graphe ; N : integer) : integer ;

qui, étant donné un graphe G d'ordre N calcule son degré maximum. Quel est le coût de cette fonction ?

Question 1.2.

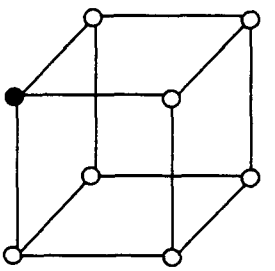
a) Montrer que pour tout entier $d \geq 1$, et pour toute paire i, j, $(M^d)_{i,j}$ est égal au nombre de chemins de longueur d distincts reliant les deux sommets i et j.

b) Déduire de la question (a) un algorithme permettant de calculer l'excentricité d'un sommet. Indiquer pourquoi un tel algorithme se révélerait cependant coûteux.

Dans la question suivante, nous allons montrer comment il est possible de calculer l'excentricité d'un sommet de façon plus efficace que la méthode de la question 1.2.b.

Question 1.3.

a) Recopier la figure ci-dessous en indiquant à côté de chaque sommet sa distance par rapport au sommet marqué en noir. Calculer l'excentricité du sommet marqué en noir.



b) Déduire de 1.3.a un algorithme permettant de calculer l'excentricité d'un sommet quelconque d'un graphe. Ecrire une fonction PASCAL

function excentricite (G : graphe ; N,x : integer) : integer ;

qui, étant donnés un graphe G d'ordre N et un sommet de G numéroté x, calcule l'excentricité de ce sommet. Quel est le coût de cette fonction ?

Partie 2. Graphe de Moore

Motivation : construction de grands réseaux satisfaisant à des contraintes technologiques raisonables (faible degré) mais permettant des communications rapides (faible diamètre).

Soit un graphe G de diamètre D et de degré maximum Δ. On note $N(\Delta, D)$ le nombre maximum de sommets que peut posséder un tel graphe.

Question 2.1.

a) Calculer $N(2, D)$ *et* $N(\Delta, 1)$.

b) Montrer que $N(\Delta, 2) \leq \Delta^2 + 1$.

Nous allons nous restreindre au cas $D = 2$ et chercher à déterminer s'il existe des graphes de diamètre 2, de degré maximum Δ et de $\Delta^2 + 1$ sommets. Ces graphes sont dits *graphes de Moore.*

Question 2.2.

a) Montrer qu'un graphe de Moore G *de degré maximum* Δ *satisfait* $\deg(x) = \Delta$ *pour tout sommet* x *de* G.

b) Montrer qu'entre tout couple de sommets x *et* y *d'un graphe de Moore. il n'existe qu'un seul chemin de longueur* $d(x, y)$.

c) Montrer que la matrice d'adjacence M *d'un graphe de Moore vérifie* $M^2 + M - (\Delta - 1)I = J$ *où* J *est la matrice* $N \times N$ *dont tous les coefficients sont égaux à 1 et* I *est la matrice identité.*

Question 2.3.

a) Déterminer les valeurs propres de J *et les sous-espaces propres associés.*

b) Montrer que Δ *est valeur propre de* M.

c) Montrer que la multiplicité de Δ *est 1.*

d) Montrer que les autres valeurs propres de M *sont solutions de* $x^2 + x - \Delta + 1 = 0$.

e) Montrer qu'il existe deux entiers positifs ou nuls m_1 *et* m_2 *vérifiant* $m_1 + m_2 = N - 1$ *et* $\Delta + m_1 r_1 + m_2 r_2 = 0$ *où* r_1 *et* r_2 *sont les solutions de l'équation de la question précédente.*

Question 2.4.

a) Exprimer m_1 *et* m_2 *en fonction de* Δ. *En déduire que si* $\Delta \neq 2$, *alors* Δ *est nécessairement de la forme* $k^2 + k + 1$ *avec* $k^4 \equiv 1 (mod\, 2k + 1)$.

b) Montrer que pour tout entier positif k, $16k^4 \equiv 1 (mod\, 2k + 1)$. *En déduire qu'un graphe de Moore vérifie nécessairement* $\Delta \in \{2, 3, 7, 57\}$.

c) Existe-t-il un graphe de Moore de diamètre 2 et degré maximum 3 ?

Remarque : *il existe un graphe de Moore de diamètre 2 et degré 7, mais personne ne sait s'il existe un graphe de Moore de diamètre 2 et degré 57 (si un tel graphe existe, il a 3250 sommets.)*

Terminologie et notation (suite)

Dans la suite du problème, nous allons considérer des graphes particuliers appelés *arbres*.

Un chemin $x_0, e_1, x_1, e_2, x_2, \ldots, e_k, x_k$ dans un graphe $G = (S, A)$ est dit *simple* si, pour tout i et tout j dans $\{1, \ldots, k\}$, $i \neq j$, $e_i \neq e_j$. Un *arbre* est un graphe tel qu'il n'existe qu'un seul chemin simple entre deux sommets quelconques.

Soit T un arbre et r un sommet de T appelé *racine*. L'arbre T de racine r peut se décomposer en *niveaux*. Le niveau 0 est composé de la racine r, le niveau 1 est composé de tous les voisins de la racine et, plus généralement, le niveau $i+1$ est composé de tous les voisins des sommets du niveau i n'apparaissant pas déjà au niveau $i-1$. Etant donné un sommet x de niveau i, le *père* de x est l'unique sommet du niveau $i-1$ voisin de x et les *fils* de x sont tous les sommets du niveau $i+1$ voisins de x. La racine n'a pas de père. Les sommets n'ayant pas de fils sont appelés *feuilles*. La *hauteur* d'un arbre T de racine r est la valeur du plus grand niveau.

Dans la suite du problème, on ne considérera (sauf lorsque ce sera explicitement spécifié) que des arbres de degré maximum borné par une constante Δ.

Nous allons représenter un arbre T de N sommets, de degré maximum Δ, et de racine r de la façon suivante :

```
cons Nmax = 1024 ; (* par exemple *)
     delta_max = 10 ;
type arbre = array [1..Nmax,0..delta_max] of integer ;
var T : arbre ;.
    r : integer ;
```

Pour une variable T de type arbre, de racine r, de degré maximun Δ et de N sommets :

- $T[r, 0] = 0$;
- pour tout i, $1 \leq i \leq N$, $i \neq r$, $T[i, 0]$ désigne le père du sommet i ;
- pour tout i, j, $1 \leq i \leq N$, $1 \leq j \leq \Delta$, $T[i, j]$ désigne le $j$ème fils du sommet i avec $T[i, j] = 0$ pour tout j strictement supérieur au nombre de fils de i ;

Partie 3. Plongements

Motivation : pouvoir concevoir des programmes sur un réseau d'ordinateurs en supposant qu'ils sont connectés selon une certaine forme (architecture) alors qu'ils le sont selon une autre.

Un *plongement* d'un graphe $G = (S(G), A(G))$ dans un graphe $H = (S(H), A(H))$ vérifiant $|S(G)| = |S(H)|$ est défini par la donnée de deux applications p et q satisfaisant :

- $p : S(G) \mapsto S(H)$ est une fonction bijective ;
- $q : A(G) \mapsto \{\text{chemins de } H\}$.

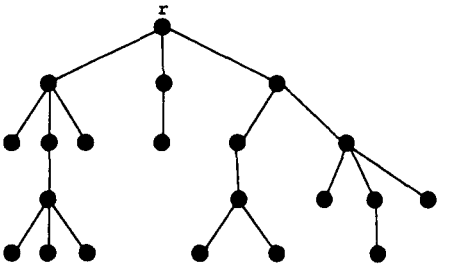
Dans cette partie, on s'intéresse aux plongements des *chaînes* dans les arbres. Une chaîne est un arbre dont tous les sommets ont un degré égal à 2 au plus (tous les sommets sont de degré 2 sauf les deux extrêmités de la chaîne). On supposera toujours les sommets de la chaîne numérotés de 1 à N de sorte qu'il y a une arête entre i et $i+1$ pour $i = 1, \ldots, N-1$.

Etant donné un arbre T de racine r et un sommet u de T, le sous-arbre de T de racine u est la partie de T formée de u, ses fils, ses petits-fils, etc.

Question 3.1.

Une numérotation en profondeur d'abord ν *des sommets d'un arbre T de racine r vérifie $\nu(r) = 1$ et, pour tout sommet u numéroté t, ses d fils $v_1, v_2, \ldots, v_d$ (s'ils existent) sont numérotés respectivement $t+1$, $t+k_1+1, \ldots, t+\sum_{i=1}^{d-1} k_i + 1$ où k_i est le nombre de sommets du sous-arbre de T de racine v_i.*

Donner une numérotation en profondeur d'abord de l'arbre T de racine r ci-dessous:



Question 3.2.

a) *Etant donné un plongement (p, q) d'un graphe G dans un graphe H, la* congestion *d'une arête e de H est définie comme le nombre de chemins de $Im(q)$ passant par e. La congestion d'un plongement est le maximum de la congestion de chacune des arêtes de H. Quelle est la congestion du plongement (p, q) d'une chaîne dans un arbre où p est définie par une numérotation des sommets de l'arbre en profondeur d'abord, et où $q(e)$, $e \in A(G)$, est défini comme l'unique chemin de l'arbre reliant les images des extrémités de e ?*

b) *Etant donné un plongement (p, q) d'un graphe G dans un graphe H, la* dilatation *d'une arête e de $A(G)$ est la longueur du chemin $q(e)$. La* dilatation *du plongement est le maximum de la dilatation des arêtes de G. Quelle est la dilatation du plongement (p, q) d'une chaîne dans un arbre où (p, q) est défini dans la question a ?*

Question 3.3.

Décrire un algorithme de numérotation des sommets d'un arbre en profondeur d'abord. Ecrire une procédure PASCAL

procedure numerote_en_profondeur (T : arbre ; N,r : integer ; var nu : array [1..Nmax] of integer) ;

qui numérote les sommets de l'arbre T d'ordre N de racine r en profondeur d'abord et range cette numérotation dans le tableau nu. Quel est son coût ?

Question 3.4.

Une étoile *est un arbre dont tous les sommets sauf un sont des feuilles.*

a) *Montrer que l'on peut plonger la chaîne de N sommets dans l'étoile de N sommets avec une congestion d'au plus 2 et une dilatation d'au plus 2.*

b) *Montrer que l'on peut plonger la chaîne de N sommets dans tout arbre de N sommets avec une congestion d'au plus 2 et une dilatation d'au plus 3.*

Partie 4. Diffusion dans un arbre

Motivation : on s'intéresse au temps nécessaire à l'opération de *diffusion*, c'est-à-dire l'envoi d'une donnée d'un sommet quelconque d'un réseau d'ordinateurs à tous les autres ordinateurs.

Le modèle de communication est le suivant :

1. une communication ne s'effectue qu'entre *deux* sommets *voisins* ;
2. un sommet ne peut participer qu'à au plus une communication à la fois ;
3. durant une communication, l'un des deux sommets impliqués peut envoyer à l'autre sommet autant d'informations qu'il le souhaite ;
4. plusieurs communications peuvent se dérouler simultanément ;
5. une communication prend un temps 1.

On note $b(G, u)$ le temps nécessaire à une diffusion à partir d'un sommet u d'un graphe G. On note $b(G)$ le temps de diffusion maximum dans G, c'est-à-dire $b(G) = \max_u b(G, u)$.

Question 4.1.

a) Soit T un arbre de racine r et de hauteur h tel que chaque sommet de niveau $< h$ ait exactement k fils. Montrer que $b(T, r) = kh$.

b) Montrer que $b(T, u) \geq \lceil \log_2 N \rceil$ pour tout sommet u d'un arbre T de N sommets.

c) Soit u un sommet d'un arbre T de racine r. Soit $v_1, v_2, \ldots, v_d$ les d fils de u. Pour pouvoir informer ces d fils au plus tard aux temps $t_1, t_2, \ldots, t_d$, respectivement, à quelle date u doit-il être informé au plus tard ? (On pourra supposer $t_1 \leq t_2 \leq \ldots \leq t_d$.)

Question 4.2.

a) Soit u un sommet d'un arbre T de racine r. On suppose que les temps auxquels doivent être informés au plus tard les fils de u sont rangés dans un tableau

```
type tableau = array [1..Delta_max] of integer ;
var tps_fils : tableau
```

tel que tps_fils[i] est le temps auquel doit être informé au plus tard le $i$ème fils de u. Ecrire une fonction PASCAL

```
function temps_pere (d : integer ; tps_fils : tableau) : integer ;
```

qui calcule le temps auquel le sommet u doit être informé au plus tard pour informer à temps ses d fils.

b)Décrire un algorithme qui calcule le temps $b(T, r)$ d'une diffusion à partir de r dans un arbre T. Ecrire une fonction PASCAL

```
function temps (T : arbre ; N,r : integer) : integer ;
```

qui calcule le temps $b(T, r)$ d'une diffusion à partir de r dans T d'ordre N.