I21: Introduction à l'algorithmique

Cours 8: Parcours dans une grille

Nicolas Méloni

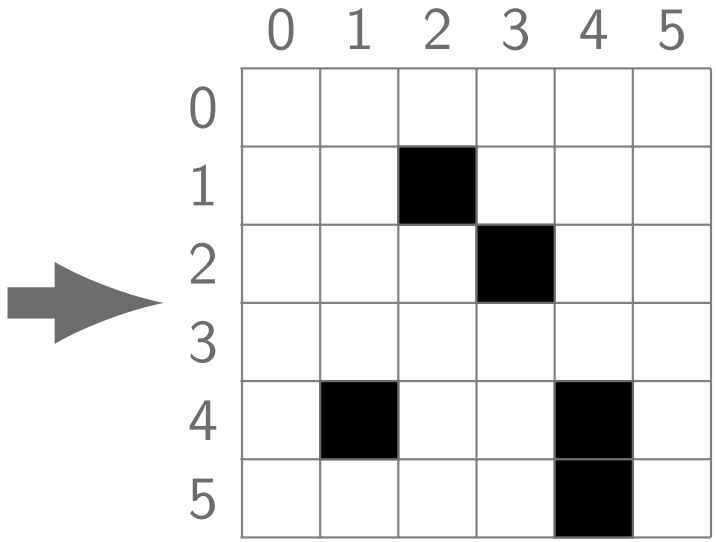
Licence 1 (2017-2020)

Modélisation d'un espace 2D

- 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 inaccessbiles ;
- on autorise seulement les déplacement orthogonaux.

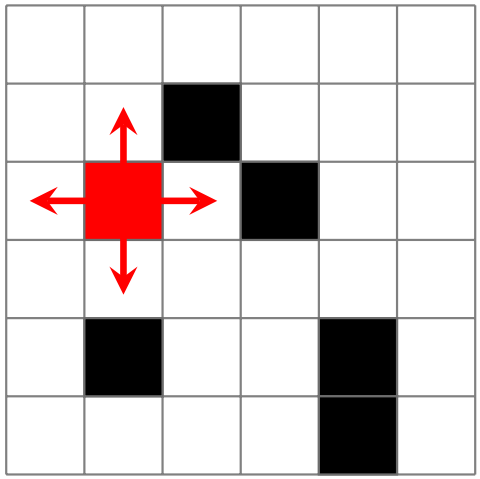
Modélisation d'un espace 2D

```
[[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]]
```

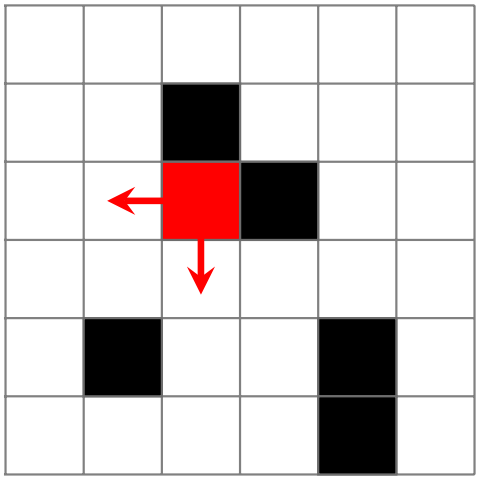


Modélisation d'un espace 2D

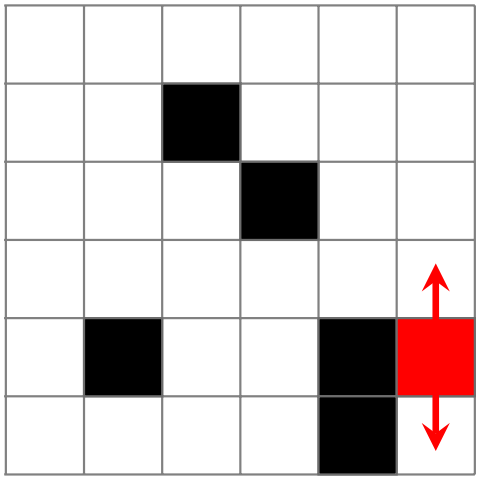
- On se donne fonction de déplacement retournant une file contenant les cases atteignables à partir d'une case donnée.



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



(2,1) (3,2)



(3,5) (5,5)

Modélisation d'un espace 2D

```
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 : $\Theta(1)$

Modélisation d'un espace 2D

On s'intéresse à deux problèmes spécifiques :

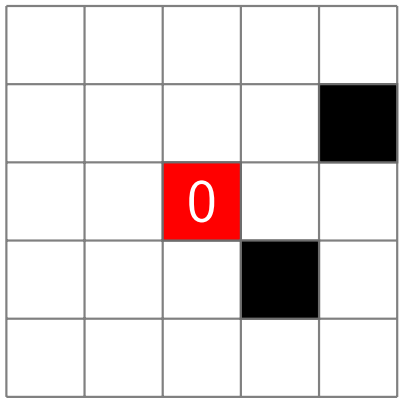
- le calcul du voisinage ;
- la recherche de chemin.

Calcul de voisinage

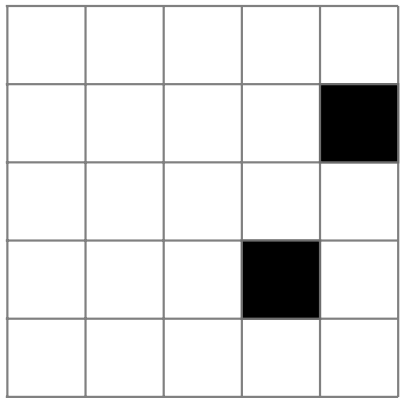
Problème : Calcul de voisinage

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.

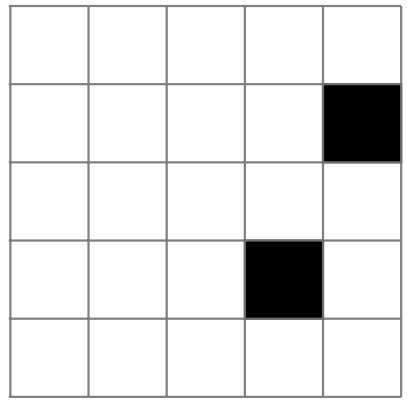
Calcul de voisinage



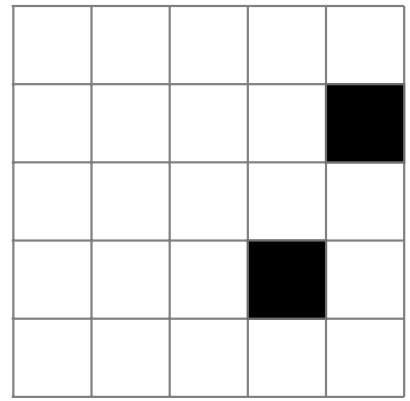
$k = 0$



$k = 1$

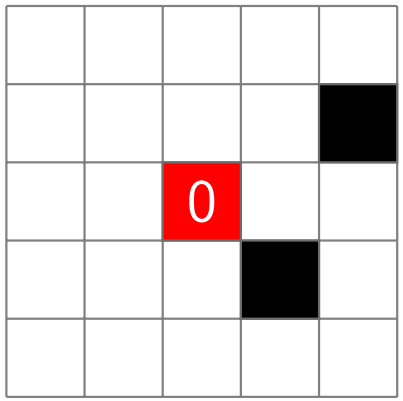


$k = 2$

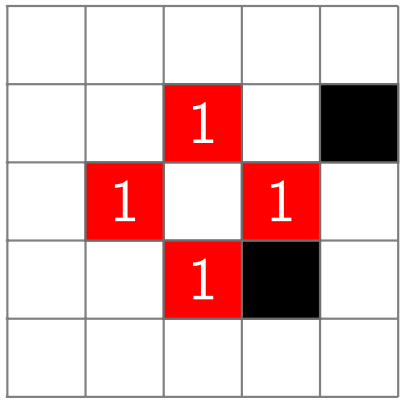


$k = 3$

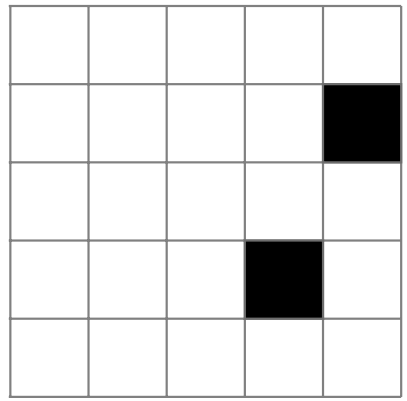
Calcul de voisinage



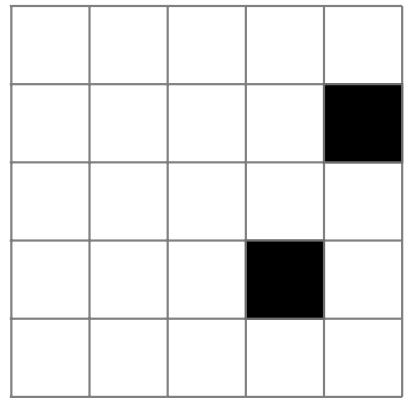
$k = 0$



$k = 1$

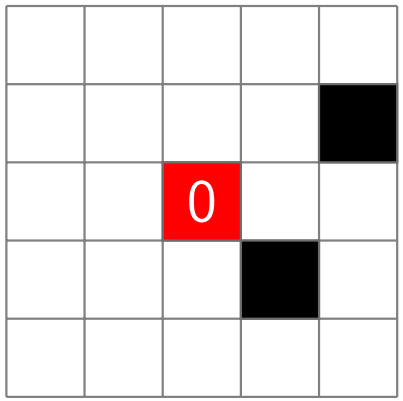


$k = 2$

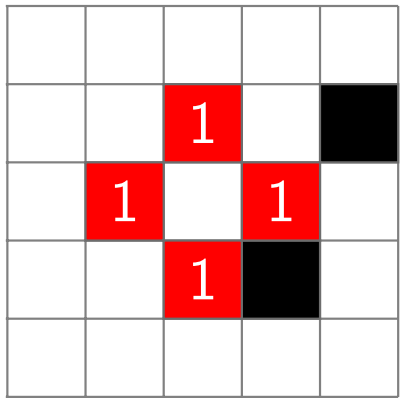


$k = 3$

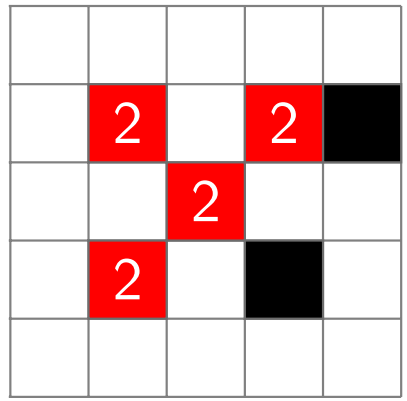
Calcul de voisinage



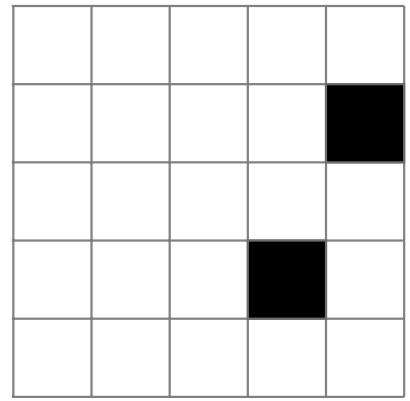
$k = 0$



$k = 1$

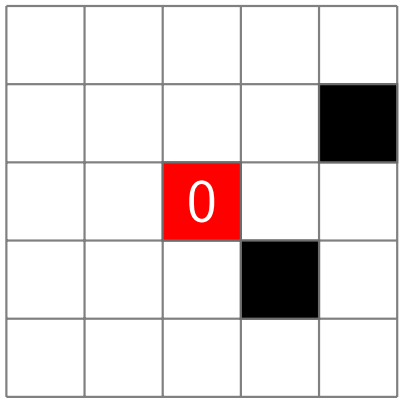


$k = 2$

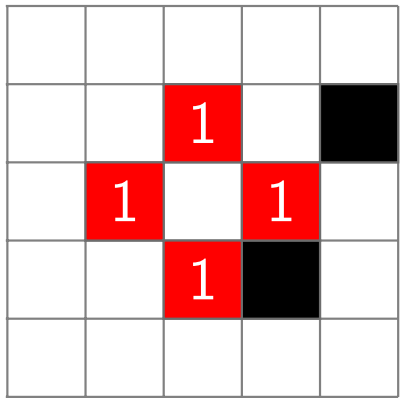


$k = 3$

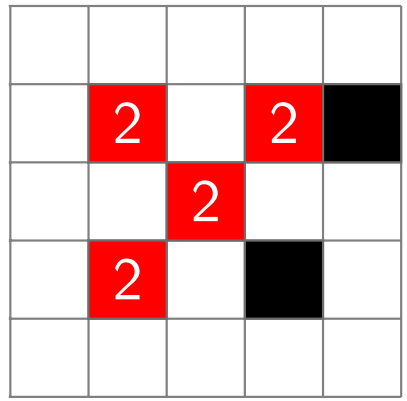
Calcul de voisinage



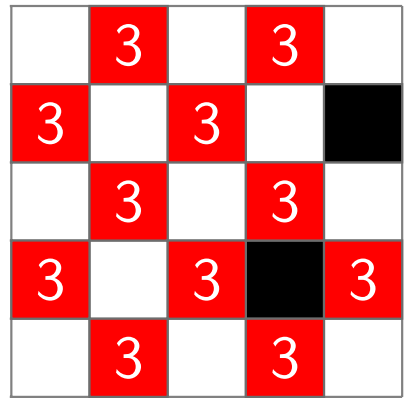
$k = 0$



$k = 1$



$k = 2$



$k = 3$

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_vide(F)=FAUX FAIRE
13      L,C ← defiler(F)
14      vois ← Voisins(G,L,C)
15      TQ file_vide(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    FTQ
27    RENVOYER kvois
28  FIN
```

Recherche de chemin

Problème : Recherche de chemin

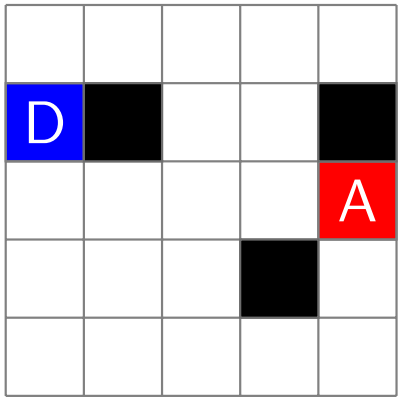
Entrée : Une grille 2D G de taille $n \times m$, les coordonnées d'une case de départ (ld, cd) et d'une case d'arrivée (la, ca).
Sortie : Un chemin allant de la case de départ à la case d'arrivée.

Recherche de chemin

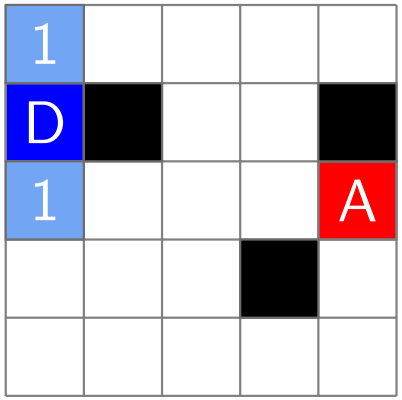
Idée générale :

- 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 ;

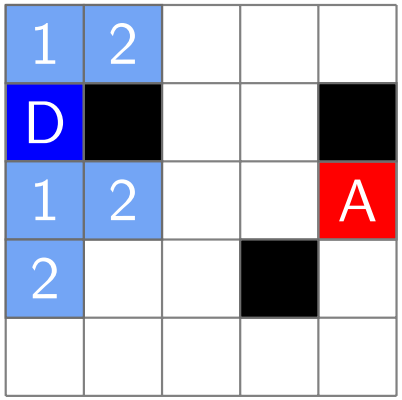
Recherche de chemin



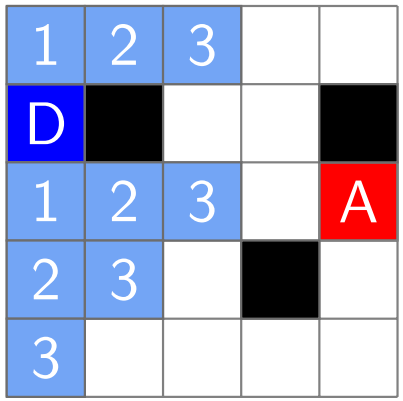
Recherche de chemin



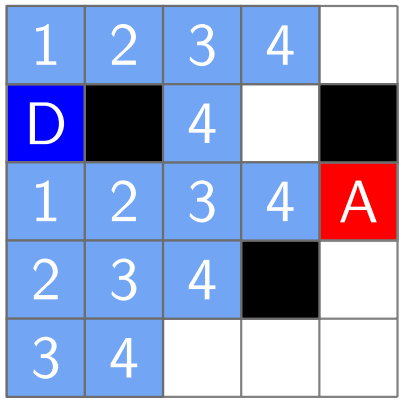
Recherche de chemin



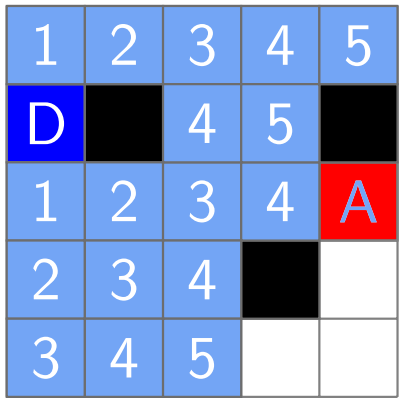
Recherche de chemin



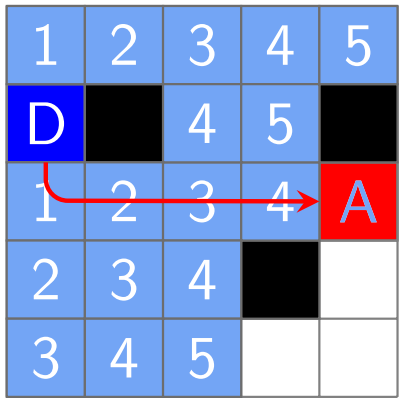
Recherche de chemin



Recherche de chemin



Recherche de chemin



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

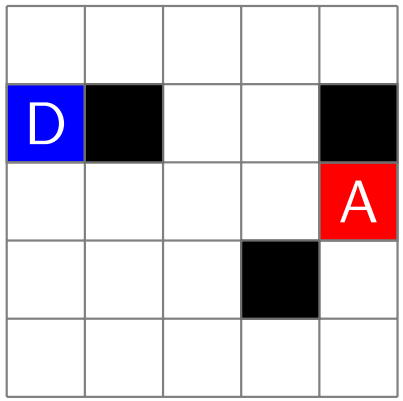
Recherche de chemin

```
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 initiallise a nm
8    L,C,l,c: entiers
9  DEBUT
10   enfiler(F,[ld,cd])
11   TQ file_vide(F)=FAUX FAIRE
12     L,C ← defiler(F)
13     vois ← Voisins(G,L,C)
14     TQ file_vide(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   RENVOYER g
23 FIN
```

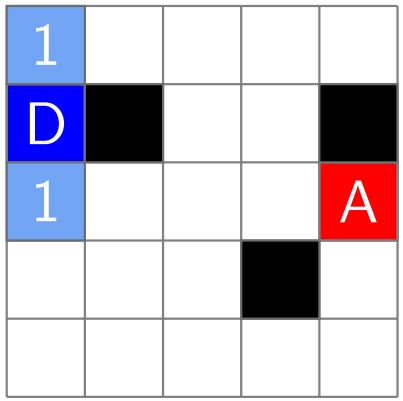
Recherche de chemin

```
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 initiallise 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   RENVOYER P
24 FIN
```

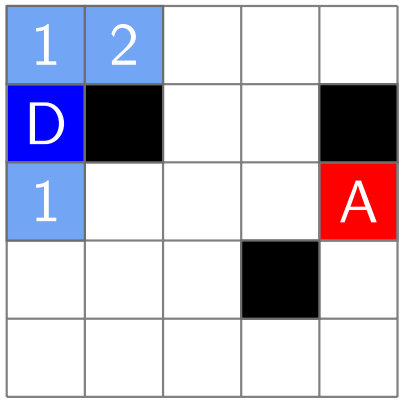
Recherche de chemin en profondeur



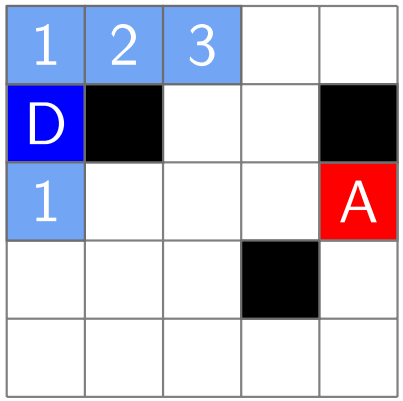
Recherche de chemin en profondeur



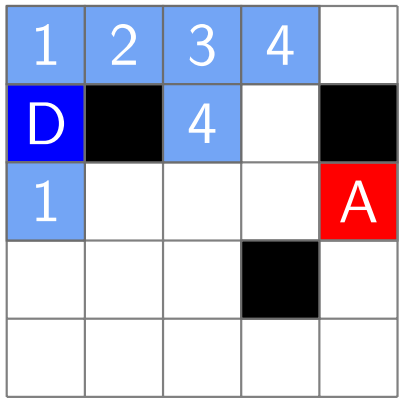
Recherche de chemin en profondeur



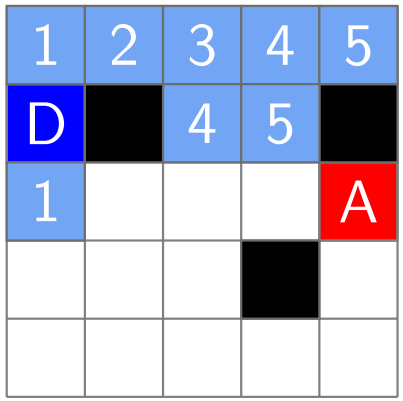
Recherche de chemin en profondeur



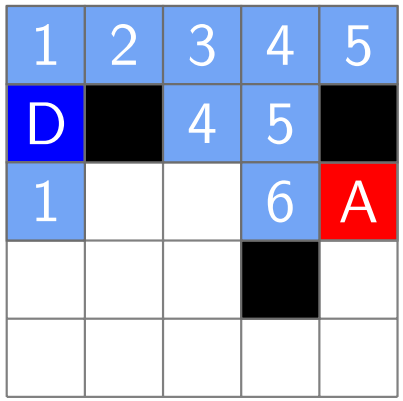
Recherche de chemin en profondeur



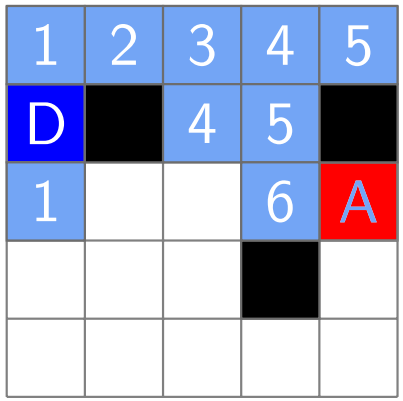
Recherche de chemin en profondeur



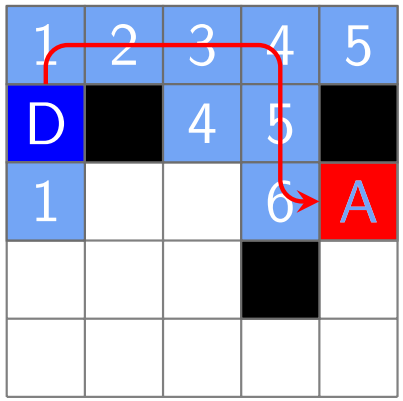
Recherche de chemin en profondeur



Recherche de chemin en profondeur



Recherche de chemin en profondeur



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