

I21: Introduction à l'algorithmique

Cours 8: Parcours dans une grille

Nicolas Méloni

Licence 1: 2ème semestre
(2017/2018)

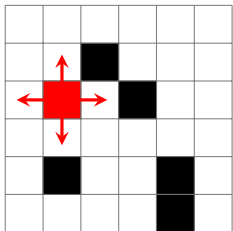
- ❑ On représente un espace en deux dimensions par une grille de cases rectangulaires (comme un plateau d'échecs) ;
- ❑ on modélise une telle grille à l'aide d'un tableau d'entiers à deux dimensions ;
- ❑ les cases contenant le nombre 0 seront considérées comme libres ;
- ❑ les cases contenant le nombre 1 seront considérées comme inaccessibles ;
- ❑ on autorise seulement les déplacements orthogonaux.

```
[[0,0,0,0,0,0,0,0],  
 [0,0,1,0,0,0,0,0],  
 [0,0,0,1,0,0,0,0],  
 [0,0,0,0,0,0,0,0],  
 [0,1,0,0,0,0,1,0],  
 [0,0,0,0,0,0,1,0]]
```

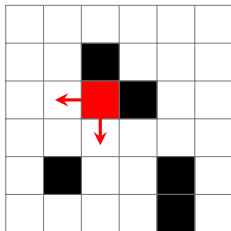


	0	1	2	3	4	5
0						
1			■			
2				■		
3						
4		■			■	
5					■	

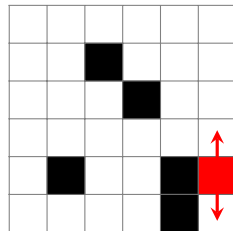
- On se donne fonction de déplacement retournant une file contenant les cases atteignables à partir d'une case donnée.



(2,0) (1,1) (2,3) (4,1)



(2,1) (3,2)



(3,5) (5,5)

```
1  ALGORITHME Voisins(G, lig , col ):
2  DONNEES
3    G: tableau de taille n x m
4    lig , col: entiers
5  VARIABLES:
6    voisins: File
7    dep: tableau de taille 4 x 2
8    i,l,c: entiers
9  DEBUT
10   dep ← [[1,0],[0,1],[-1,0],[0,-1]]
11   i ← 1
12   TQ i ≤ 4 FAIRE
13     l ← lig+dep[i][0]
14     c ← col+dep[i][1]
15     SI 1≤l≤n ET 1≤c≤m ET G[l][c]=0 ALORS
16       enfiler(voisins , [l,c])
17     FSI
18     i ← i+1
19   FTQ
20   RENVoyer voisins
21  FIN
```



Complexité : $\Theta(1)$

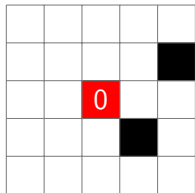
On s'intéresse à deux problèmes spécifiques :

- le calcul du voisinage ;
- la recherche de chemin.

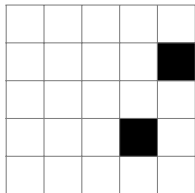
Problème : Calcul de voisinage

Entrée : Une grille 2D G de taille $n \times m$, les coordonnées d'une case de la grille (lig, col) et un entier k .

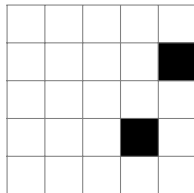
Sortie : L'ensemble des cases atteignables en partant de (lig, col) en k déplacements.



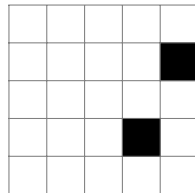
$k = 0$



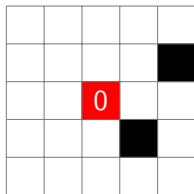
$k = 1$



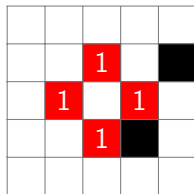
$k = 2$



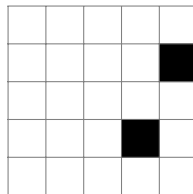
$k = 3$



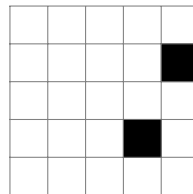
$k = 0$



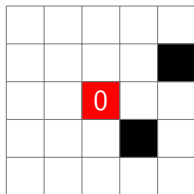
$k = 1$



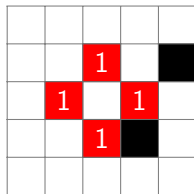
$k = 2$



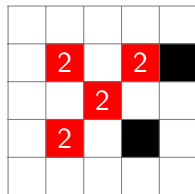
$k = 3$



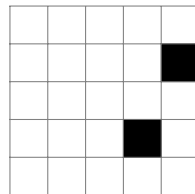
$k = 0$



$k = 1$

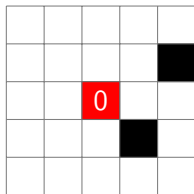


$k = 2$

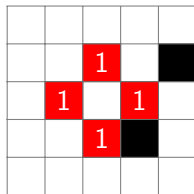


$k = 3$

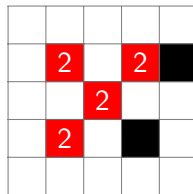
Calcul de voisinage



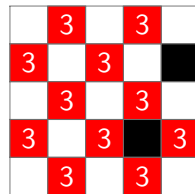
$k = 0$



$k = 1$



$k = 2$



$k = 3$

Calcul de voisinage

```
1  ALGORITHME Voisinage(G, lig , col , k):  
2  DONNEES  
3    G: tableau de taille n x m  
4    lig , col: entiers  
5  VARIABLES:  
6    g: tableau de taille n x m  
7    F, vois , kvois: File  
8    L, C, l, c: entiers  
9  DEBUT  
10   g[lig][col] ← 0  
11   enfiler(F, [lig , col])  
12   TQ file_vider(F)=FAUX FAIRE  
13     L, C ← defiler(F)  
14     vois ← Voisins(G, L, C)  
15     TQ file_vider(vois)=FAUX FAIRE  
16       l, c ← defiler(vois)  
17       SI g[l][c] < g[L][C]+1 ALORS  
18         g[l][c] ← g[L][C]+1  
19         SI g[l][C]=k ALORS  
20           enfiler(kvois , [l , c])  
21         SINON  
22           enfiler(F, [l , c])  
23       FSI  
24     FSI  
25   FTQ  
26   REVOYER kvois  
27 FIN
```

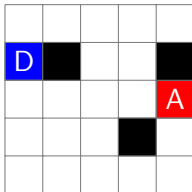
Problème : Recherche de chemin

Entrée : Une grille 2D G de taille $n \times m$, les coordonnées d'une case de départ (ld, cd) et d'une case d'arrivée (la, ca) .

Sortie : Un chemin allant de la case de départ à la case d'arrivée.

Idée générale :

- Calculer la distance de chaque case de la grille à la case de départ ;
- une fois la case d'arrivée atteinte, remonter jusqu'à la case de départ en se déplaçant à chaque fois sur une case de distance inférieure de 1 ;







1	2	3		
D				
1	2	3		A
2	3			
3				

Recherche de chemin

1	2	3	4	
D		4		
1	2	3	4	A
2	3	4		
3	4			

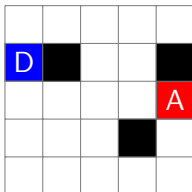
1	2	3	4	5
D		4	5	
1	2	3	4	A
2	3	4		
3	4	5		

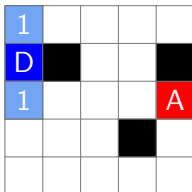
1	2	3	4	5
D		4	5	
1	2	3	4	A
2	3	4		
3	4	5		

- Chemin : (0,1), (0,2), (1,2), (2,2), (3,2), (4,2)

```
1 ALGORITHME Distance(G,ld,cd):
2 DONNEES
3   G: tableau de taille n x m
4   ld,cd,la,ca: entiers
5 VARIABLES:
6   F,vois: Files
7   g: tableau de taille n x m initialise a nm
8   L,C,l,c: entiers
9 DEBUT
10  enfiler(F,[ld,cd])
11  TQ file_vider(F)=FAUX FAIRE
12    L,C ← defiler(F)
13    vois ← Voisins(G,L,C)
14    TQ file_vider(vois)=FAUX FAIRE
15      l,c ← defiler(vois)
16      SI g[L][C]+1 < g[l][c] ALORS
17        g[l][c] ← g[L][C]+1
18        enfiler(F,[l,c])
19      FSI
20    FTQ
21  FTQ
22  RENDRE g
23 FIN
```

```
1  ALGORITHME Chemin(G,ld ,cd ,la ,ca ):
2  DONNEES
3    G: tableau de taille n x m
4    ld ,cd ,la ,ca: entiers
5  VARIABLES:
6    P: Pile
7    vois: File
8    g: tableau de taille n x m initialise a nm
9    L,C,l,c: entiers
10 DEBUT
11   g ← Distance(G,ld ,lc)
12   L,C ← la ,ca
13   empiler(P,[L,C])
14   TQ g[L][C]>0 FAIRE:
15     vois ← Voisins(G,L,C)
16     l,c ← defiler(vois)
17     TQ g[l][c]≠g[L][C]-1 FAIRE
18       l,c ← defiler(vois)
19     FTQ
20     empiler(P,[l,c])
21     L,C ← l,c
22   FTQ
23   RENDRE P
24 FIN
```







1	2	3		
D				
1				A

1	2	3	4	
D		4		
1				A

1	2	3	4	5
D		4	5	
1				A

1	2	3	4	5
D		4	5	
1			6	A

1	2	3	4	5
D		4	5	
1			6	A



- Chemin : $(0,1)$, $(0,0)$,
 $(1,0)$, $(2,0)$, $(3,0)$,
 $(3,1)$, $(3,2)$, $(4,2)$