

Devoir surveillé écrit -- Février 2009

Documents, ordinateurs, calculatrices, etc... INTERDITS.

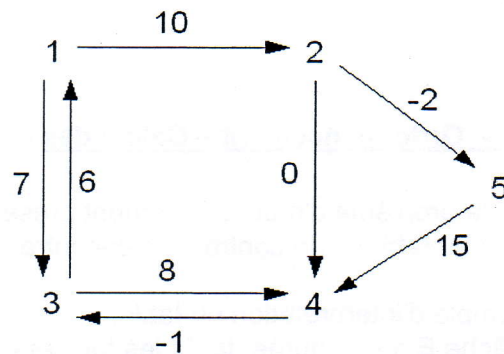
Pour la dernière question (QCM), vous devez répondre sur la feuille de l'énoncé et la joindre à votre devoir.

Q 1 - Représentation d'un graphe orienté en mémoire

Considérons les structures de données suivantes permettant de représenter un graphe orienté non valué :

```
typedef struct G1 {  
    bool M [ ] [ ] ;  
    // M [x] [y] = vrai      si l'arc (x,y) existe  
    // M [x] [y] = faux     sinon  
} G1 ;  
  
typedef struct G2 {  
    int M [ ] [ ] ;  
    // Pour chaque arc (x,y) du graphe, il existe une et une seule valeur 'a'  
    // telle que :  
    //      M [x] [a] = +1  
    //      M [y] [a] = -1  
    //      M [z] [a] = 0   pour tout 'z' différent de 'x' et 'y'  
} G2 ;
```

Considérons maintenant le graphe orienté valué ci-contre :



Q 1.1. Que faut-il modifier et/ou ajouter à la définition de G1 pour pouvoir représenter ce graphe ?

Q 1.2. Même question pour G2.

Pour les question 1.1 et 1.2, vous devez fournir la déclaration C des structures de données que vous proposez et expliquer son fonctionnement.

Q 1.3. Écrivez (en C) une fonction permettant de transférer le contenu de votre première structure de données (Q 1.1) vers la seconde (Q 1.2).

Q 2 - Tour de Hanoï

Le jeu de la Tour de Hanoï est décrit comme suit :

1. 3 tours A, B, C permettent d'empiler des disques les uns sur les autres ;
2. au départ, n disques sont empilés sur une seule tour ;
3. les disques sont de tailles différentes, allant du plus petit (1) jusqu'au plus grand (n) ;

V

4. sur une même tour, les disques ne peuvent être empilés de bas en haut que du plus grand au plus petit ;
 5. on ne peut déplacer qu'un disque à chaque opération.
- Le jeu consiste à déplacer tous les disques d'une tour sur une autre.

Q 2.1. Comment modéliser ce jeu à l'aide d'un graphe ?

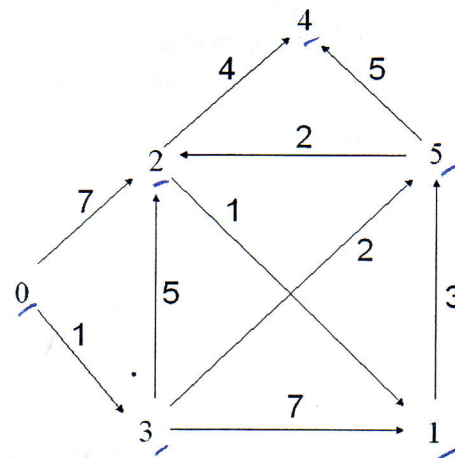
Q 2.2. Représentez graphiquement ce graphe dans le cas de 2 disques de tailles 1 et 2.

Q 2.3. Quel genre d'algorithme de la théorie des graphes utiliseriez-vous pour résoudre ce problème ?

Q 3 - Dijkstra

Déroulez l'algorithme de Dijkstra sur le graphe ci-contre.

Votre résultat doit être représenté sous la forme d'un tableau synthétique indiquant les valeurs des plus courts chemins allant du sommet '0' à tous les autres sommets.



Q 4 - Ordonnancement – Calcul des calendriers

Soit le problème d'ordonnancement présenté à l'aide du tableau de contraintes ci-contre :

Exemple d'interprétation du tableau :
la tâche E a une durée de 4 ; les tâches A et D doivent être terminées avant que E puisse commencer.

Q 4.1. Calculez le calendrier au plus tôt, en incluant les dates de début et de fin des travaux.

Q 4.2. Calculez le calendrier au plus tard en prenant une marge de 4 sur la date au plus tard de la fin des travaux par rapport à sa date au plus tôt.

Tâche	Durée	Contraintes
A	3	--
B	2	--
C	3	B
D	5	A
E	4	A D
F	2	A E
G	2	D
H	4	D E G
I	5	D
J	1	B I C
K	2	F H

Devoir surveillé écrit - 4 février 2009

Document à destination des étudiants, etc... INTERDITS.

Pour la partie 1.1 (GCM), vous devez répondre sur la feuille de l'exercice et la rendre avec vous.

1.1. Représentation d'un graphe orienté et pondéré

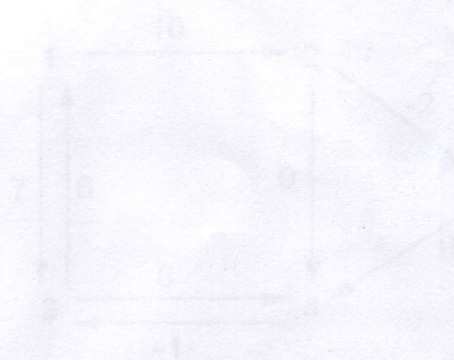
Considérons les structures de données suivantes permettant de représenter un graphe orienté pondéré.

```
graph = struct G;
graph.val = 1;
// si (x,y) = val / si lab(x,y) existe
// M(x,y) = lab(x,y) / sinon
```

```
G = struct G;
G.val = 1;
// Pour (x,y) et (x,y) du graphe, il existe une et une seule valeur
// lab(x,y)
// M(x,y) = val
// M(y,x) = -val
// M(x,x) = 0 pour tout x différent de y
```

1.2.

On se donne maintenant le graphe ci-dessous.



Q 1.1. Que faut-il modifier pour ajouter à la définition de G1 pour pouvoir représenter ce graphe ?

Q 1.2. Même question pour G2.

Pour les questions 1.1 et 1.2, vous devez fournir la description C des structures de données que vous proposez et expliquer son fonctionnement.

Q 1.3. Écrire (en C) une fonction permettant de transférer le contenu de votre première structure de données (Q 1.1) vers la seconde (Q 1.2).

2. Le Tour de Hanoi

Le jeu de 3 disques de Hanoi est décrit comme suit :

1. Trois disques A, B, C permettent l'empilage des disques les uns sur les autres.
2. Au départ, n disques sont empilés sur une seule tour.
3. Les disques sont de tailles différentes, allant du plus petit (1) jusqu'au plus grand (n).

Nom : KOMORNICKI Prénom : Rémi

QCM

Mettez une croix en face des réponses correctes.

Attention : Il y a toujours au moins une réponse correcte, mais pour certaines questions, il peut y en avoir plusieurs.

Notation : G est un graphe. S est l'ensemble des sommets du graphe. A est l'ensemble des arêtes ou arcs du graphe.

1 Quelle matrice peut-on utiliser pour le calcul du degré d'un sommet ?

- | | |
|-------------------------------------|------------------------------------|
| ☐ | - Matrice d'adjacence $S \times S$ |
| ☒ | - Matrice d'incidente $S \times A$ |
| ☐ | - Les deux |
| ☐ | - Aucune |

2 Un graphe non orienté est un couple $G=(S,A)$ où S définit un ensemble de sommets et A un ensemble de paires de sommets appelées arcs.

- | | |
|-------------------------------------|--------|
| ☐ | - Vrai |
| ☒ | - Faux |

3 Si x est l'extrémité initiale d'un arc dont y est l'extrémité finale, alors y est le :

- | | |
|-------------------------------------|-----------------------|
| ☐ | - prédécesseur de x |
| ☒ | - successeur de x |

4 Un chemin élémentaire peut passer plusieurs fois par le même arc

- | | |
|-------------------------------------|--------|
| ☐ | - Vrai |
| ☒ | - Faux |

5 Soit un graphe défini par S l'ensemble de ses sommets et A l'ensemble de ses arcs. Une matrice d'adjacence représente une fonction définie entre les ensembles

- | | |
|-------------------------------------|--------------|
| ☐ | - S et A |
| ☒ | - S et S |
| ☐ | - A et A |

6 Dans un arbre, il peut exister plusieurs chaînes entre deux sommets donnés.

- | | |
|-------------------------------------|---------------------------------------|
| ☐ | - Vrai |
| ☒ | - Faux |
| ☐ | - Dans les arbres binaires uniquement |

7 Un graphe non orienté est simple si et seulement si il est sans boucle

- | | |
|-------------------------------------|--------|
| ☐ | - Vrai |
| ☒ | - Faux |

Nom : KOMORNIKI Prénom : Rémi

8 Un point d'entrée peut appartenir à un circuit si un nombre impair d'arcs lui sont incidents

- ☐ - Vrai
☒ - Faux

9 Un graphe $G=(S,A)$ est d'ordre N si

- ☐ $|S| < N$
☐ $|S| \leq N$
☒ $|S| = N$
☐ $|S| > N$
☐ $|S| \geq N$

($|S|$ est le nombre d'éléments dans S)

10 $G'=(S',A')$ est le graphe partiel de $G=(S,A)$ engendré par A' si :

- ☐ $S' \subset S$
☐ $S' \subseteq S$
☒ $A' \subset A$
☐ $A' \subseteq A$

11 Une matrice d'incidence d'un graphe $G=(S,A)$ représente une fonction définie entre les ensembles :

- ☐ S et S
☒ S et A
☐ A et A

12 Dans un graphe valué, un circuit absorbant est un circuit de coût :

- ☒ Négatif
☐ Nul
☐ négatif ou nul

13 Un graphe orienté est dit *complet* si et seulement si pour tout couple de sommets (x, y) , il existe un chemin de x à y .

- ☐ Vrai
☒ Faux

14 Une matrice Sommets x Arcs est composée d'éléments dont les valeurs peuvent être :

- ☒ 0
☒ -1
☒ 1
☐ 'vrai' ou 'faux'

15 Un graphe est dit symétrique si et seulement si :

- ☒ pour tout arc $x \rightarrow y$, il existe un arc $y \rightarrow x$
☐ pour tout chemin de x à y , il existe un chemin de y à x

Nom : KOMORNICKI Prénom : Rémi

16 Si on note $CFC(a)$ la composante fortement connexe contenant le sommet a dans un graphe, alors pour tout sommet x appartenant à $CFC(a)$, on a $CFC(x) \subset CFC(a)$

- ☐ $CFC(x) \subset CFC(a)$
☒ $CFC(x) \subseteq CFC(a)$
☐ $CFC(x)$ et $CFC(a)$ peuvent être disjointes

17 Si un graphe est sans circuit, alors les valeurs de la diagonale de sa matrice d'adjacence sont :

- ☐ des 'vrai'
☒ des 'faux'
☐ des 'vrai' ou des 'faux'

18 Pour pouvoir exécuter l'algorithme de Dijkstra, il faut que :

- ☐ Le graphe ne comporte pas de boucle
☒ Aucun arc n'a de valeur négative
☒ Il n'y a pas de circuit absorbant
☐ ...il n'y a aucune contrainte à l'exécution de l'algorithme de Dijkstra !

19 Pour pouvoir exécuter l'algorithme de Bellman, il faut que :

- ☐ Le graphe ne comporte pas de boucle
☐ Aucun arc n'a de valeur négative
☒ Il n'y a pas de circuit absorbant
☐ ...il n'y a aucune contrainte à l'exécution de l'algorithme de Bellman !

20 Pour calculer un calendrier au plus tard sur un graphe d'ordonnancement, il faut que :

- ☐ Le graphe ne comporte pas d'arc à valeur nulle
☒ La date au plus tard de la fin de projet soit fixée
☐ La date au plus tôt du début de projet soit strictement positive.

fin du devoir

DE - Th orie des Graphes

DE 18,1
+ TP 1,2

Q1.

• Q1.1.

Consid rant le graphe valu  donn , n'ayant pas 2 arcs ayant les m mes extr mit s initiales et finales, il est bel et bien possible de repr senter gr ce   la structure G1.

Cela peut se faire en ~~don~~ rempla ant les  l ments bool ens du tableau (ou matrice) par des entiers de la valeur de l'arc.

→ Cependant, la pr sence de valeurs nulles complique la t che, car il faut diff rencier un arc de valeur 0 et un arc non-existant.

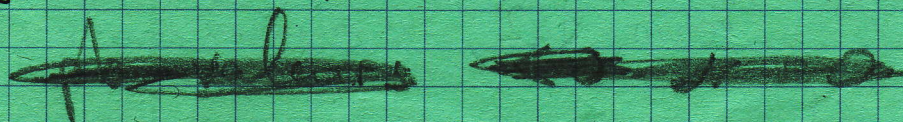
```
typedef struct G1 {  
    bool M[ ][ ]; // m me tableau : permet de conna tre l'existence.  
    int val[ ][ ]; // nouveau tableau : pour conna tre les valeurs.  
    // val[x][y] = v si arc (x,y) existe et sa valeur est v.  
    // val[x][y] = 0 sinon.  
} G1;
```

• Q.1.2.

Pour G2, ce m me principe est utilisable, mais en donnant comme valeurs $-v$ aux arcs vers l'int rieur.

```
typedef struct G2 {  
    int M[ ][ ]; // m me tableau, pour l'existence.  
    int val[ ][ ]; // nouveau tableau pour les valeurs.  
    // val[x][a] = +v si arc (x,y) existe et a pour valeur v.  
    // val[y][a] = -v  
    // val[z][a] = 0 pour tout 'z'   'x' et 'y'.  
} G2;
```

il y a
plus
simple



• Q. 1.3.

G2* transfert (G1* g1, {int N, int A}) { // N: # de sommets, A: # d'arcs.
 int l;
 int i, j, k=0; g2.val =
 G2 g2; g2.M = malloc (N * sizeof (int)); // allocation lignes de G2 (s

~~for (i=0; i<N; i++)
 {
 for (j=0; j<N; j++)
 {
 if (G1->M[i][j] != 0)
 {
 g2.M[i] = malloc (A * sizeof (int));~~

for (i=0; i<N; i++) {
 g2.val[i] = g2.M[i] = malloc (A * sizeof (int)); } // reservation colonnes (arcs)
 for (i=0; i<N; i++) // boucle sommets de départ initiaux
 { for (j=0; j<N; j++) // boucle sommets terminaux

{
 if (g1->M[i][j] != 0) // si arc existe de i à j,

{
 g2.M[i][k] = 1; // k représente le numéro de l'arc
 g2.val[i][k] = g1->val[i][j];
 g2.M[j][k] = -1;
 g2.val[j][k] = -(g1->val[i][j]);

for (l=0; l<N; l++) // remplissage des 0.

{
 if (l != i && l != j)

{
 g2.M[l][k] = 0;
 g2.val[l][k] = 0;

k++ // on passe à l'arc suivant.

return &g2; // on retourne l'adresse de la matrice g2.

}

Q2.

• Q2.1.

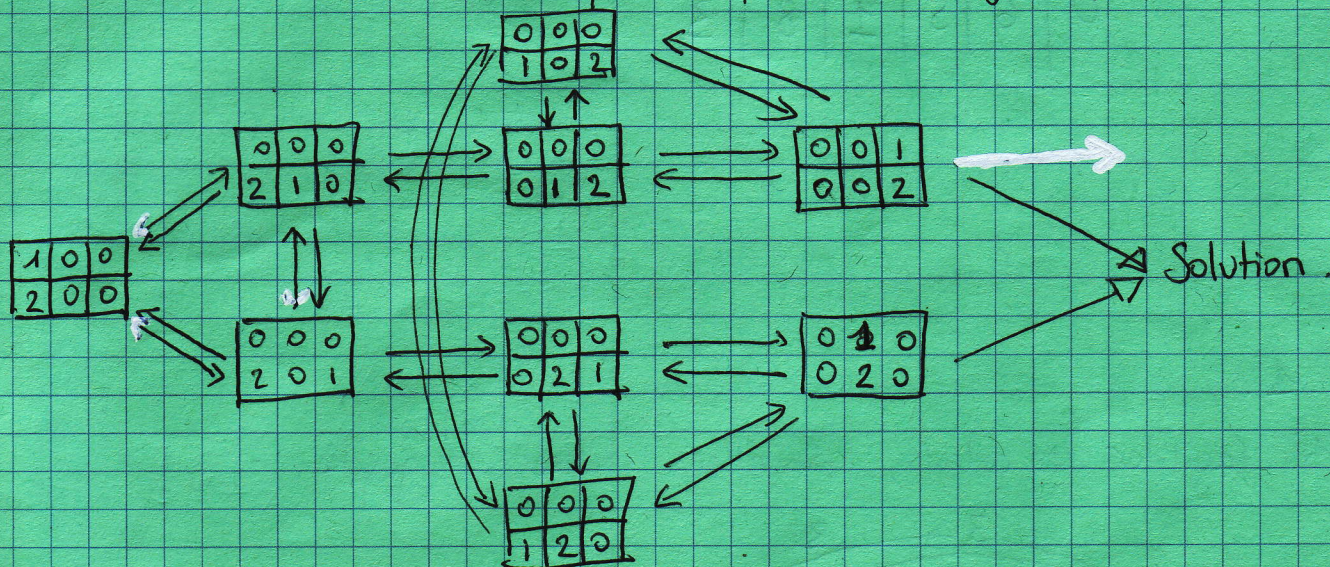
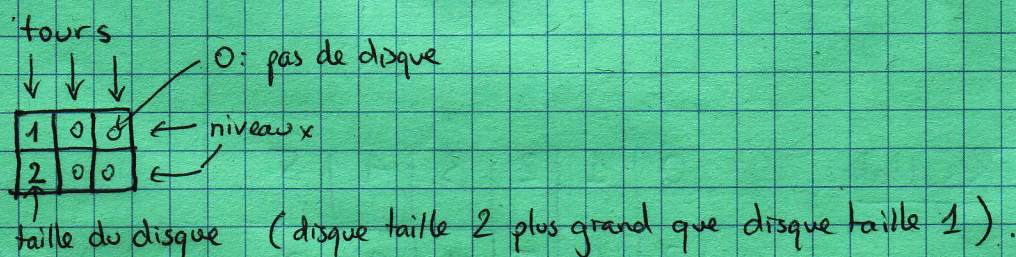
mmets)

Pour modéliser ce jeu à l'aide d'un graphe, il faut représenter sur chaque sommet une configuration (placement des disques qui respecte les règles) possible. Chaque arc est une transition possible d'un disque, qui permet d'aller d'une configuration x à une configuration y en déplaçant 1 seul disque selon les règles.

Si une configuration est une solution, elle pointe vers une fin.

• Q2.2.

Représentation:



• Q2.3.

Il faudrait, pour résoudre ce problème, définir le chemin le plus court.

Sachant que ~~chaque arc~~ les arcs ne sont pas valués, on peut utiliser l'algorithme des chemins les plus courts pour graphes non-valués (oubli du nom de l'algorithme!).

c'est bien dommage !...

donc valeur = 1

$k \backslash$	5	0	1	2	3	4	5
(Iterations)	0	0,0 _{cc}	$+\infty$ _m	7,0 _m	1,0 _m	$+\infty$ _m	$+\infty$ _m
1	/	8,3 _m	6,3 _m	1,0 _{cc}	$+\infty$ _m	3,3 _m	
2	/	8,3 _m	5,5 _m	/	8,5 _m	3,3 _{cc}	
3	/	6,2 _m	5,5 _{cc}	/	8,5 _m	/	
4	/	6,2 _{cc}	/	/	8,5 _m	/	
5	/	/	/	/	8,5 _{cc}	/	

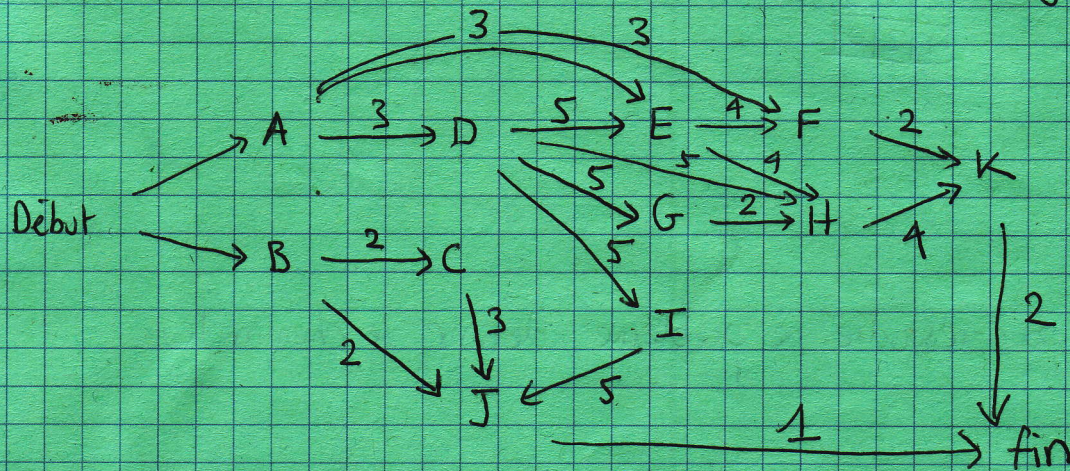
Résultat :

	0	1	2	3	4	5
	0	6	5	1	8	3

Q 4.

Q 4.1.

Pour faciliter les calculs, nous allons utiliser un graphe :

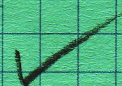


DE : Théorie des Graphes (2)

(suite) Q4.1.

Résultat:

Début	~~X~~	A	B	C	D	E	F	G	H	I	J	K	fin
Dates + tôt	~~D~~	0	0	2	3	8	12	8	12	8	13	16	18



• Q4.2.

$$T_w = z_w + 4 = 22.$$

Ainsi :

$$T_k = \min(T_w) - d_k = 20$$

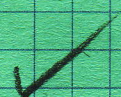
$$T_j = \min(T_w) - d_j = 21.$$

$$T_H = \min(\underbrace{T_w, T_k}_{\Gamma(H)}) - d_H = 16$$

etc.

Résultat:

Début	~~X~~	A	B	C	D	E	F	G	H	I	J	K	fin
Dates + tard	~~D~~	4	16	18	7	12	18	14	16	16	21	20	22



→ VOIR QCM.