

DE - Théorie des Graphes

DE 18, 1
+ TP 1, 2

Q1.

• Q1.1.

Considérant le graphe valué donné, n'ayant pas 2 arcs ayant les mêmes extrémités initiales et finales, il est bel et bien possible de représenter grâce à la structure G1.

Cela peut se faire en ~~donc~~ remplaçant les éléments booléens du tableau (ou matrice) par des entiers de la valeur de l'arc.

→ Cependant, la présence de valeurs nulles complique la tâche, car il faut différencier un arc de valeur 0 et un arc non-existant.

```
typedef struct G1 {  
    bool M[ ][ ]; // même tableau : permet de connaître l'existence.  
    int val[ ][ ]; // nouveau tableau : pour connaître 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 même principe est utilisable, mais en donnant comme valeurs $-v$ aux arcs vers l'intérieur.

```
typedef struct G2 {  
    int M[ ][ ]; // même 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 ( 1 * 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.

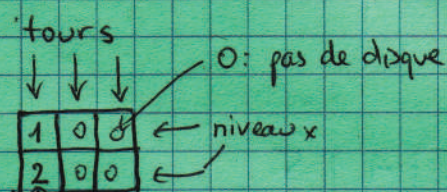
mmets)

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.

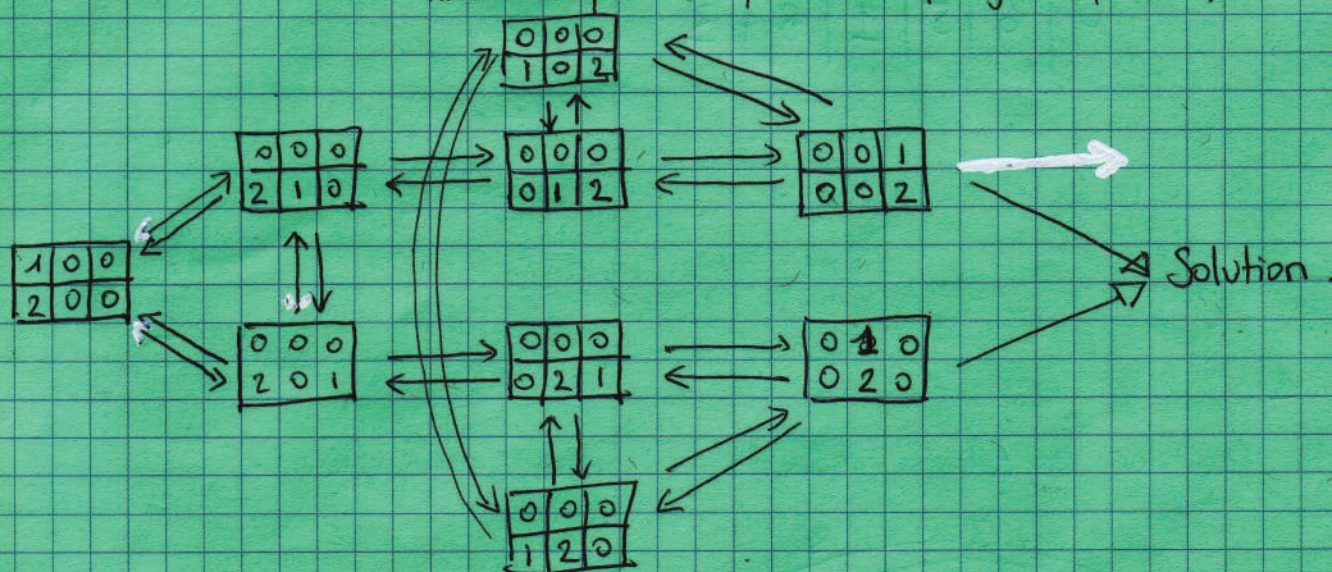
Si une configuration est une solution, elle pointe vers une fin.

• Q2.2.

Représentation:



taille du disque (disque taille 2 plus grand que disque taille 1).



• 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 \ s	0	1	2	3	4	5
(iterations) 0	0,0 _{cc}	+∞ _m	7,0 _m	1,0 _m	+∞ _m	+∞ _m
1	/	8,3 _m	6,3 _m	1,0 _{cc}	+∞ _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}	/

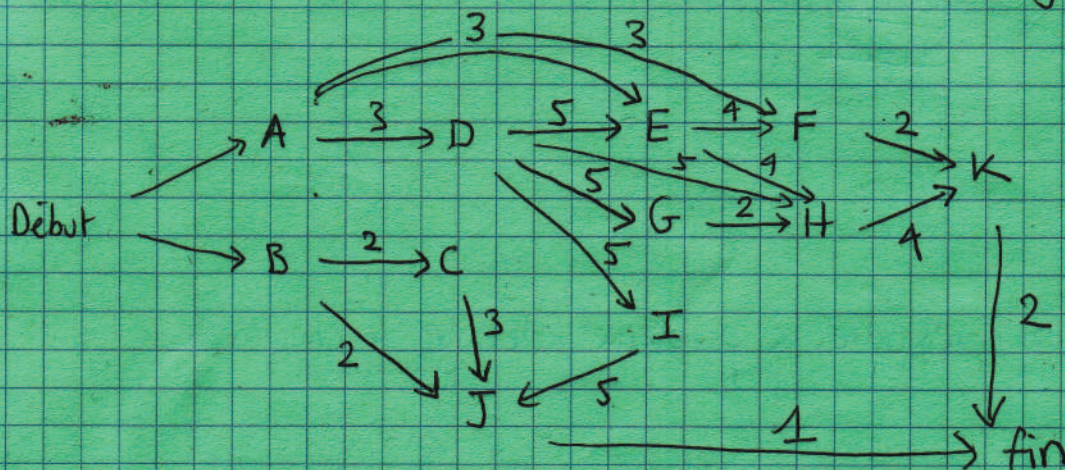
Résultat :

	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 :

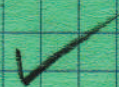


DE : Théorie des Graphes (2)

(suite) Q4.1.

Résultat:

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



• Q4.2.

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

Ainsi :

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

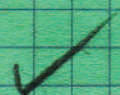
$$T_j = \min(T_w) - d_j = 21.$$

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

etc.

Résultat:

Début	~~X~~	A	B	C	D	E	F	G	H	I	J	K	fin
Dates + tard	0	4	16	18	7	12	18	14	16	16	21	20	22



→ VOIR QCM.