Recherche de chemin et labyrinthe

(Starcraft 2)

vincent.thomas@loria.fr

Atelier ISN 2019 – 7 Mars 2019

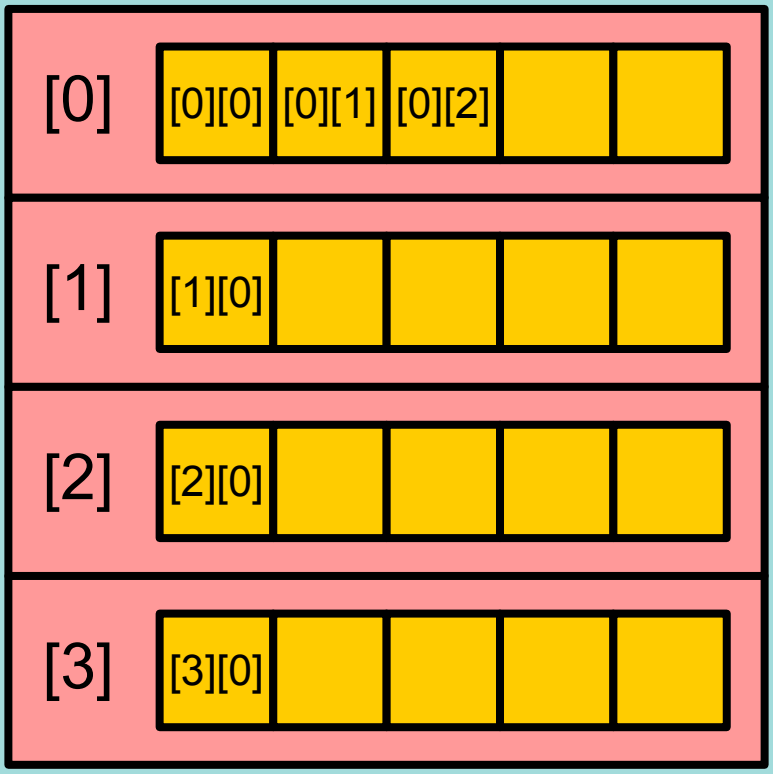
Plan

- Structure de labyrinthe
- Recherche de chemin
  - Algorithme de Lee / la vague
- Extensions
  - Dijkstra
  - A *
  - Autres problèmes

Structure de labyrinthe

- Cf Code python fourni
  - Tableau à deux dimensions

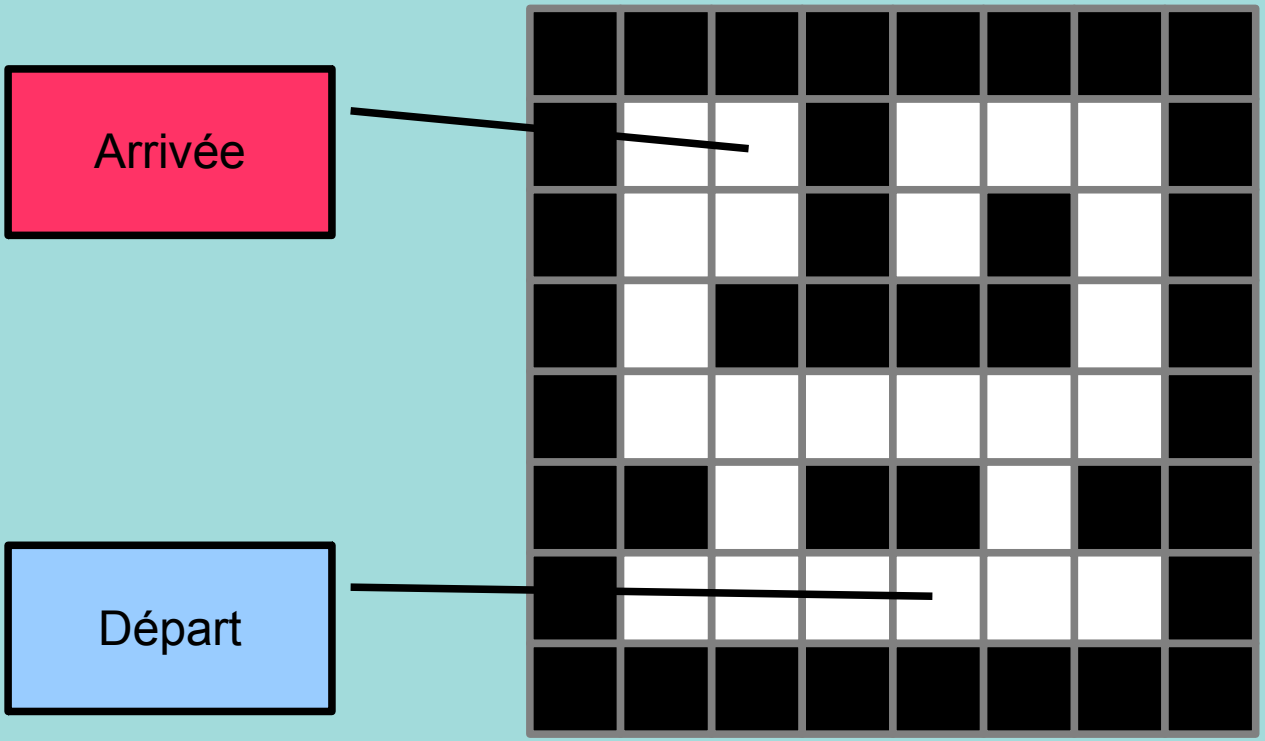
```
laby = []
laby +=[[1,1,1,1,1,1,1,1]]
laby +=[[1,0,0,1,0,0,0,1]]
laby +=[[1,0,0,1,0,1,0,1]]
laby +=[[1,0,1,1,1,1,0,1]]
laby +=[[1,0,0,0,0,0,0,1]]
laby +=[[1,1,0,1,1,0,1,1]]
laby +=[[1,0,0,0,0,0,0,1]]
Laby +=[[1,1,1,1,1,1,1,1]]
transpose(laby)
----------------------------
laby[x][y]
```



Plan

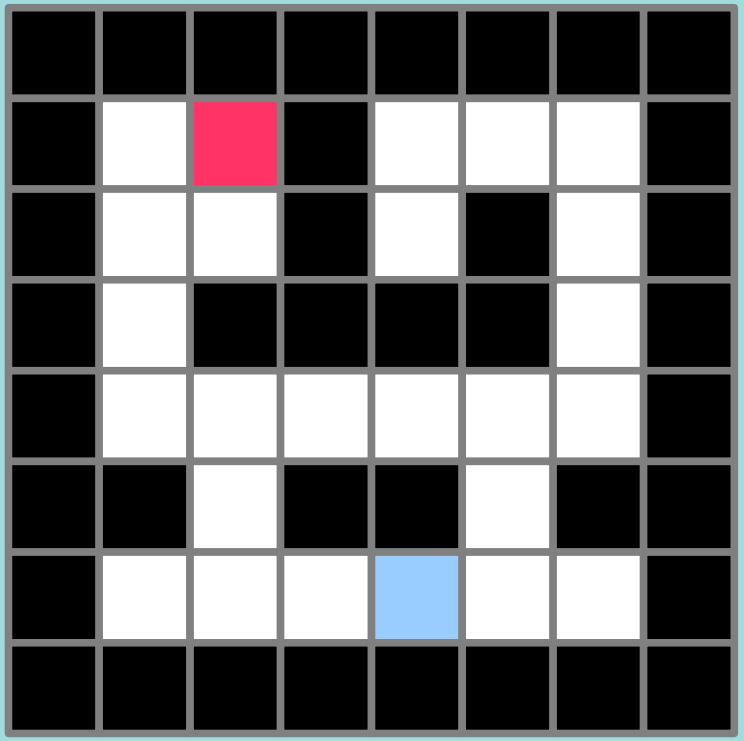
- Structure de labyrinthe
- Recherche de chemin
  - Algorithme de Lee / la vague
- Extensions
  - Dijkstra
  - A *
  - Autres problèmes

Recherche de chemins



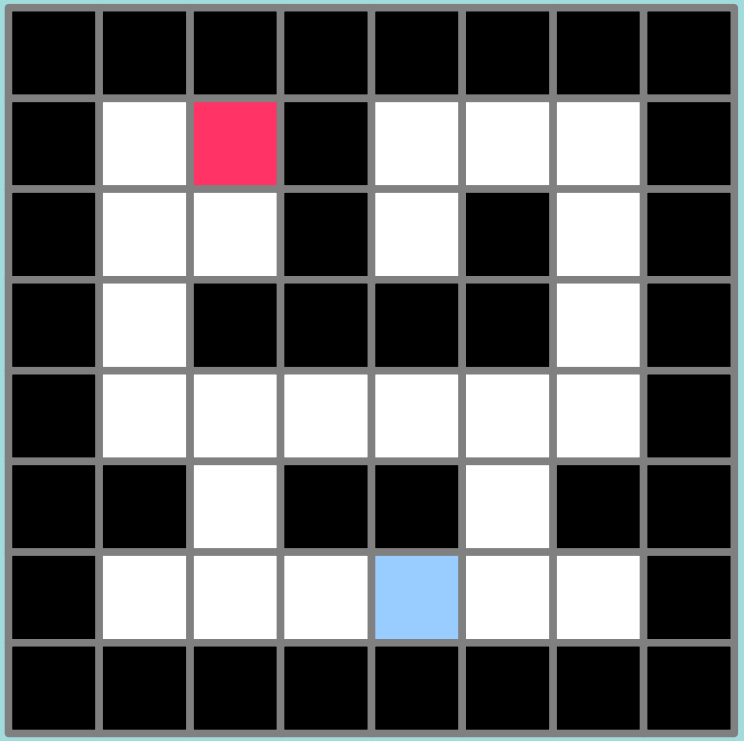
Recherche de chemins

- Quel est le chemin le plus court ?



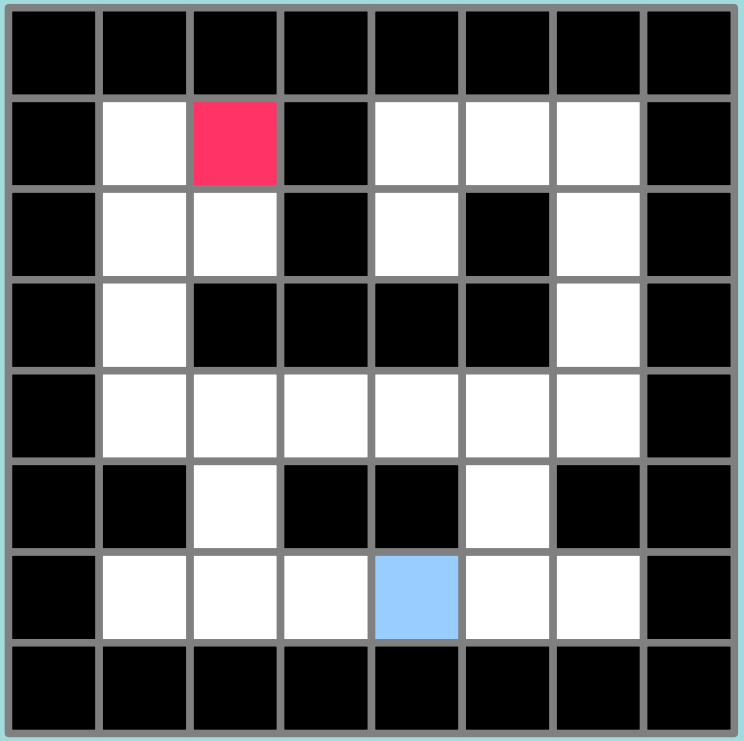
Recherche de chemins

- Quel est le chemin le plus court ?
  - Pourquoi difficile ?
- Solution possibles ?

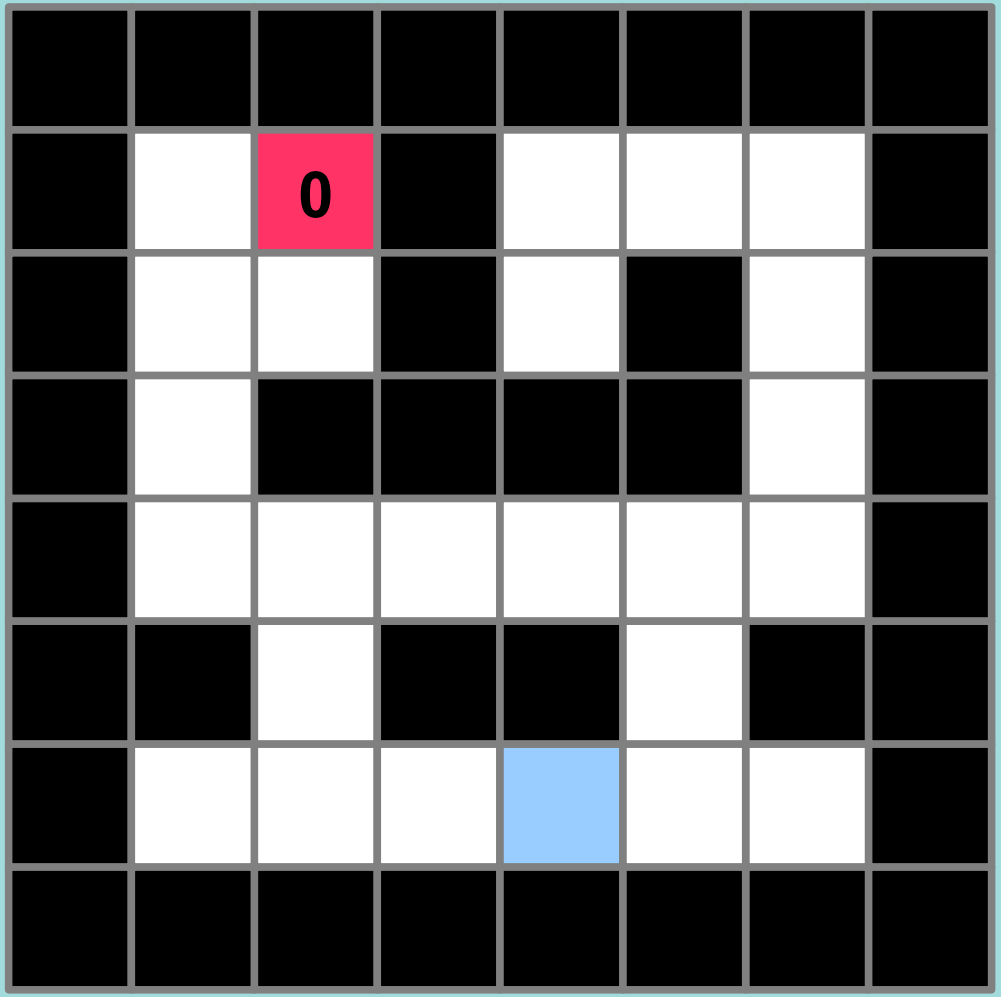


Recherche de chemins

- Quel est le chemin le plus court ?
  - Pourquoi difficile ?
- 1. Enumérer chemins
  - Explosion combinatoire
- 2. Raisonner sur distances
  - Quelle info trivial ?

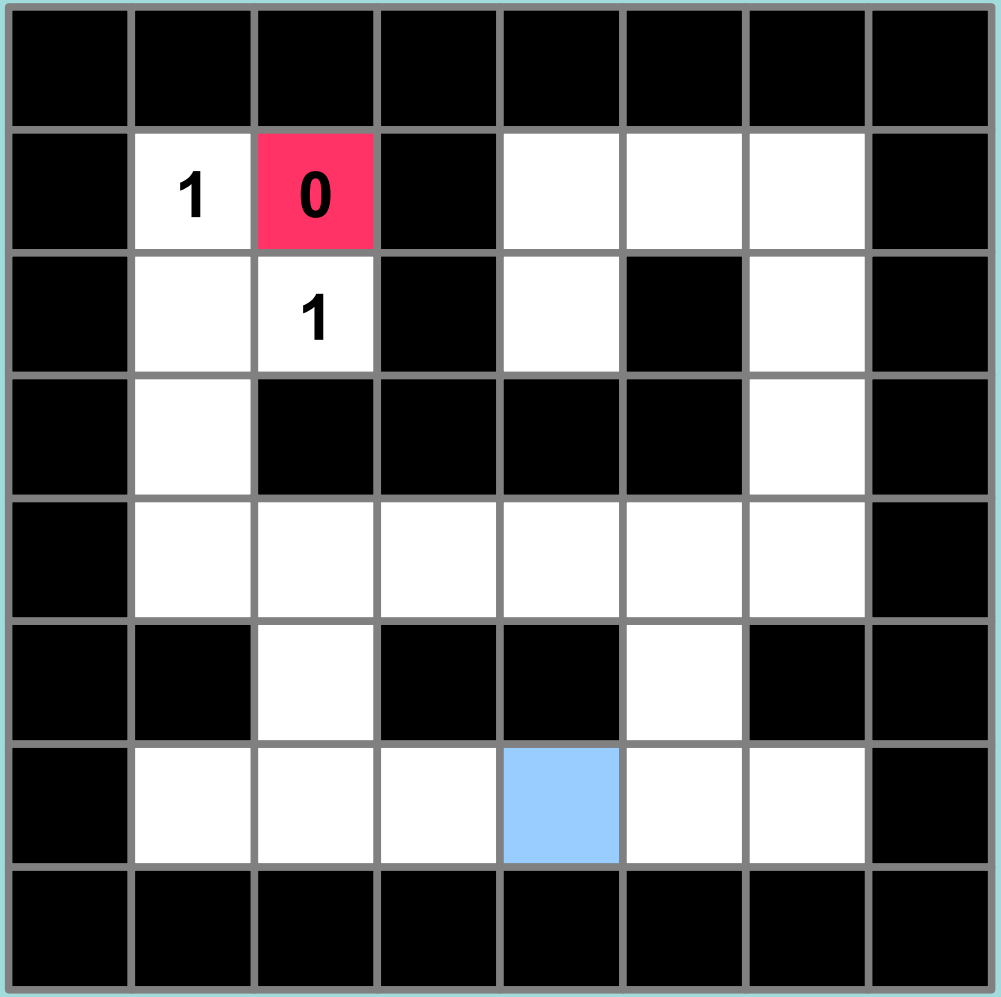


Recherche de chemins



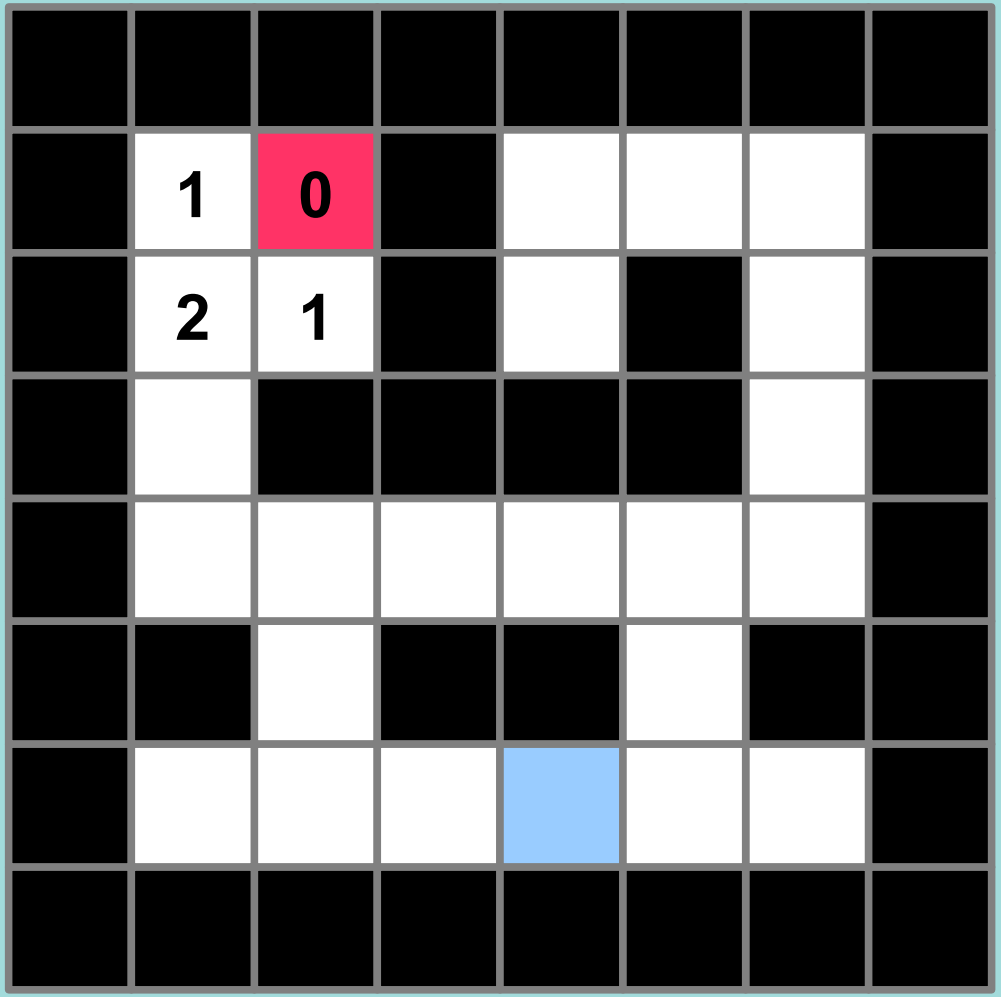
Algorithme de propagation

Recherche de chemins



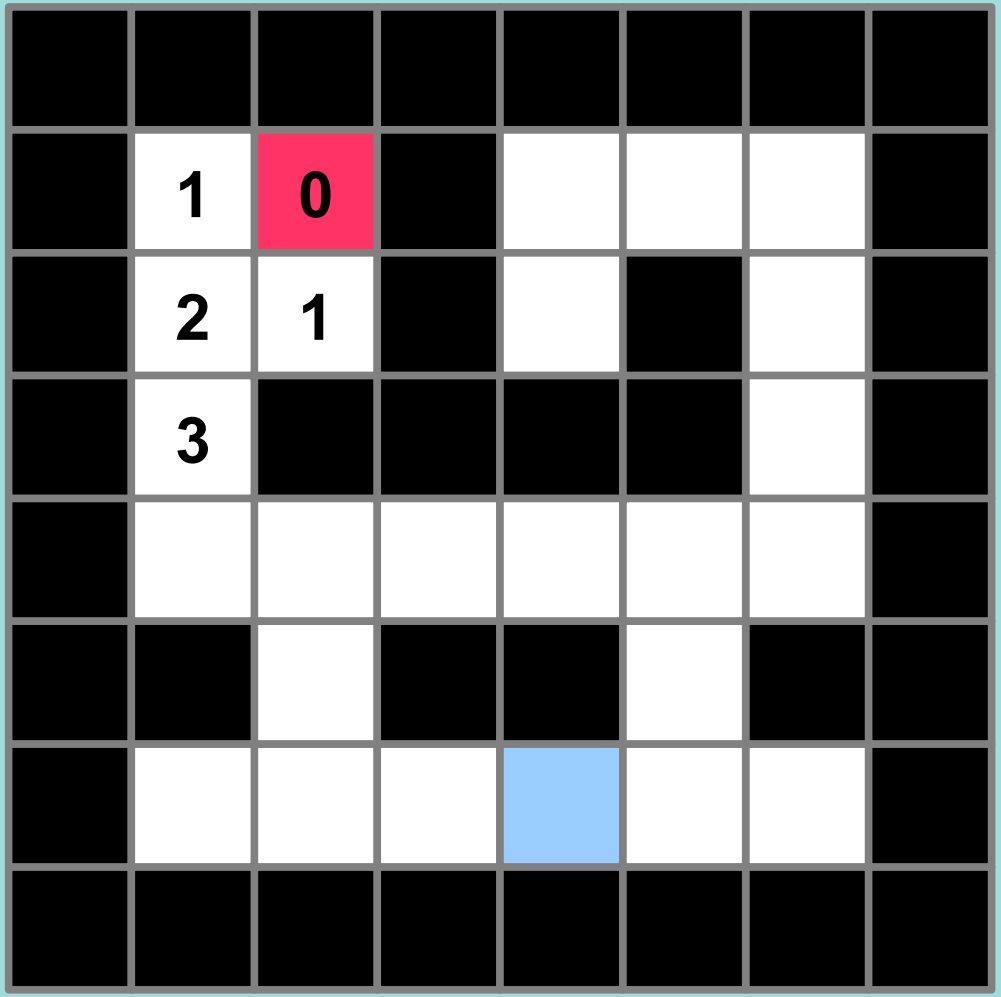
Algorithme de propagation

Recherche de chemins



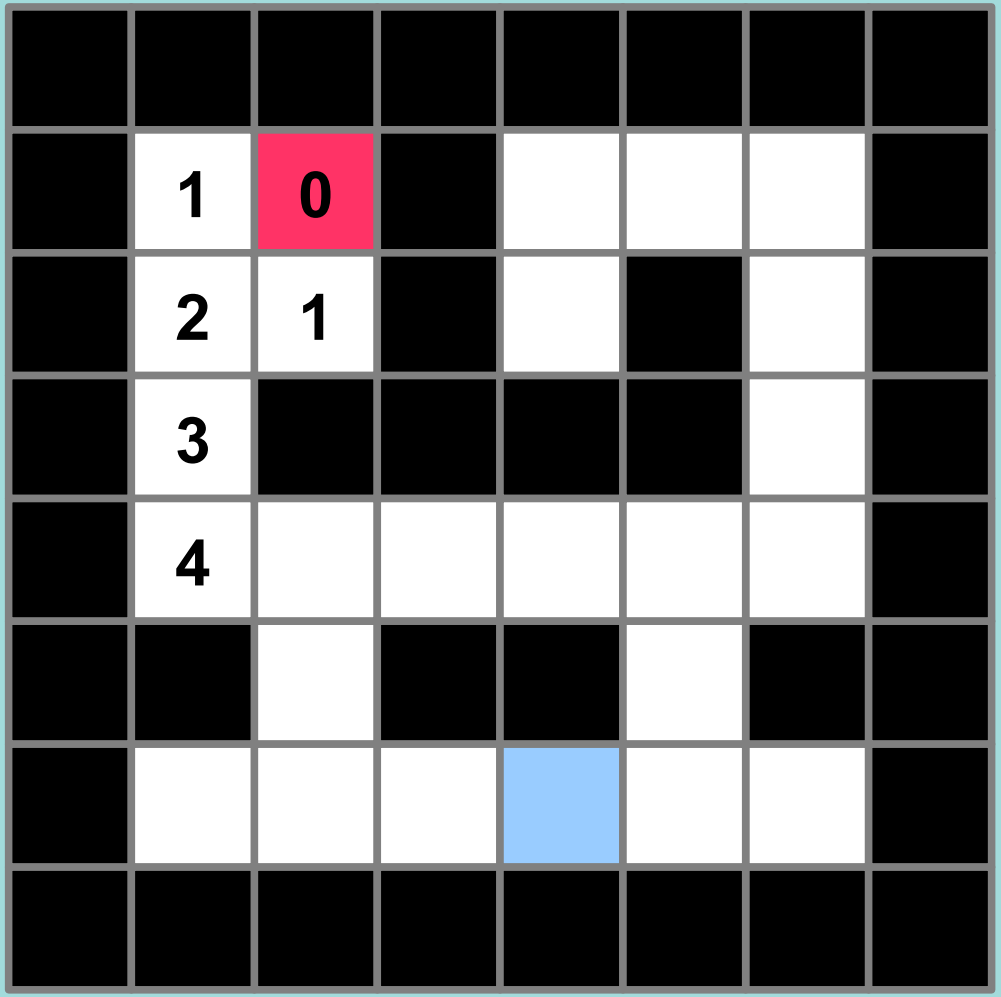
Algorithme de propagation

Recherche de chemins



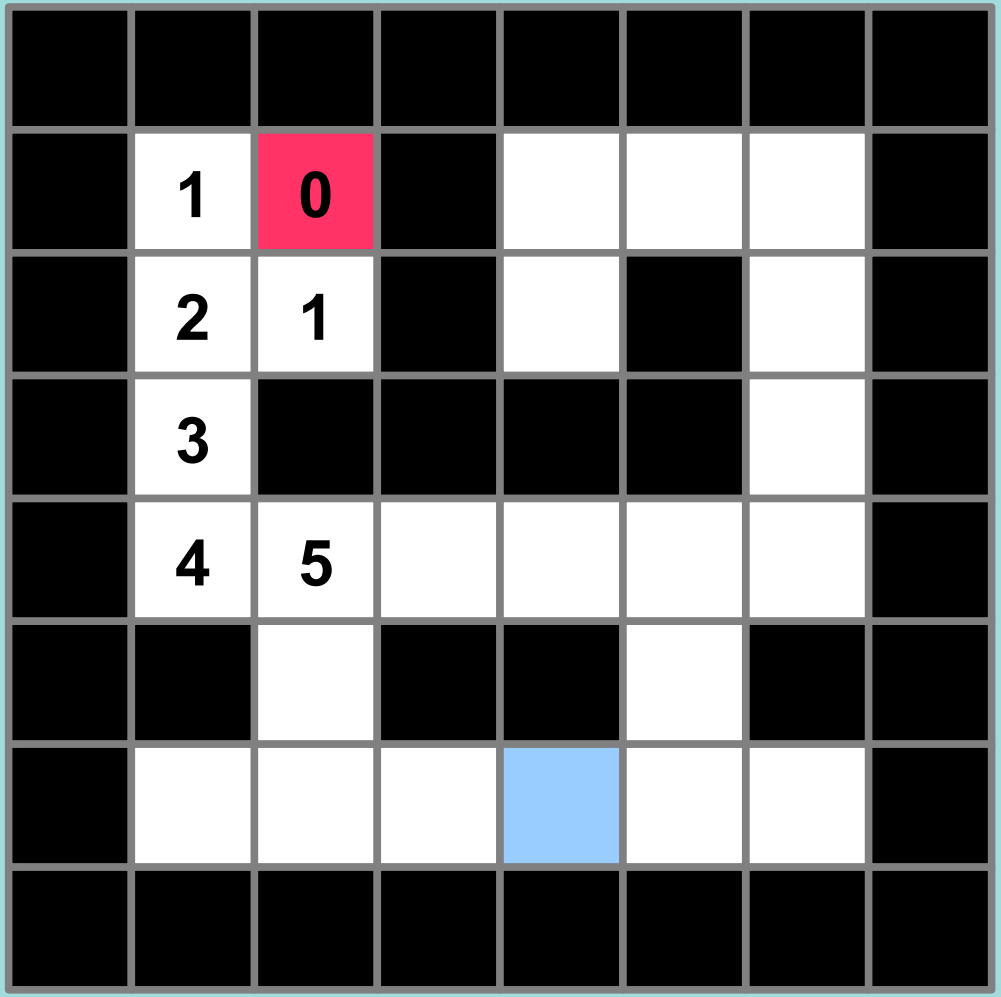
Algorithme de propagation

Recherche de chemins



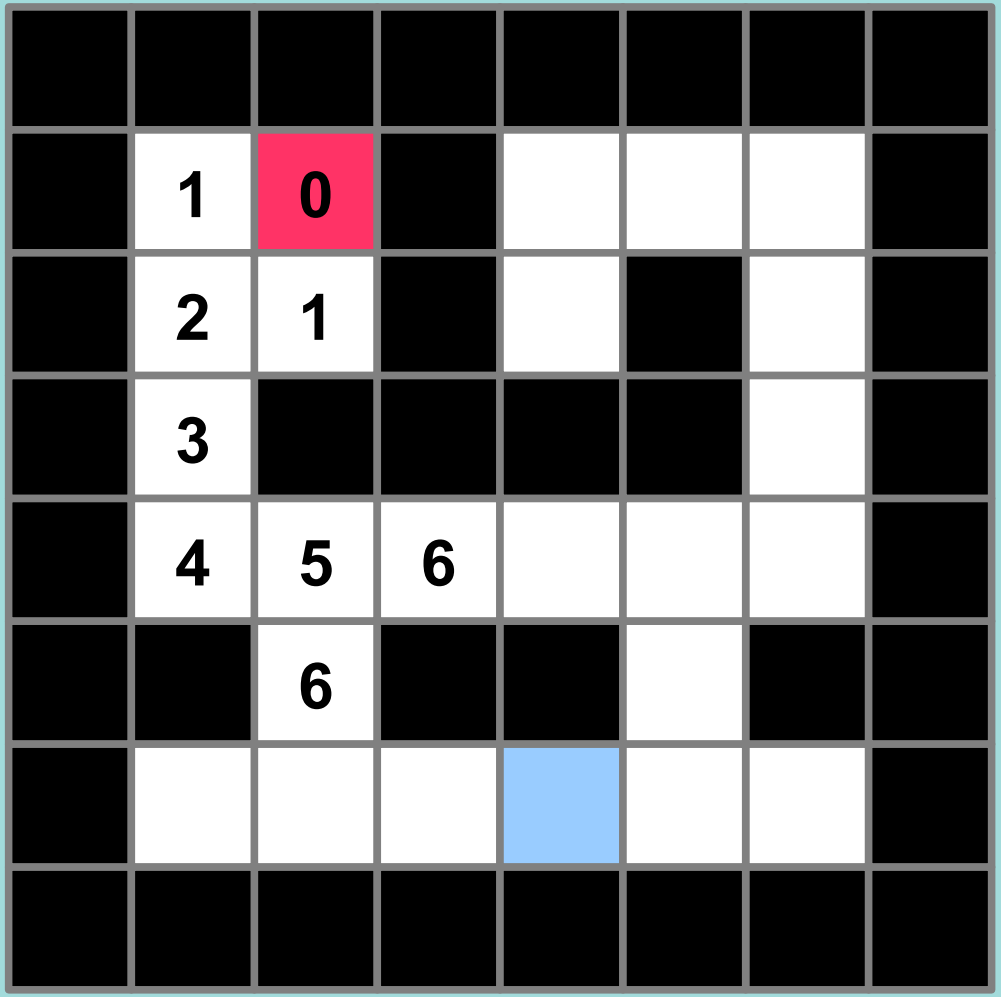
Algorithme de propagation

Recherche de chemins



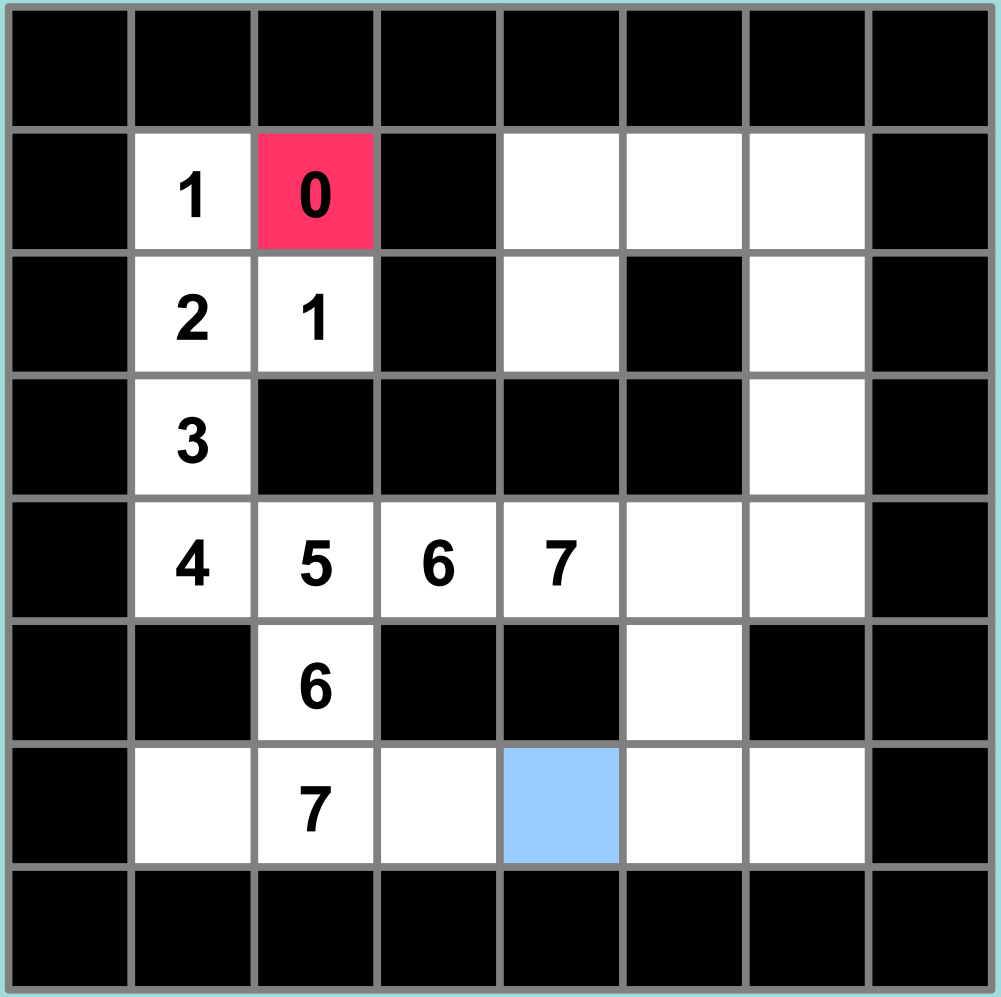
Algorithme de propagation

Recherche de chemins



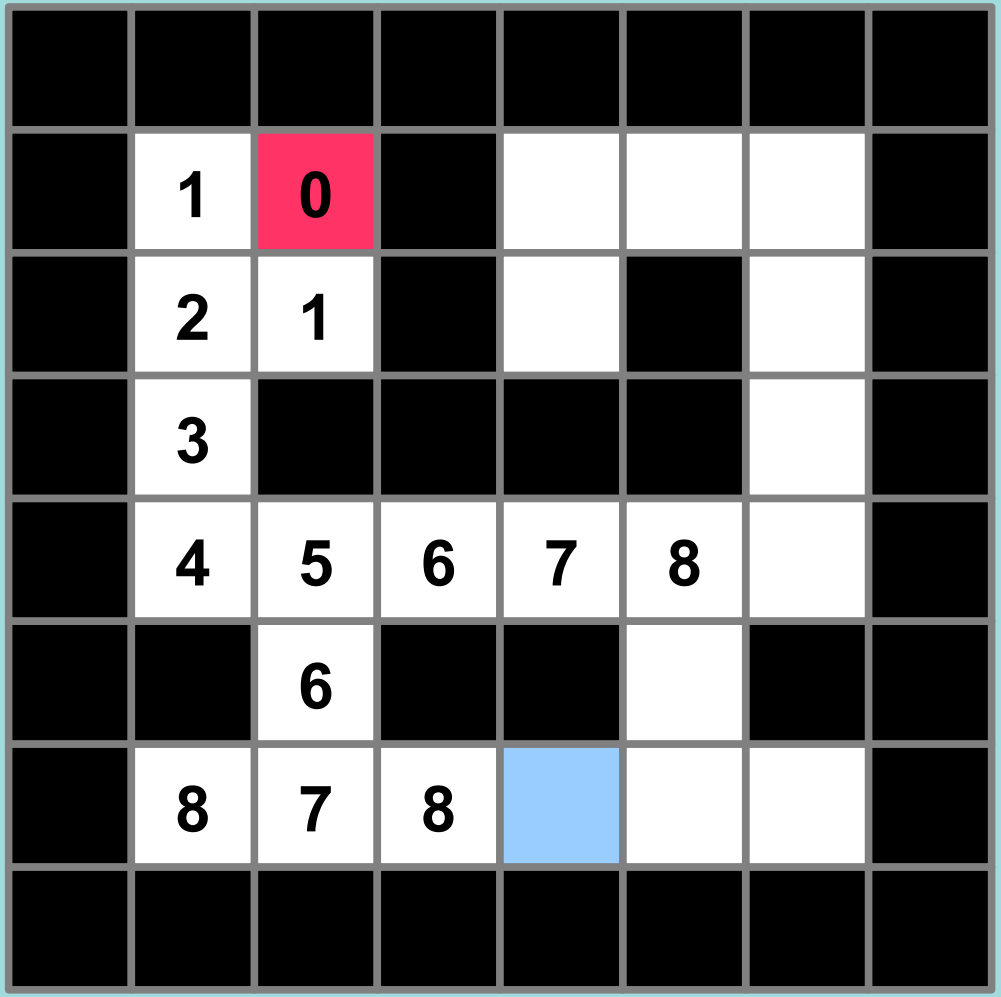
Algorithme de propagation

Recherche de chemins



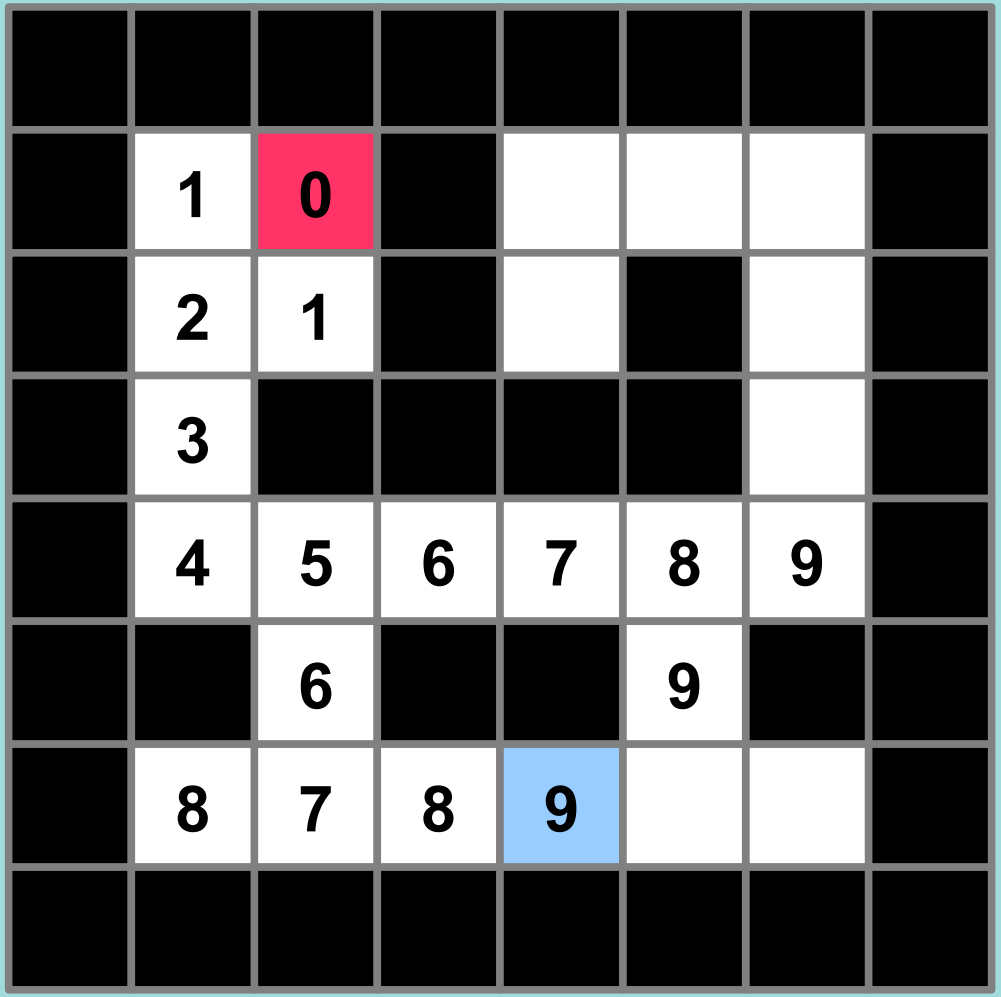
Algorithme de propagation

Recherche de chemins



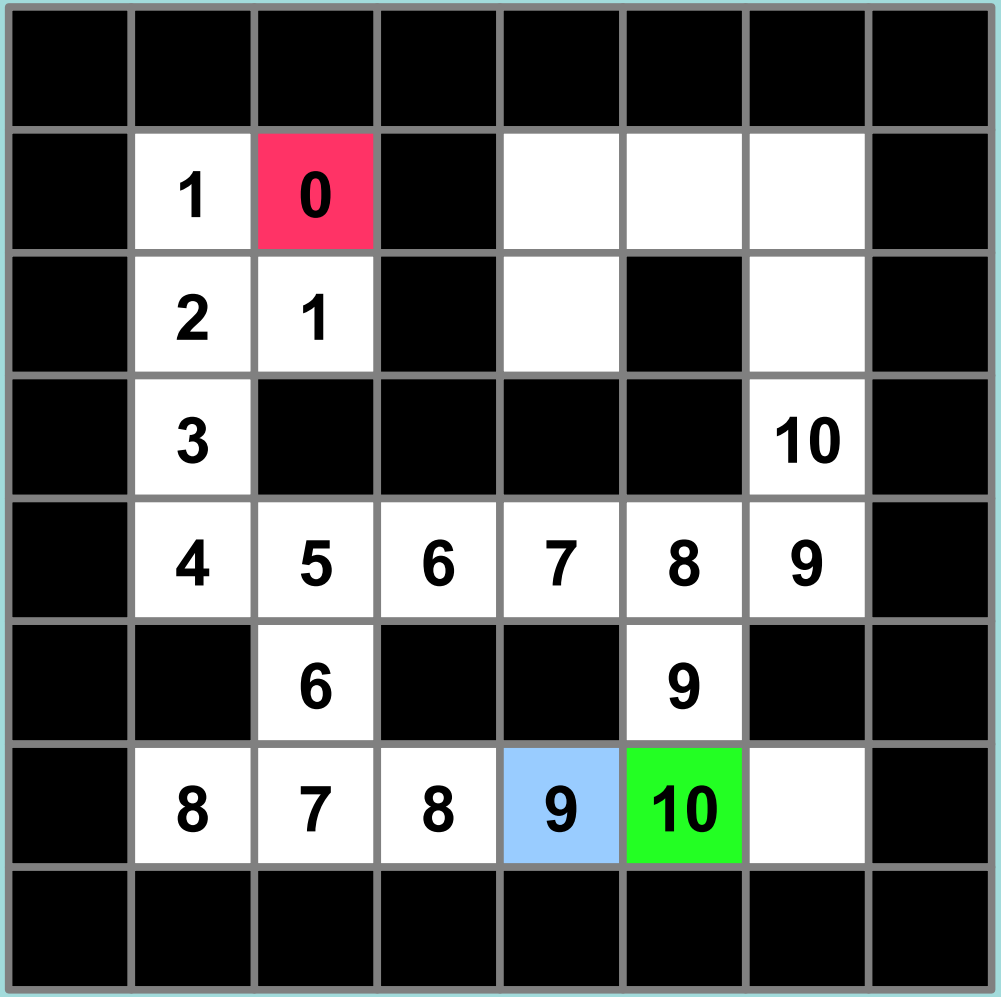
Algorithme de propagation

Recherche de chemins



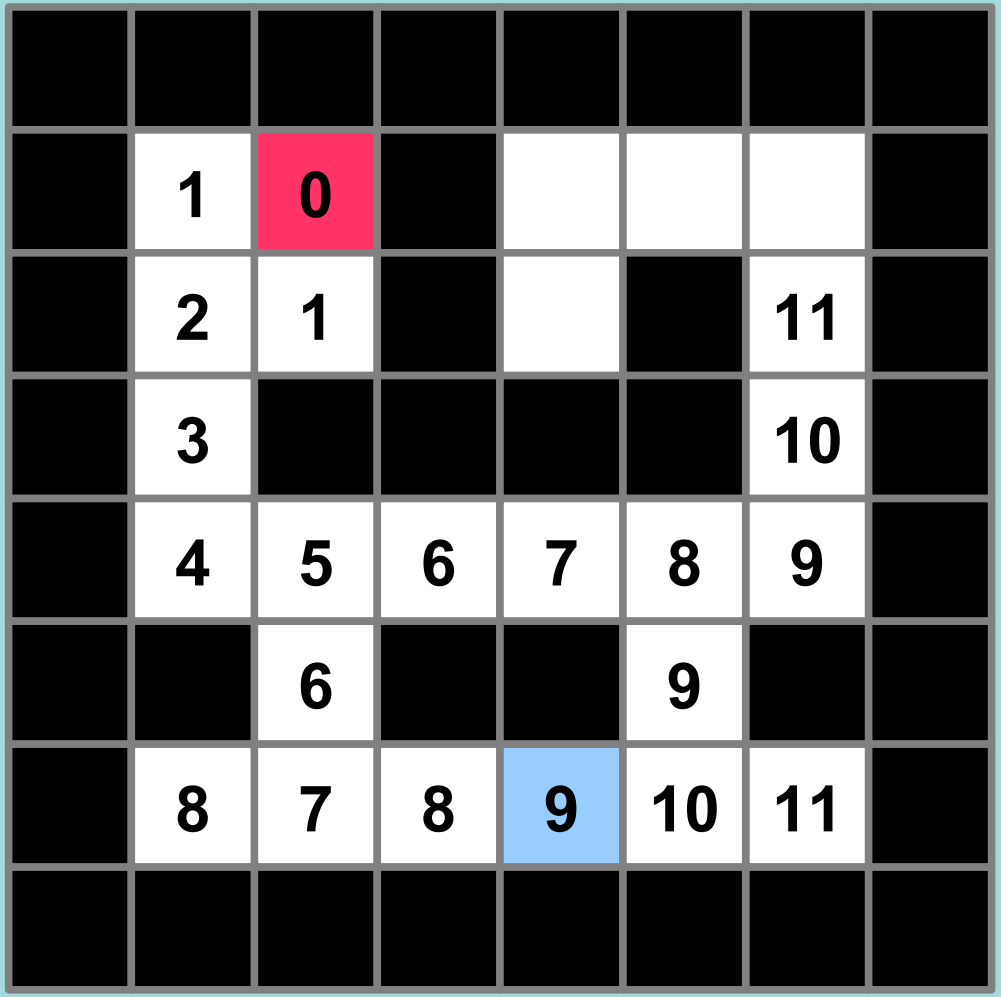
Algorithme de propagation

Recherche de chemins



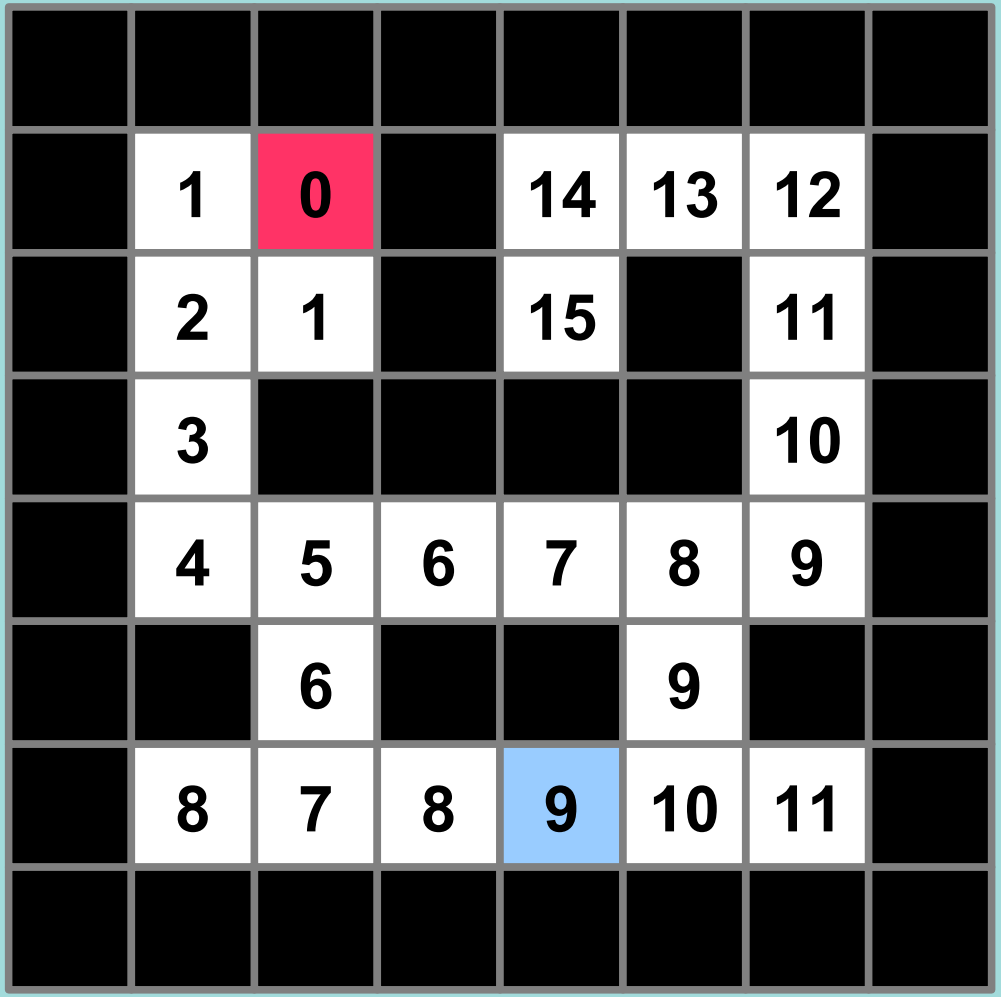
Algorithme de propagation

Recherche de chemins



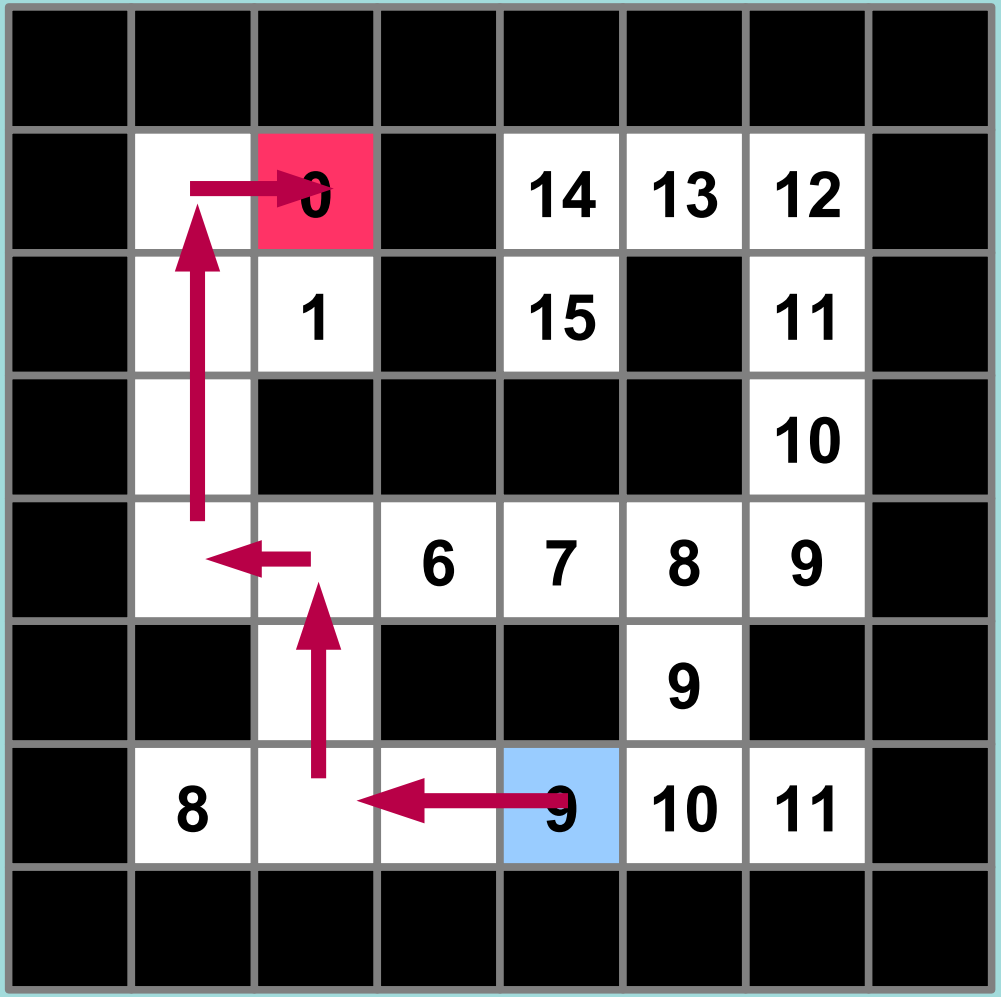
Algorithme de propagation

Recherche de chemins



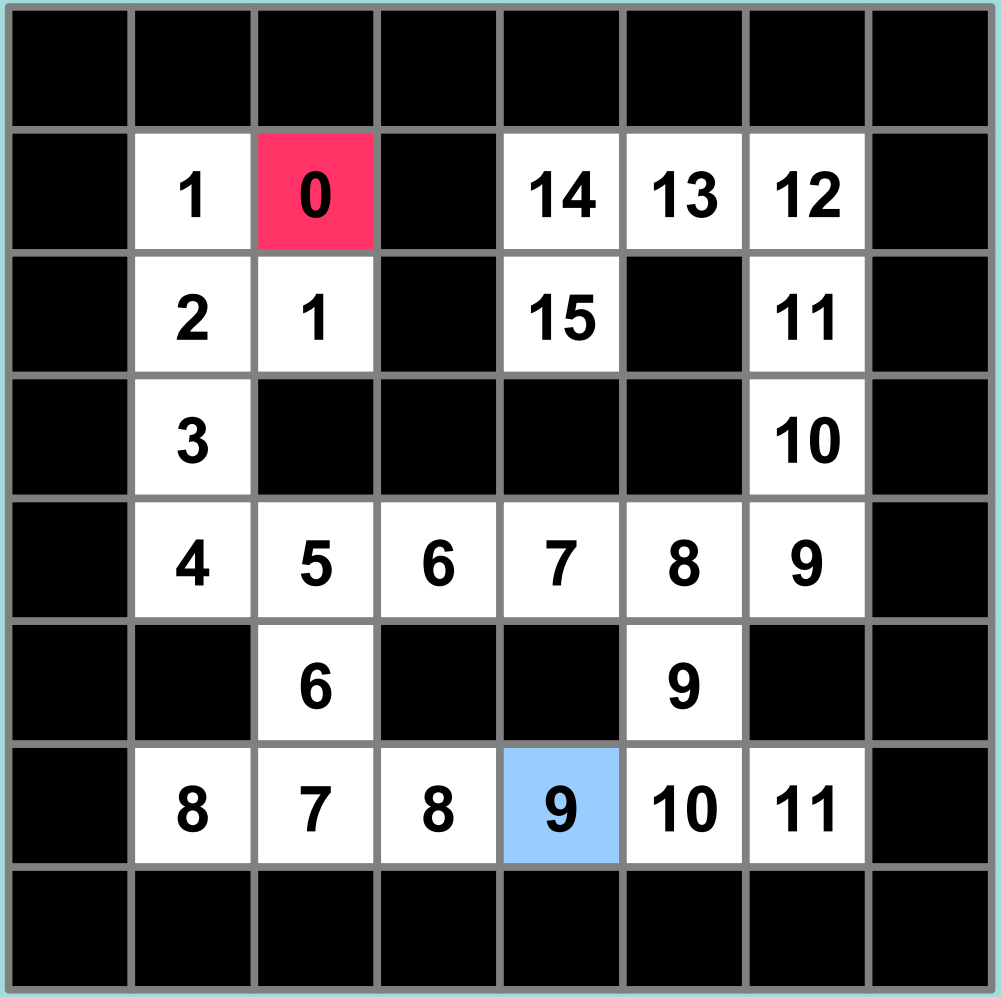
Algorithme de propagation

Recherche de chemins



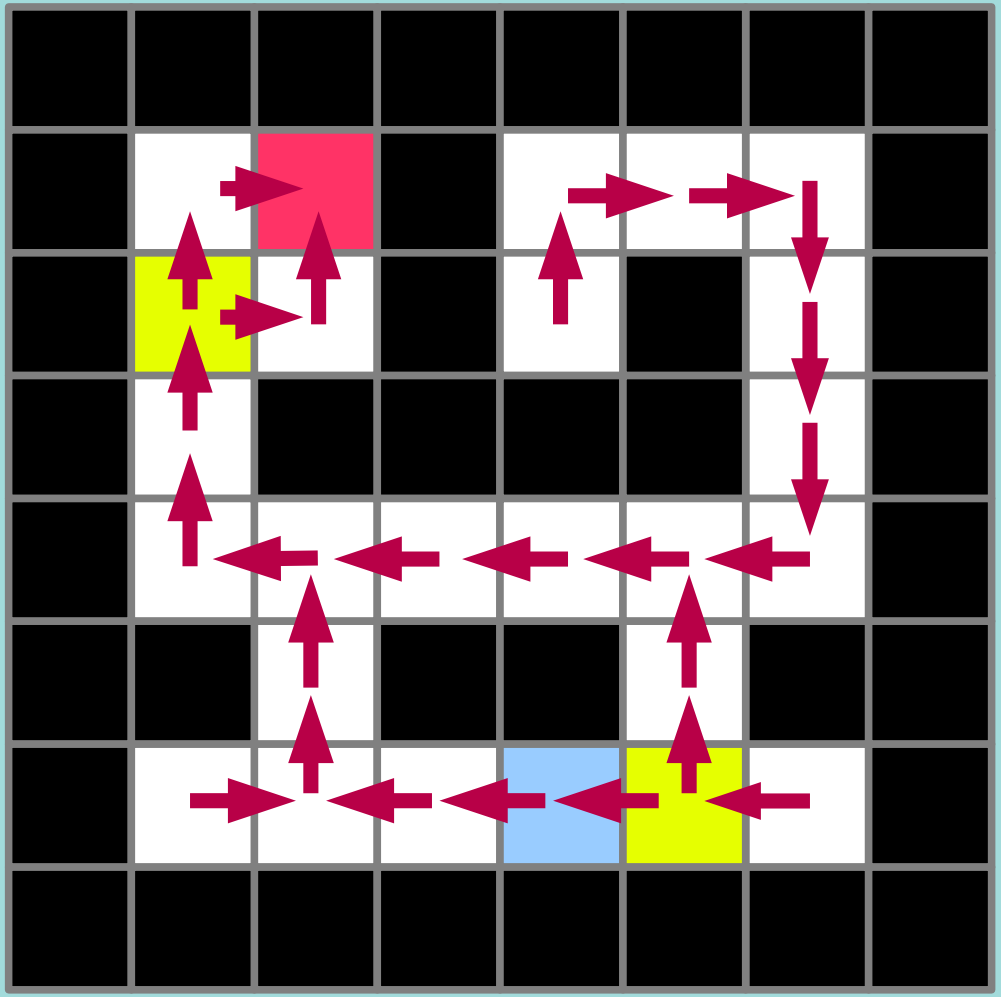
Algorithme de propagation

Recherche de chemins



Algorithme de propagation

Recherche de chemins

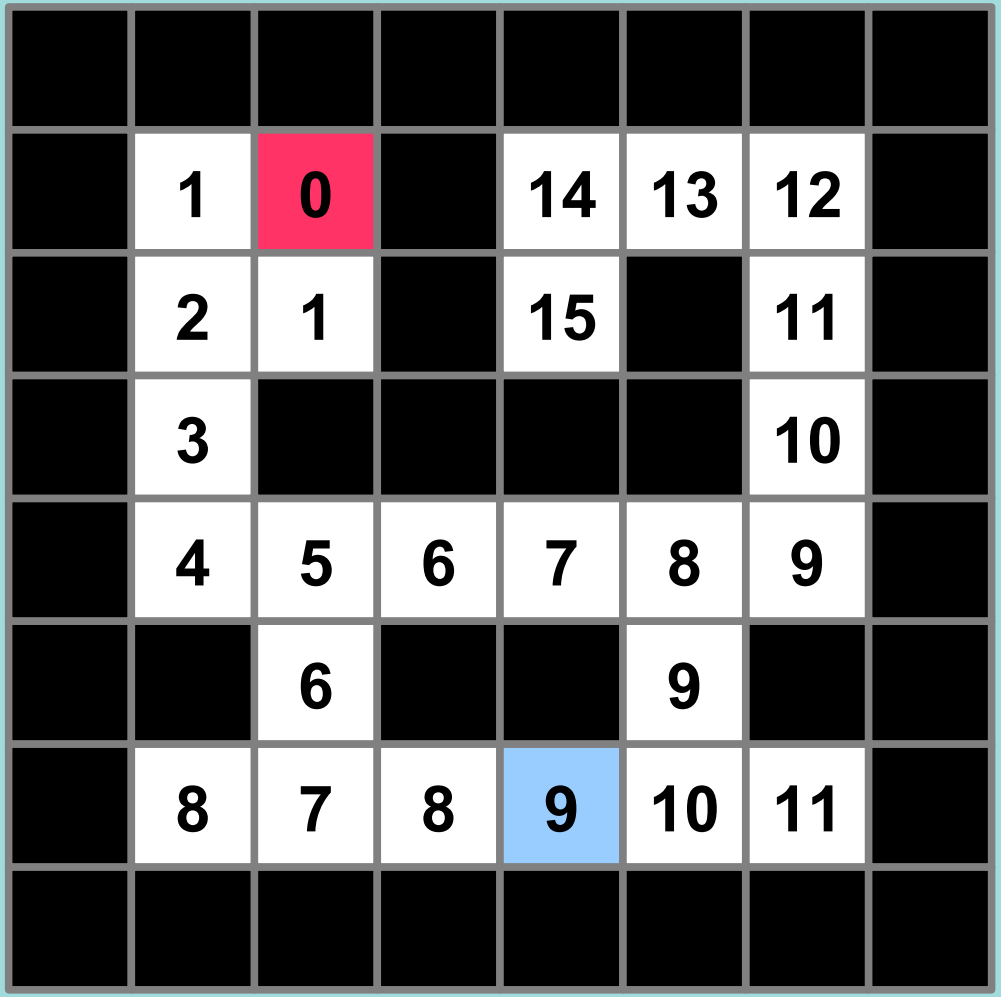


Algorithme de propagation

Coder en python

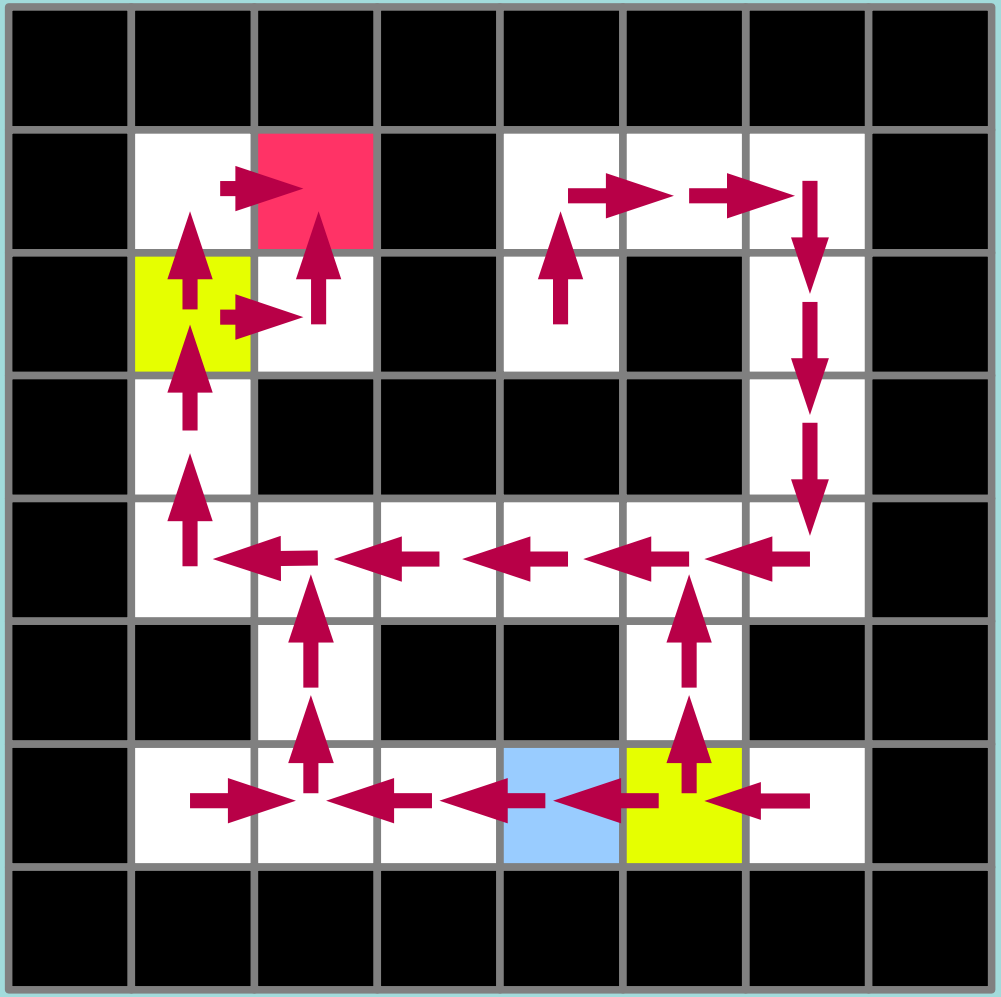
- Démarche
  - [X] Savoir résoudre à la main
  - [ ] Comprendre la stratégie mise en oeuvre
  - [ ] Ecrire algorithme de haut niveau
  - [ ] Traduire en commentaire
  - [ ] Ajouter le code entre les commentaires
- Intérêt
  - Séparer code de l'algo et réduire bugs
  - Avoir code correctement commenté

Recherche de chemins



Algorithme de propagation

Recherche de chemins



Algorithme de propagation

Messages à emporter

- 1. Algorithme de recherche de chemin
  - Principe de propagation de valeur (accessible)
  - Algorithme de Lee
  - Gestion de frontières (liste ouverte)

- 2. Manière de coder
  - Comprendre le probleme
  - Résoudre exemple
  - Ecrire les commentaires
  - Ecrire le code

Plan

- Structure de labyrinthe
- Recherche de chemin
  - Algorithme de Lee / la vague
- Extensions
  - Dijkstra
  - A *
  - Autres problèmes

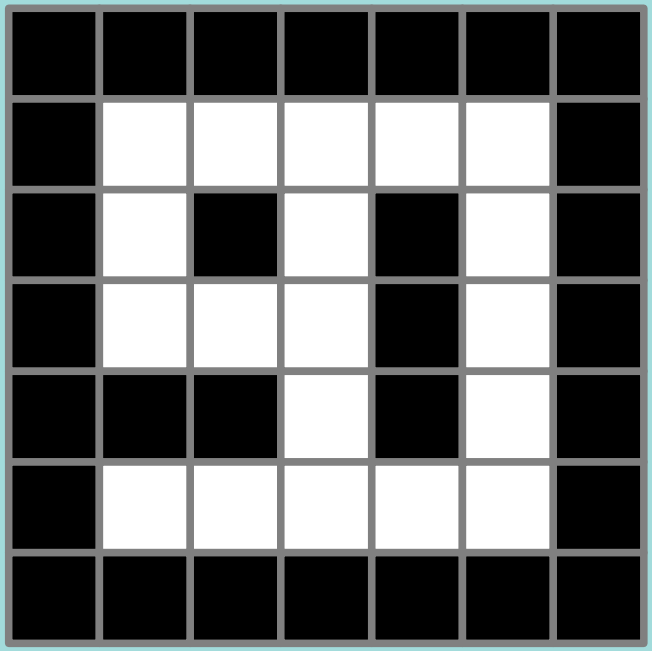
3 Extensions

- Ajoute des couts aux arcs
  - Dijkstra
- Accélérer la recherche
  - Algorithme A*
- Etendre à d'autres cadres
  - Théorie des graphes

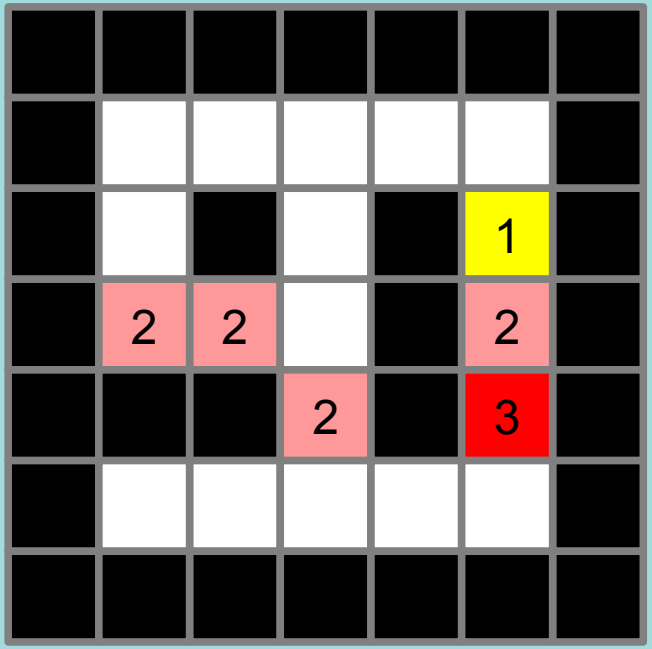
Dijkstra

Edsger Dijkstra (1930-2002)

- Tableau de coûts
  - Exemple altitude sur les cases
  - Ajoute coût de la case



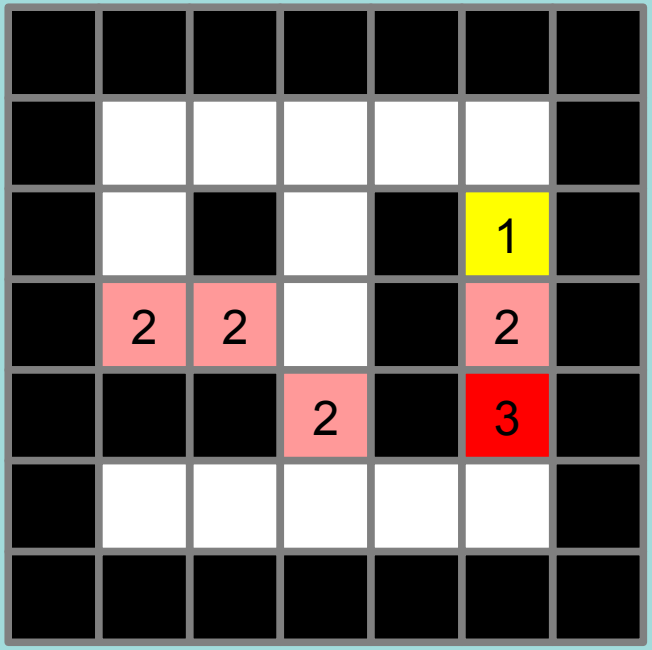
Murs



Cout = difference altitude

Dijkstra

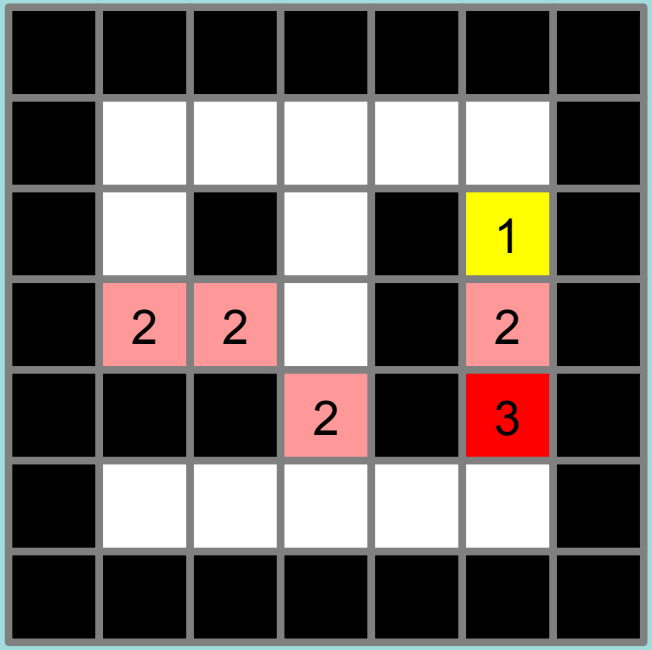
- Principe
  - Développe cases distance plus faible
  - Met à jour si valeur est meilleure
- Algo
  - Trier la liste ouverte par coût
  - Vérifier chemin réalisable
- Garantie
  - Car coûts > 0



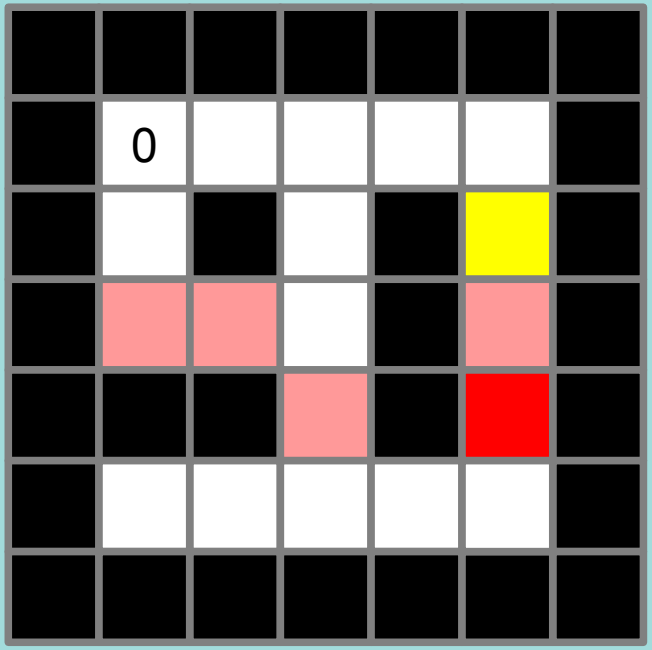
Coût = différence altitude

Dijkstra

- Principe
  - Développe cases distance plus faible
  - Met à jour si valeur est meilleure



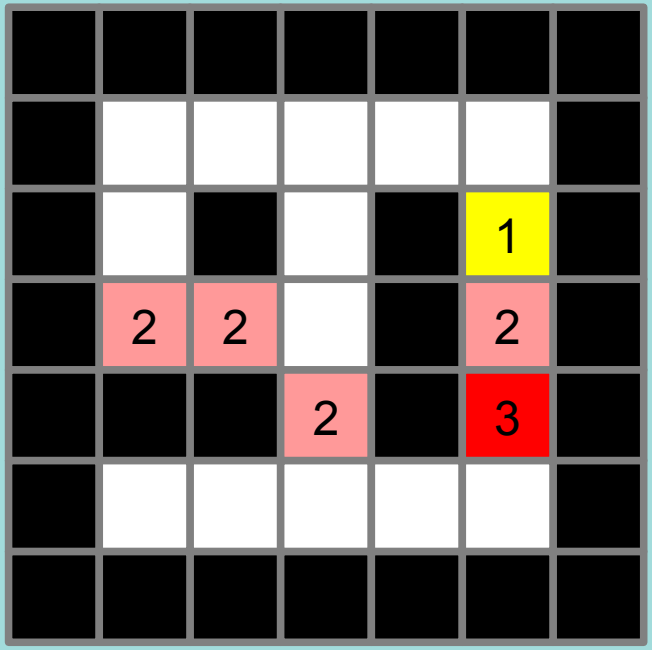
Coût = différence altitude



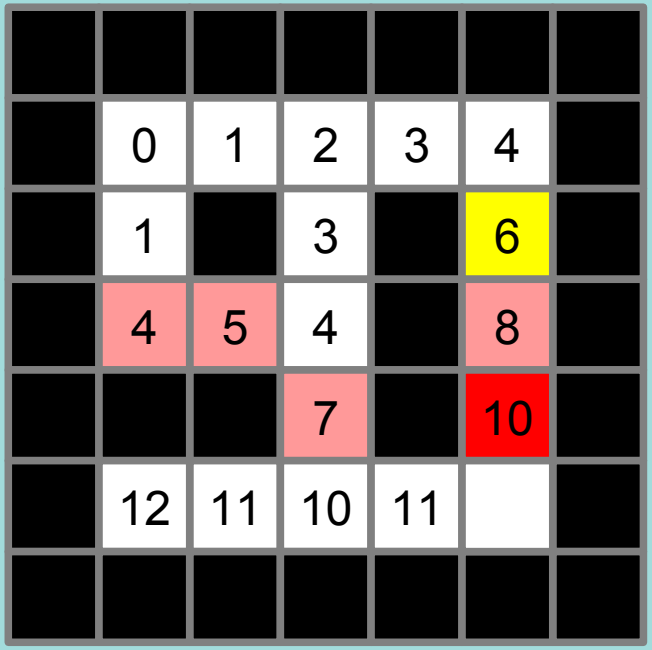
Distances

Dijkstra

- Principe
  - Développe cases distance plus faible
  - Met à jour si valeur est meilleure



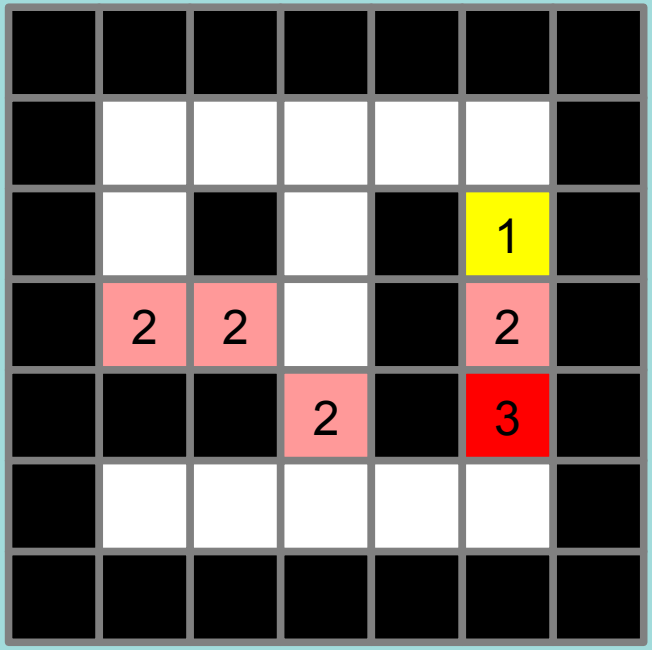
Coût = différence altitude



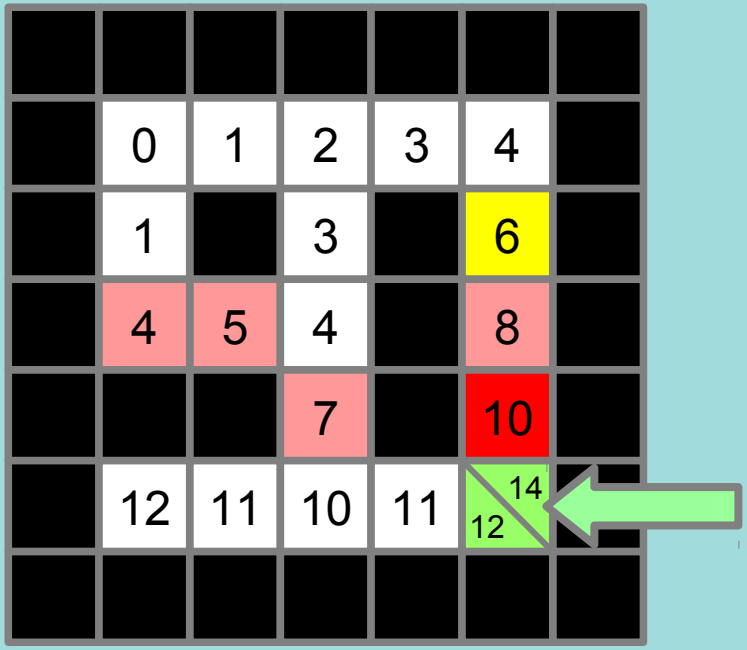
Distances

Dijkstra

- Principe
  - Développe cases distance plus faible
  - Met à jour si valeur est meilleure



Coût = différence altitude

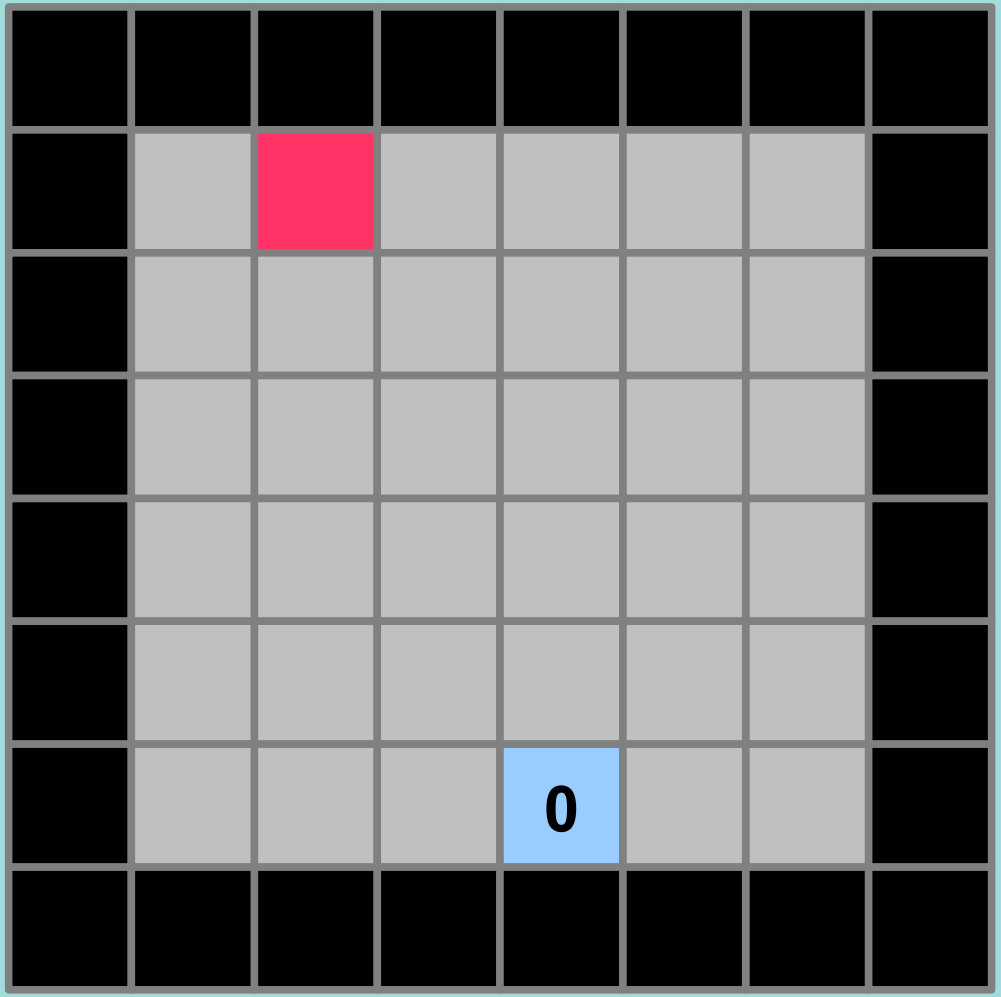


Distances

3 Extensions

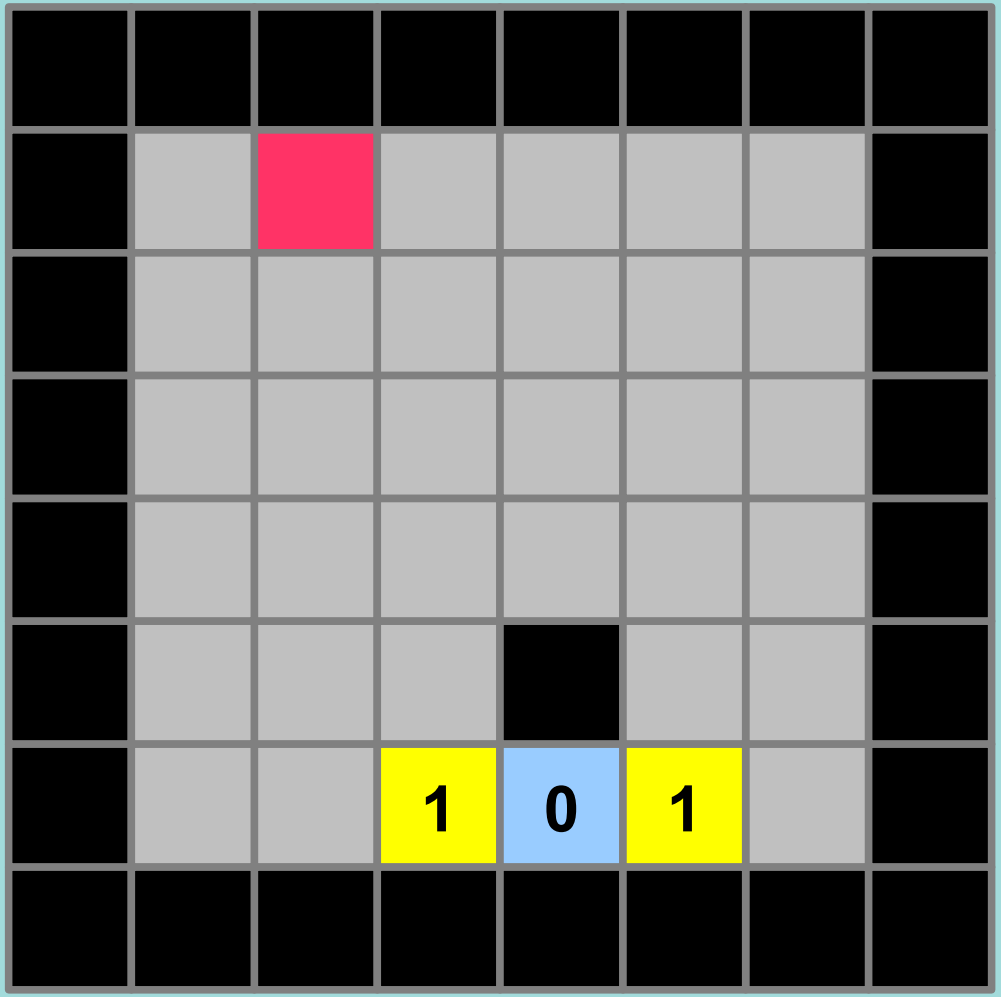
- Ajoute des couts aux arcs
  - Dijkstra
- Accélérer la recherche
  - Algorithme A*
- Etendre à d'autres cadres
  - Théorie des graphes

Algorithme A*



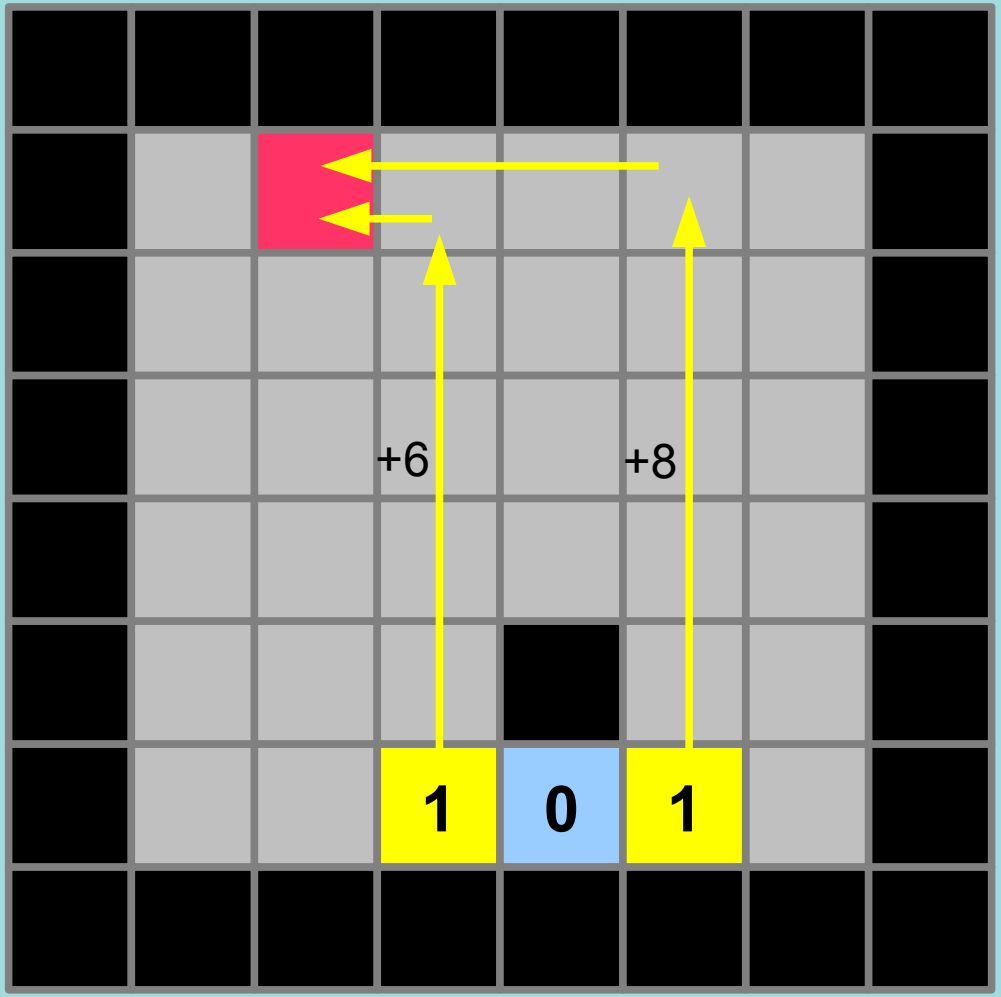
Algorithme de propagation

Algorithme A*



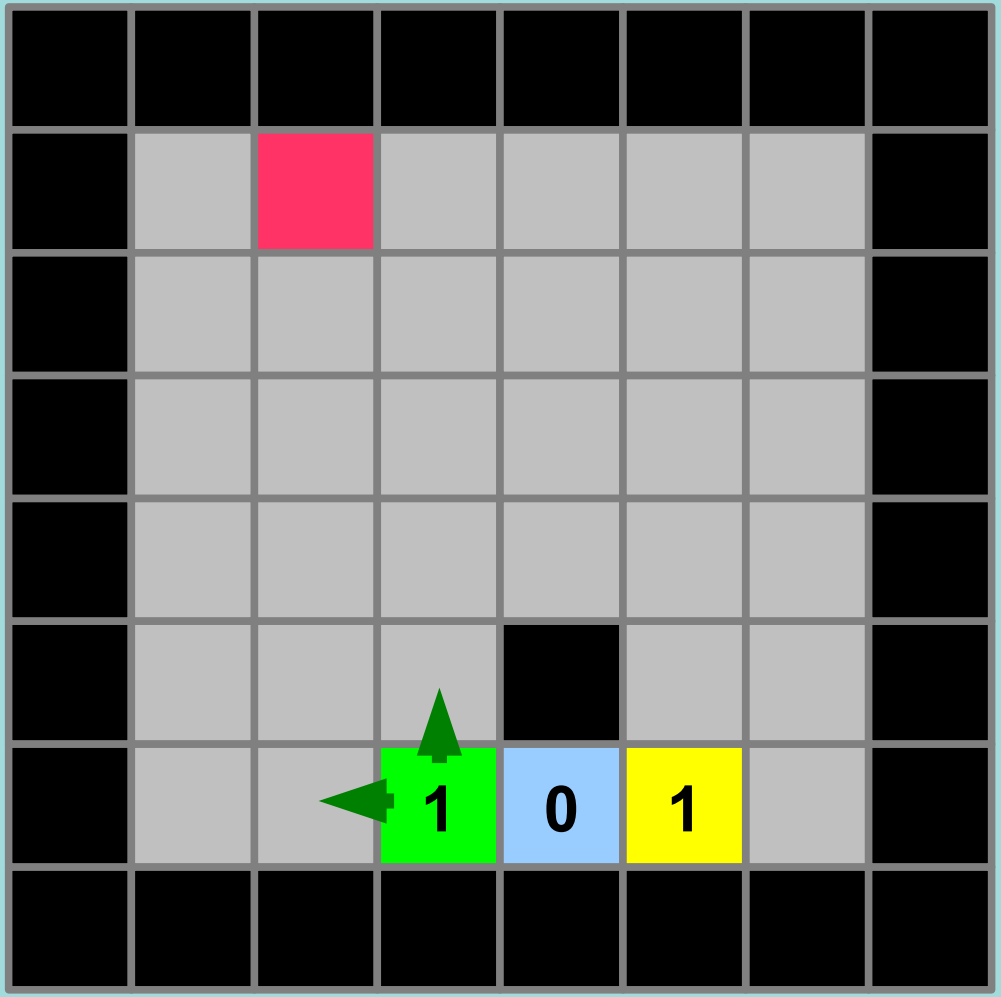
Algorithme de propagation + optimisme

Algorithme A*



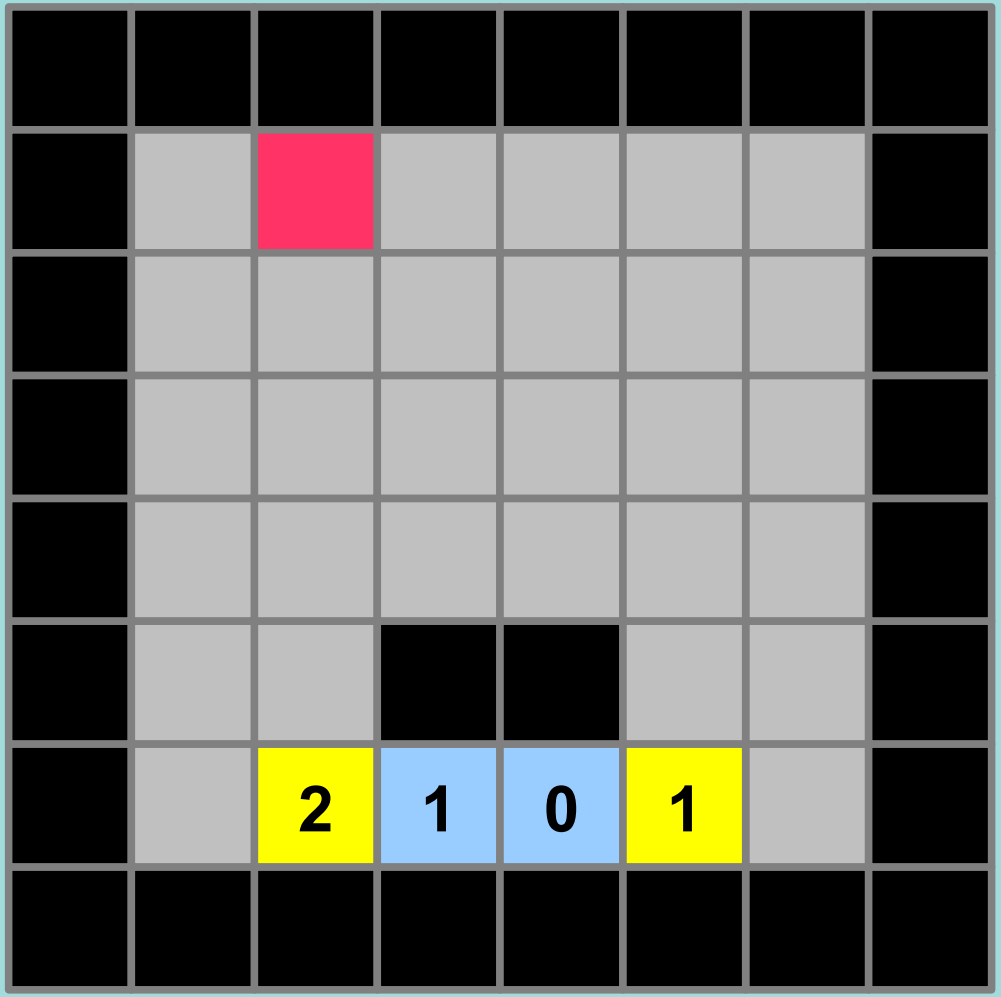
Algorithme de propagation + optimisme

Algorithme A*



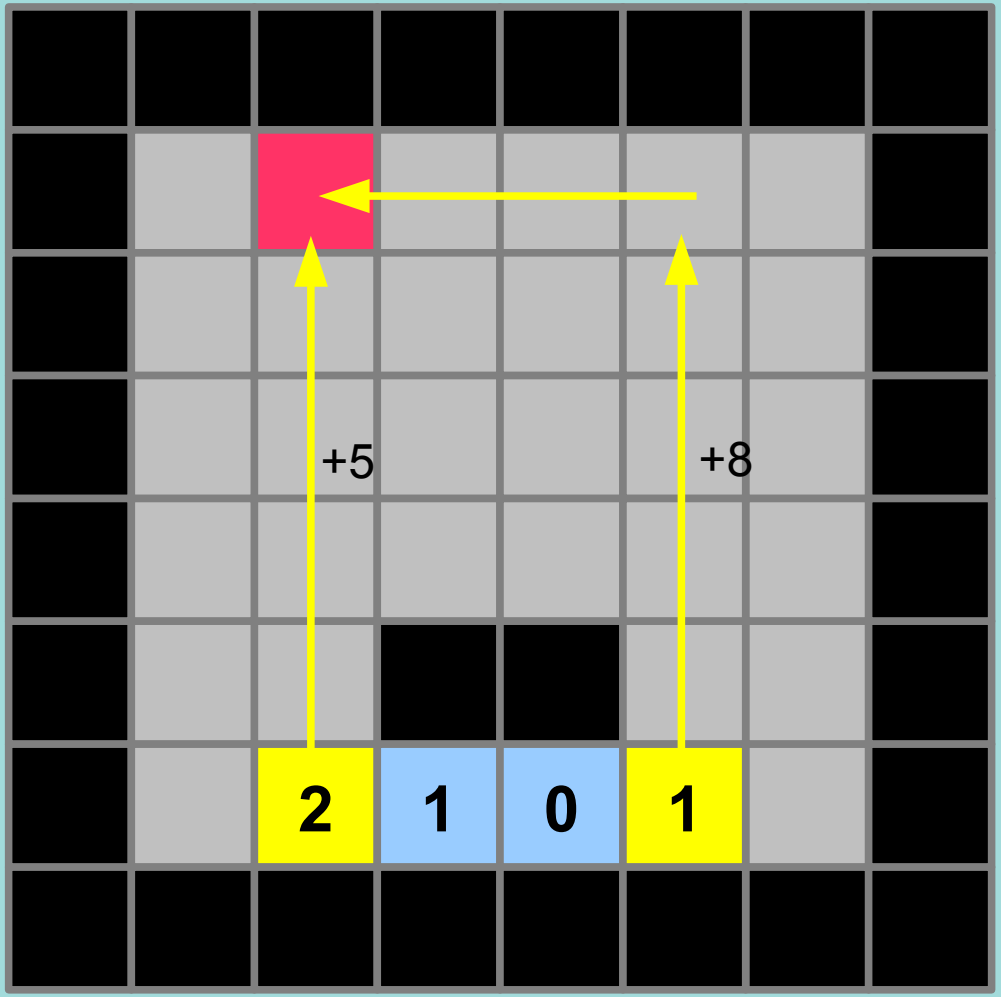
Algorithme de propagation + optimisme

Algorithme A*



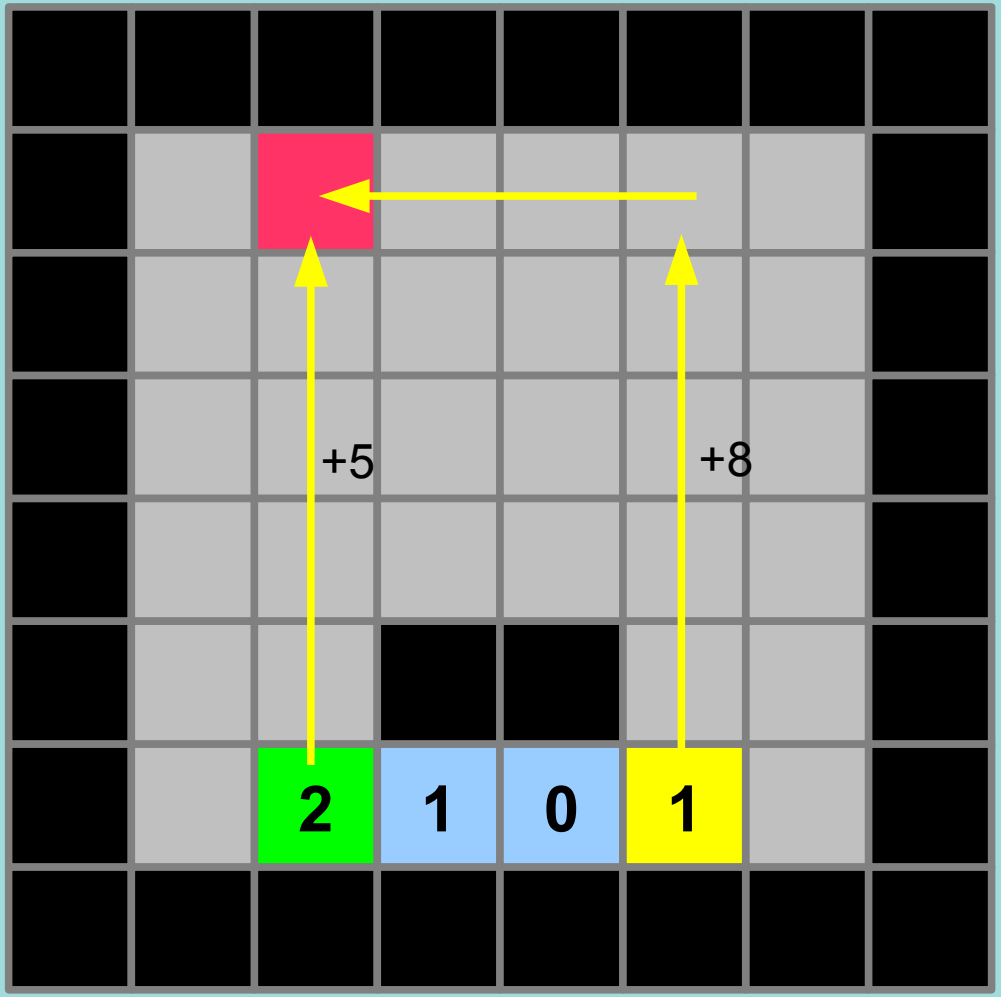
Algorithme de propagation + optimisme

Algorithme A*



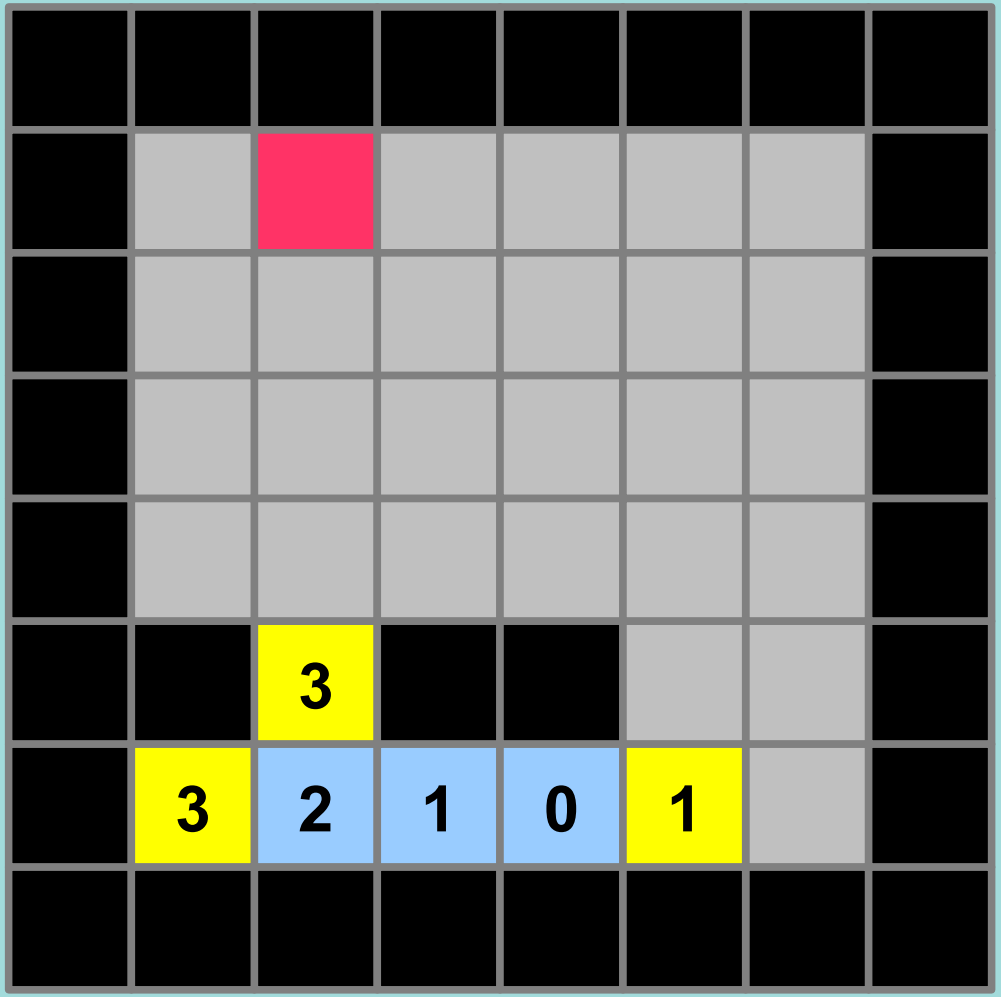
Algorithme de propagation + optimisme

Algorithme A*



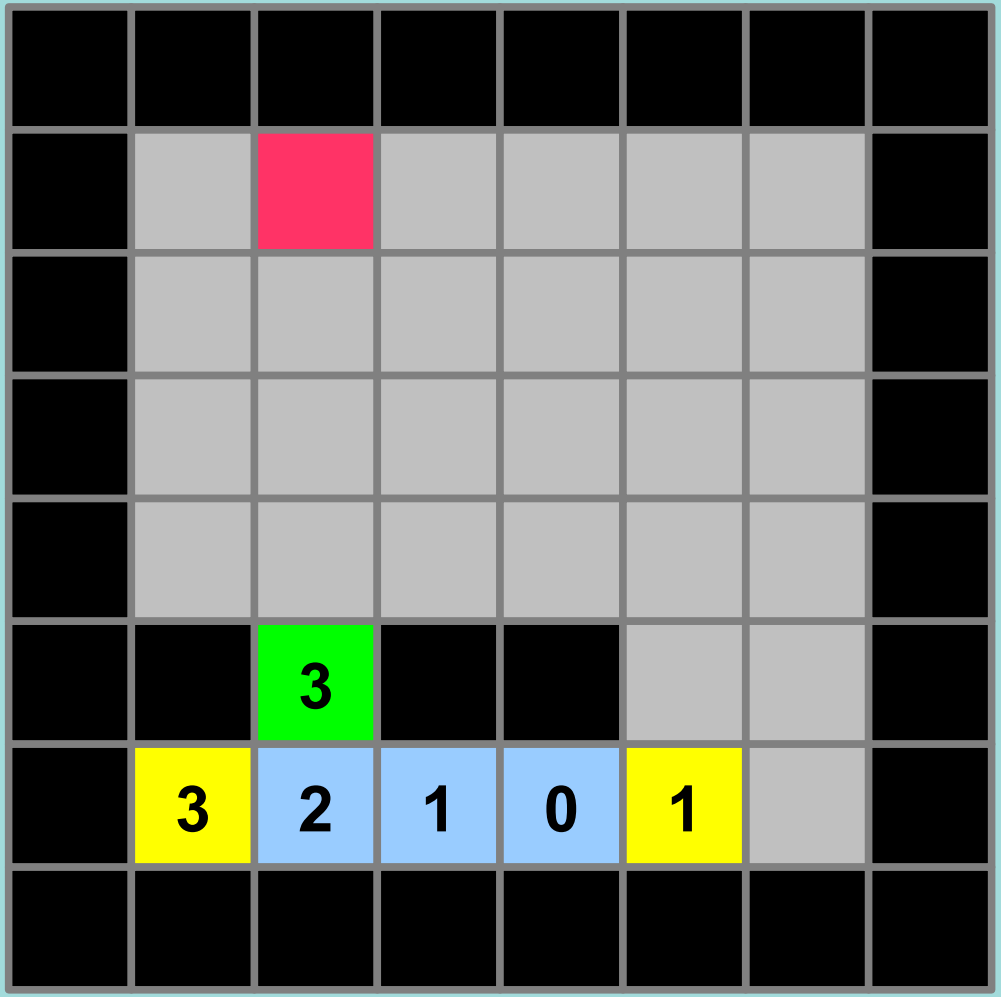
Algorithme de propagation + optimisme

Algorithme A*



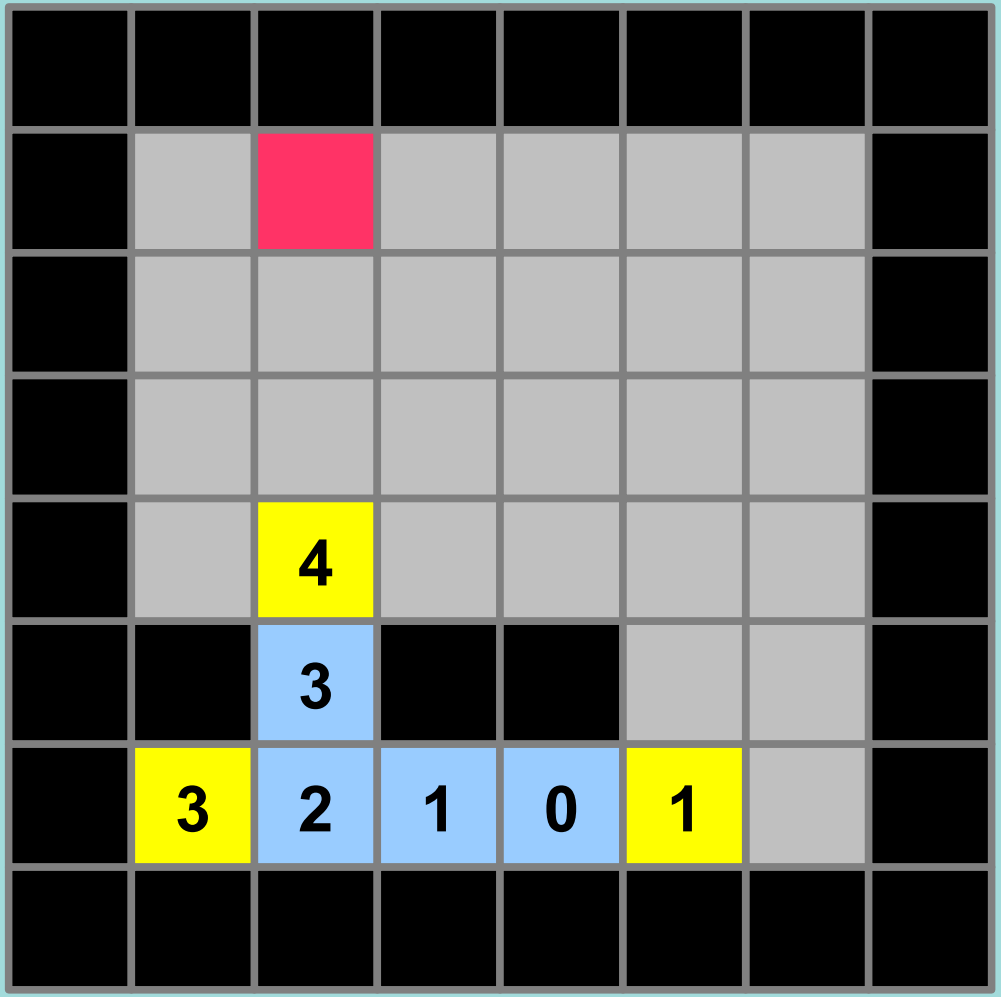
Algorithme de propagation + optimisme

Algorithme A*



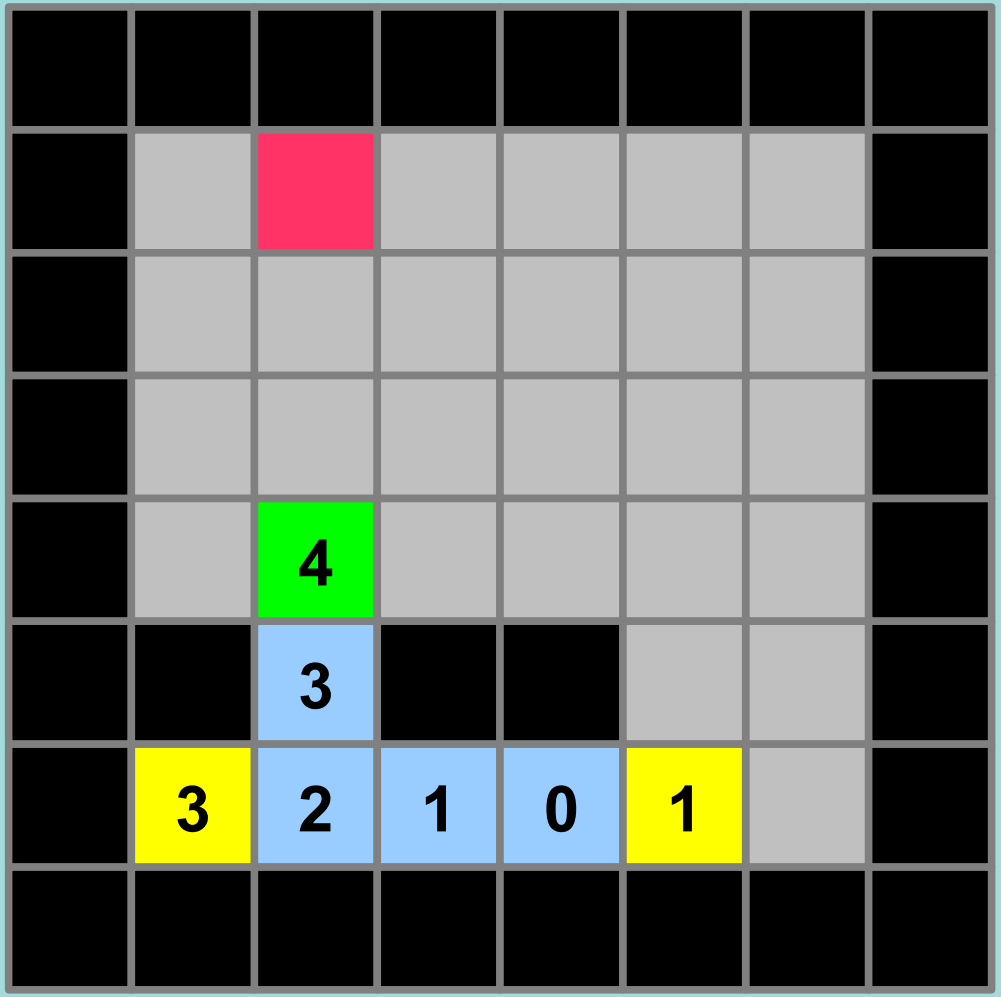
Algorithme de propagation + optimisme

Algorithme A*



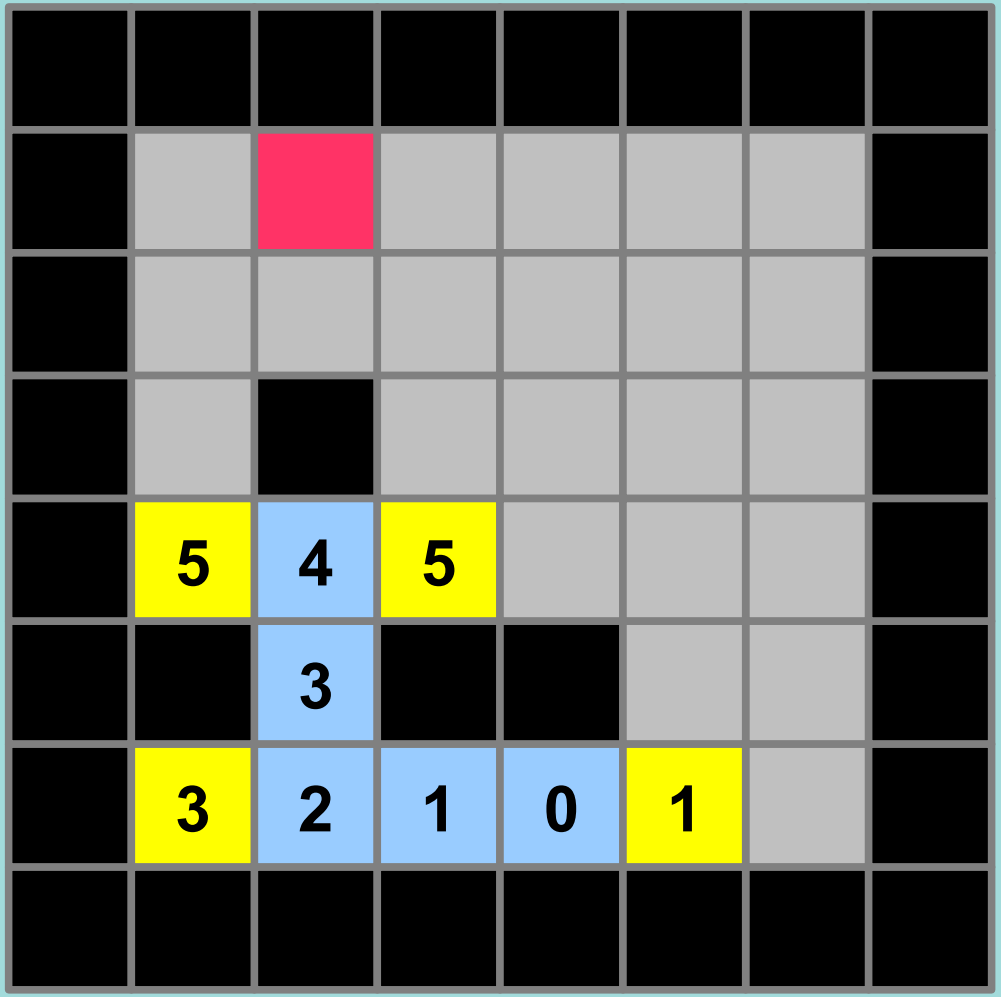
Algorithme de propagation + optimisme

Algorithme A*



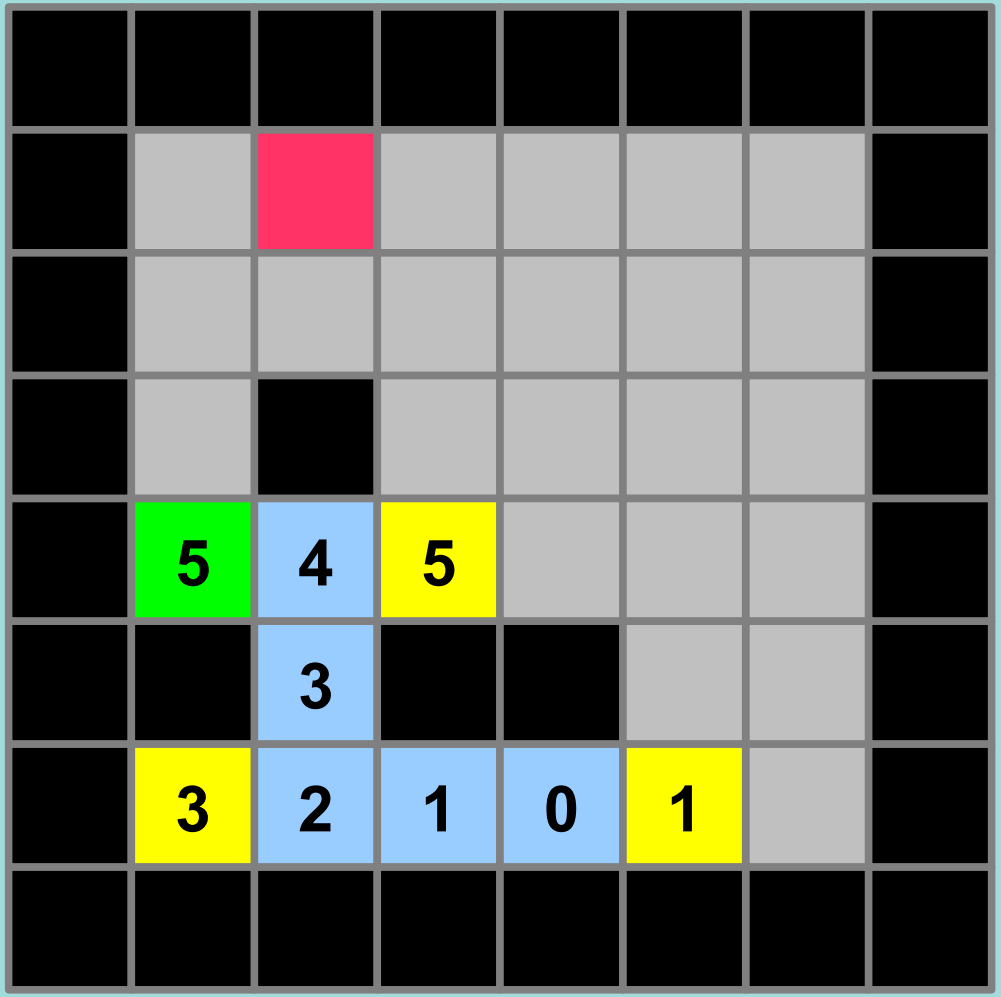
Algorithme de propagation + optimisme

Algorithme A*



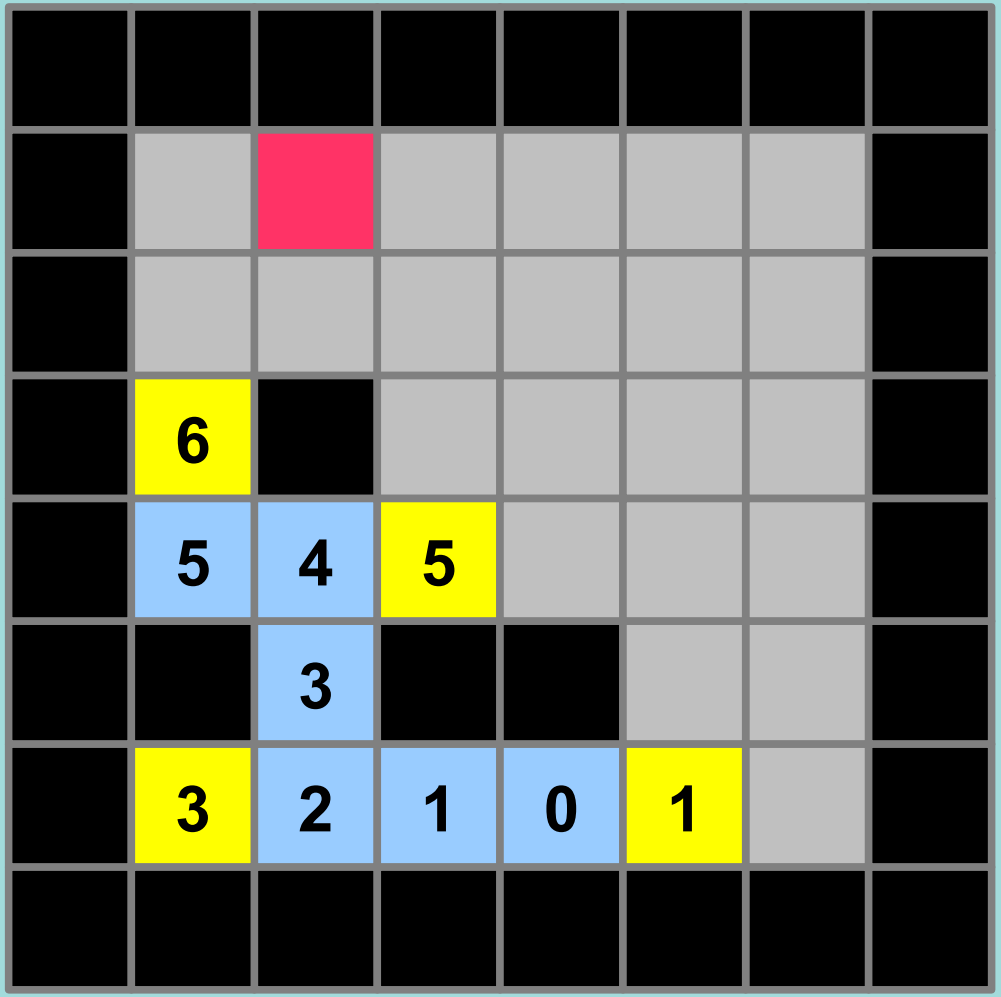
Algorithme de propagation + optimisme

Algorithme A*



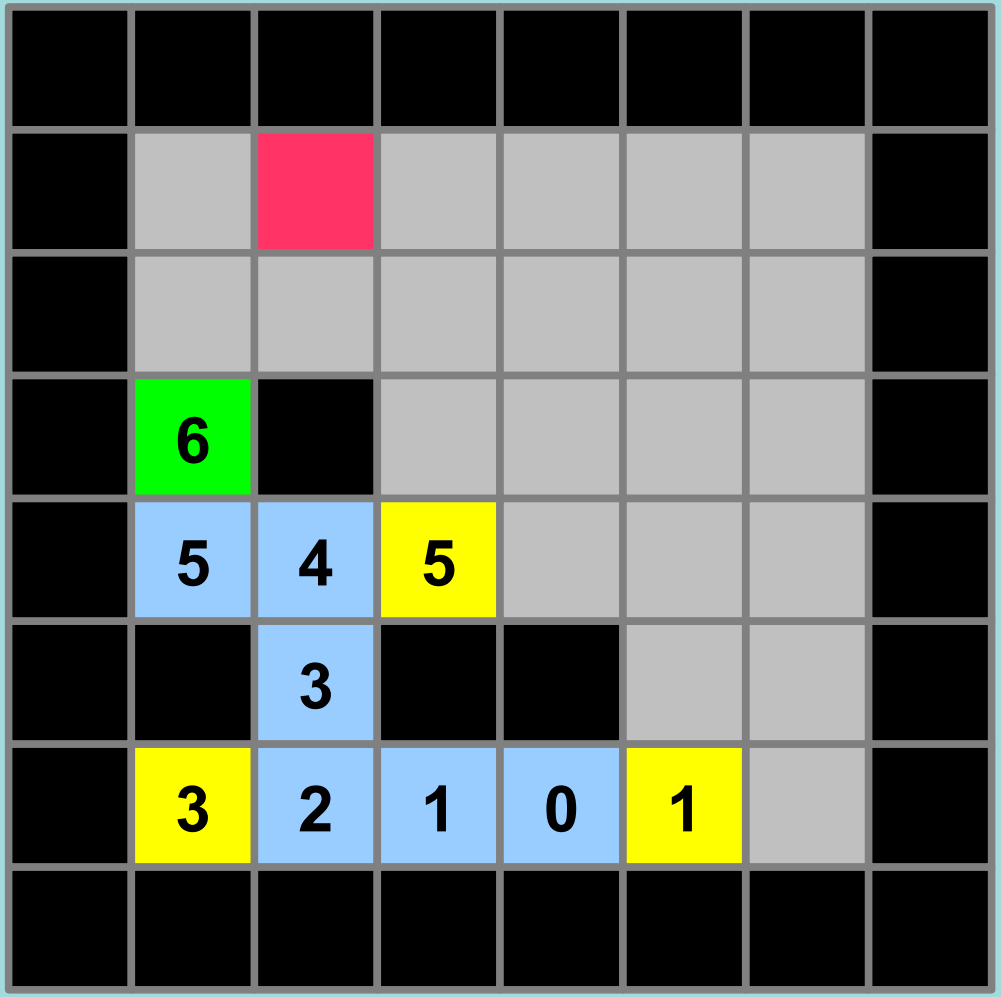
Algorithme de propagation + optimisme

Algorithme A*



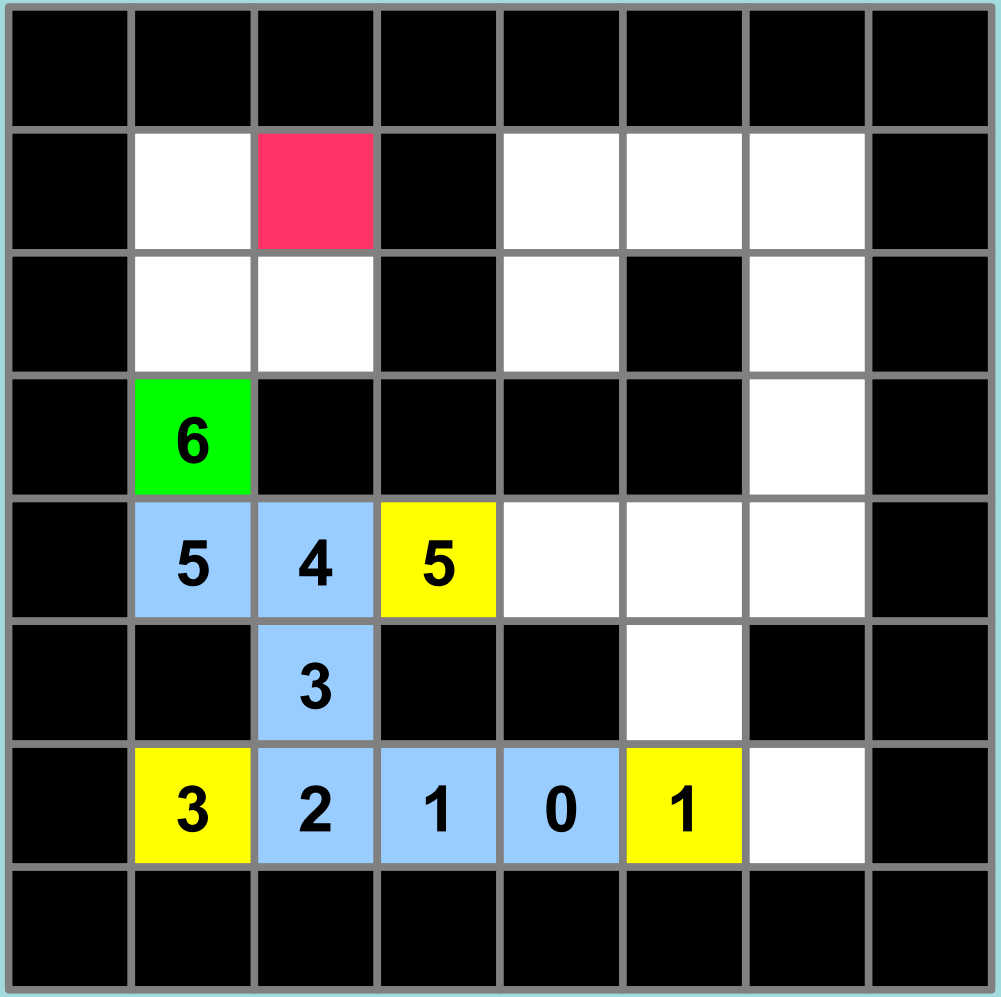
Algorithme de propagation + optimisme

Algorithme A*

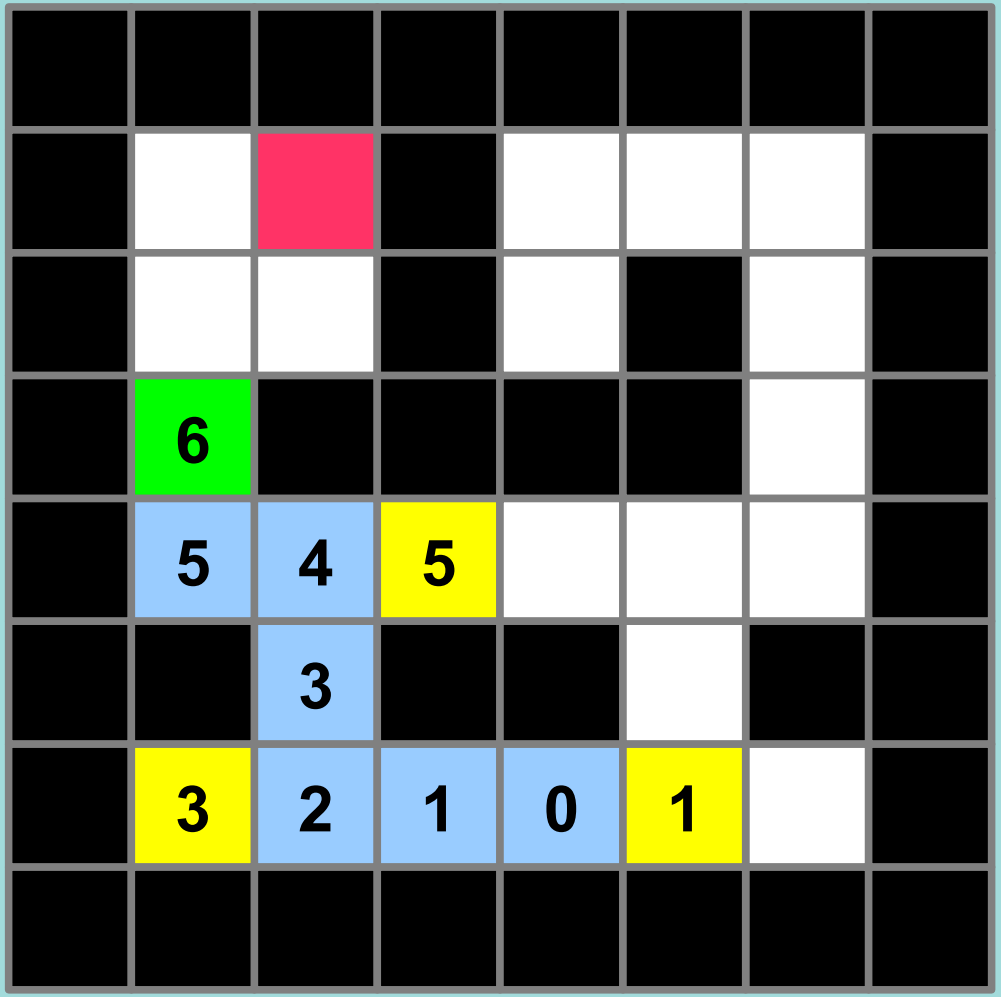


Algorithme de propagation + optimisme

Algorithme A*

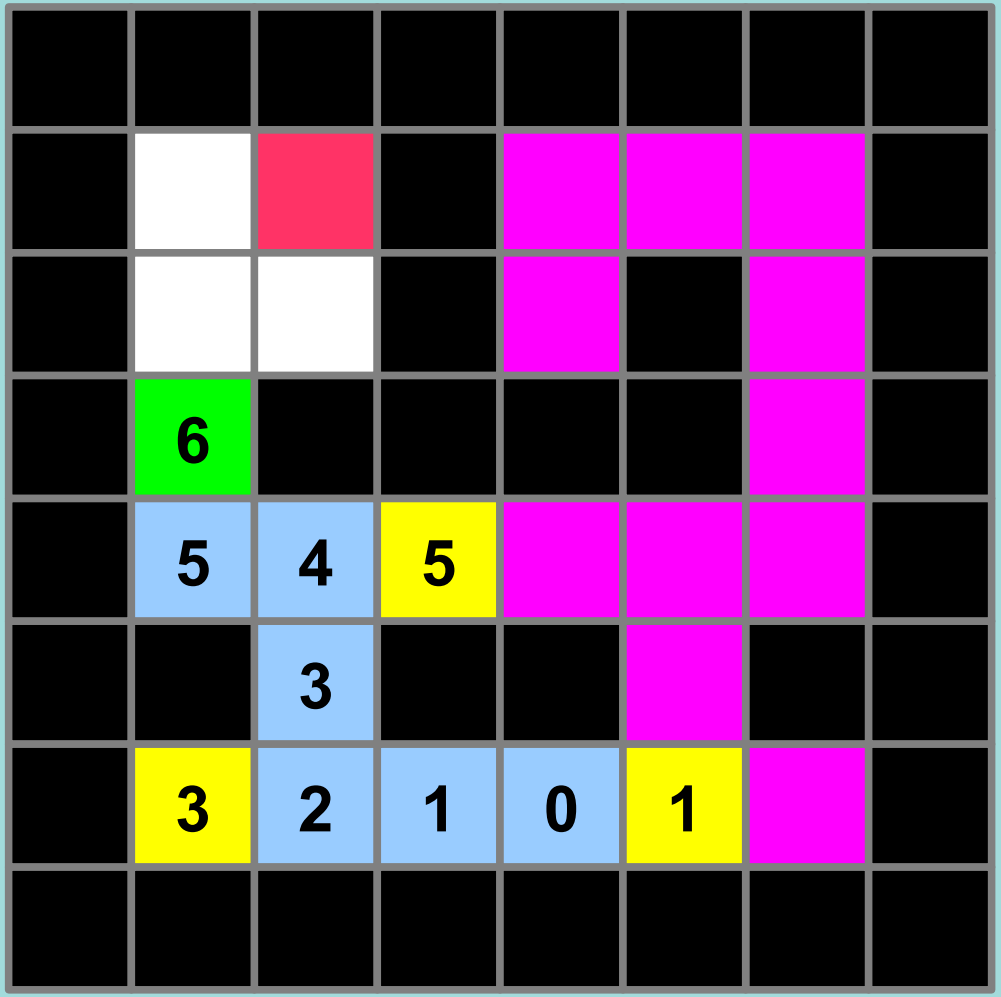


Algorithme A*



Conclusion ?

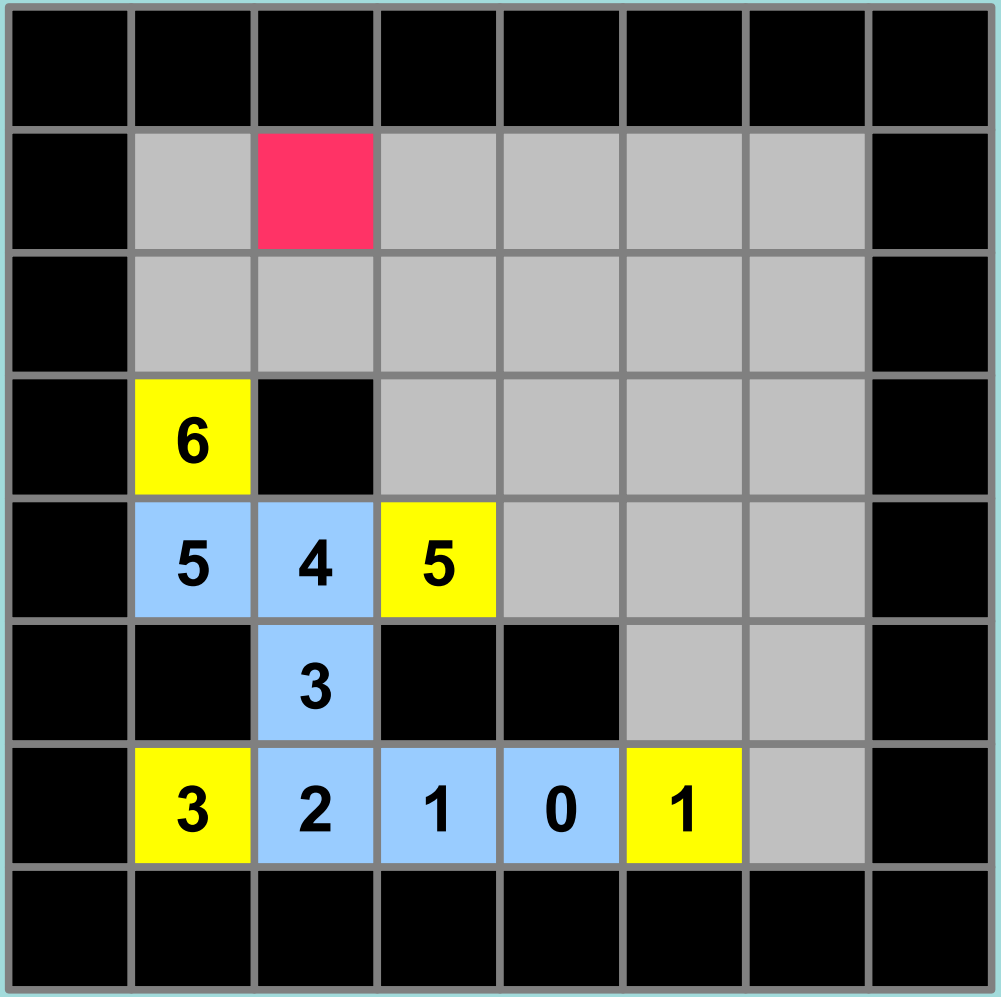
Algorithme A*



Conclusion ?

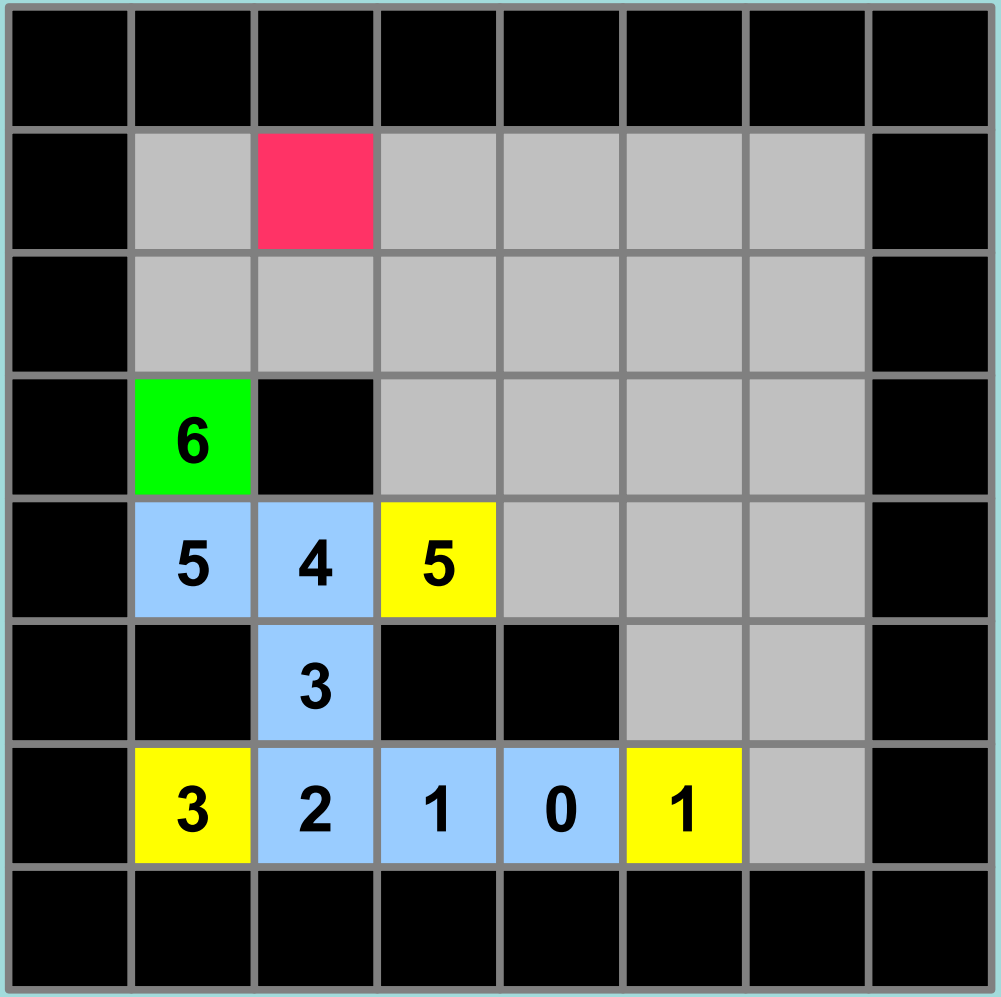
Changement labyrinthe

Changement labyrinthe



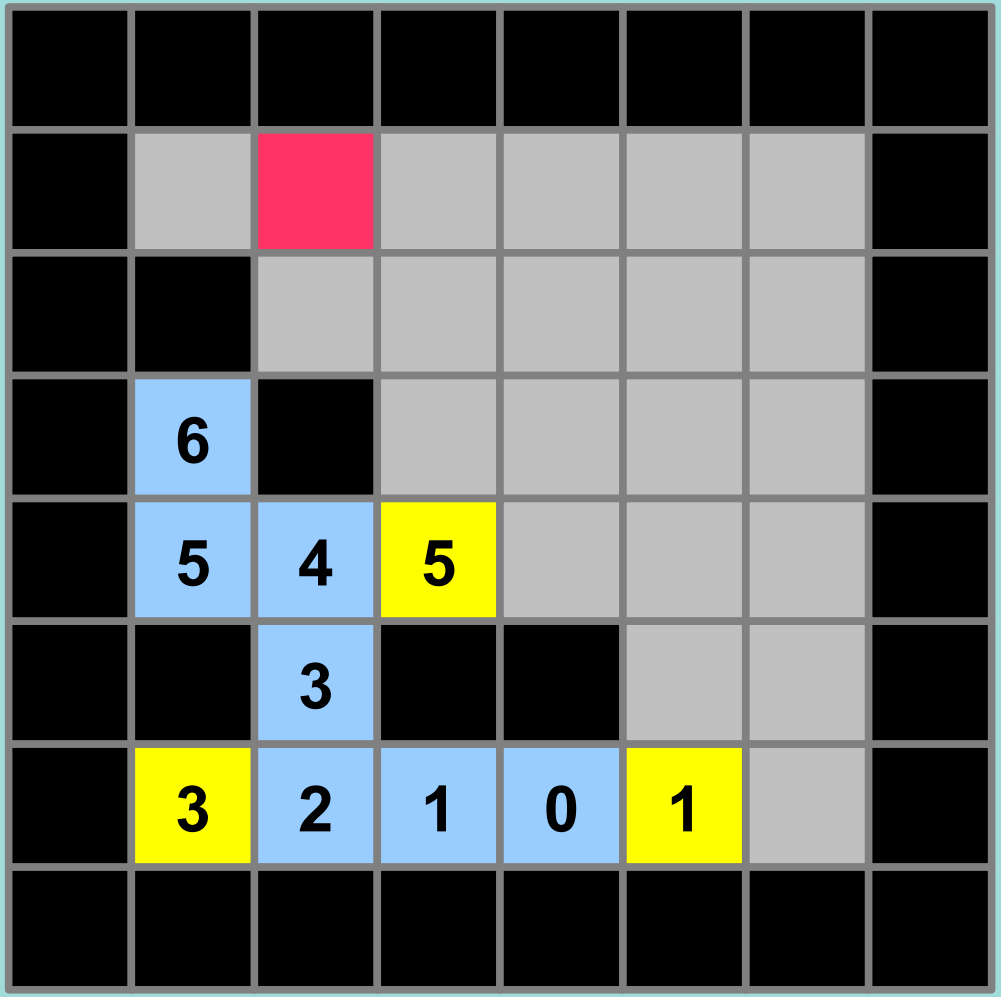
Algorithme de propagation + optimisme

Algorithme A*



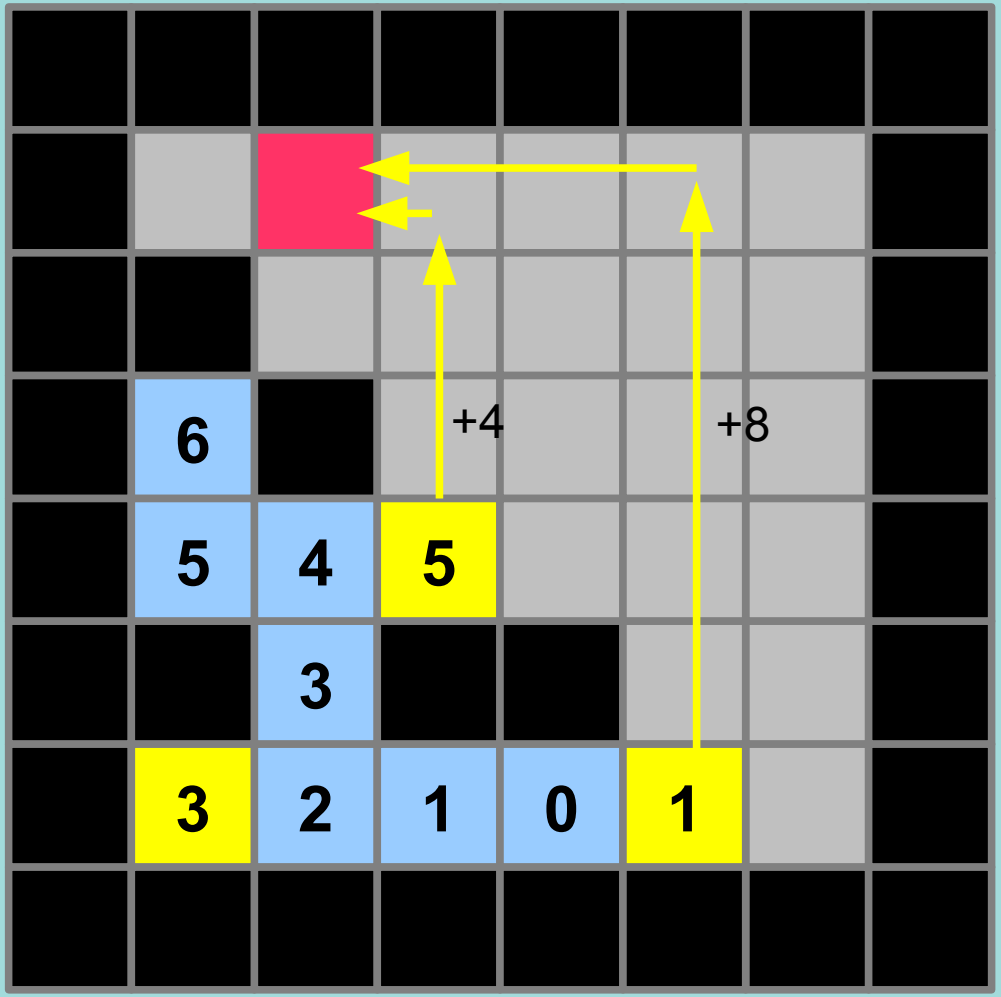
Algorithme de propagation + optimisme

Algorithme A*



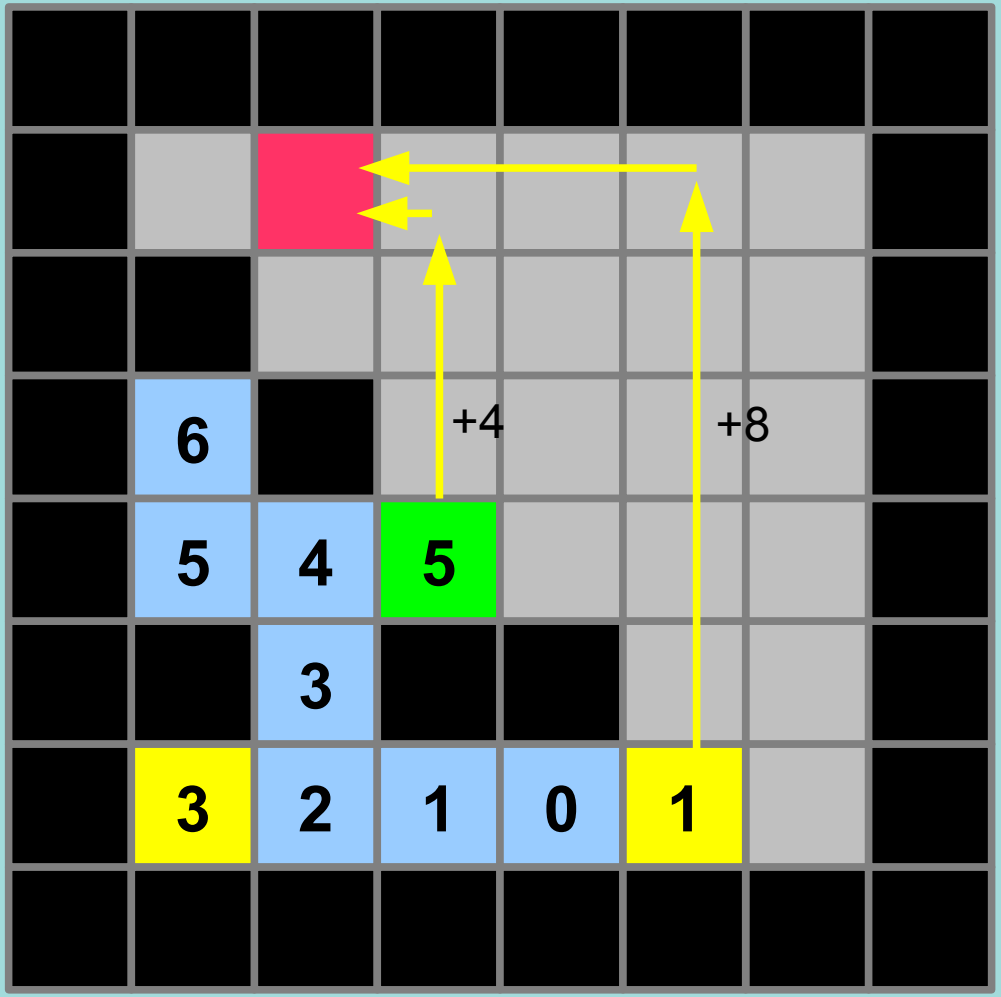
Algorithme de propagation + optimisme

Algorithme A*



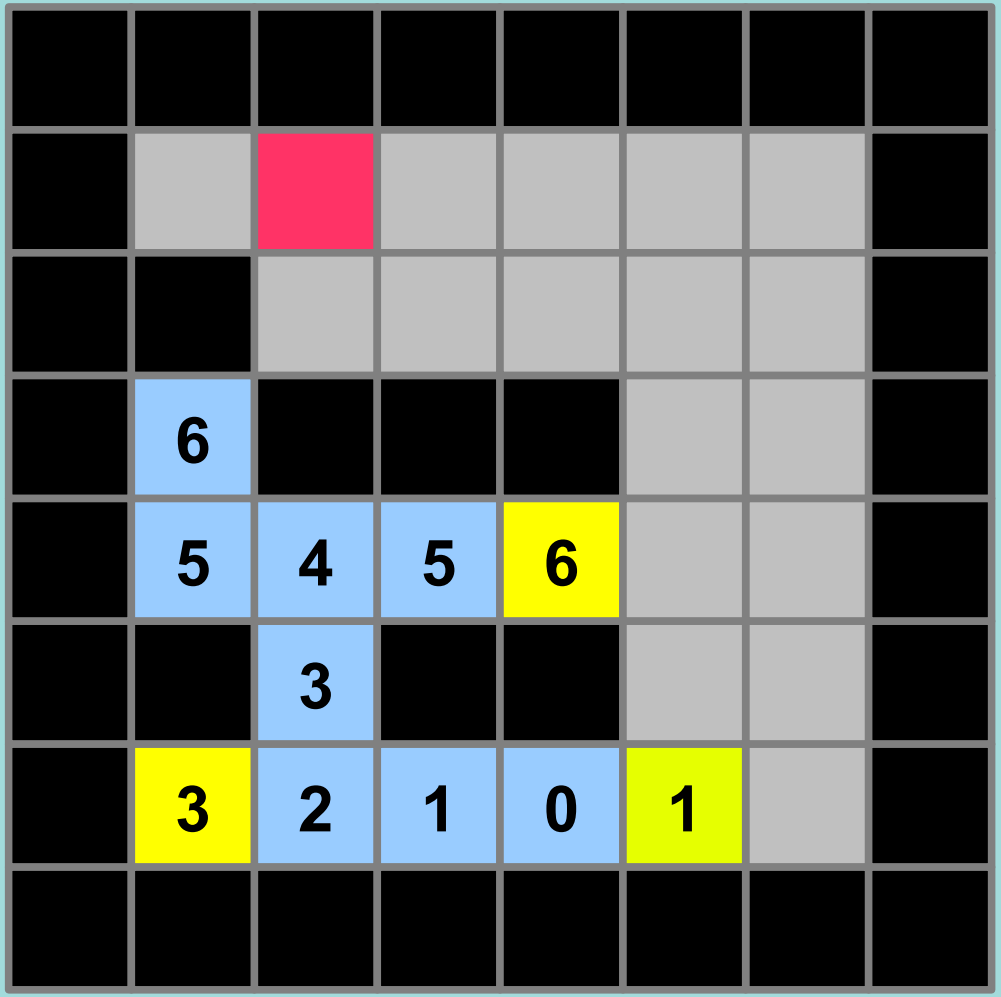
Algorithme de propagation + optimisme

Algorithme A*



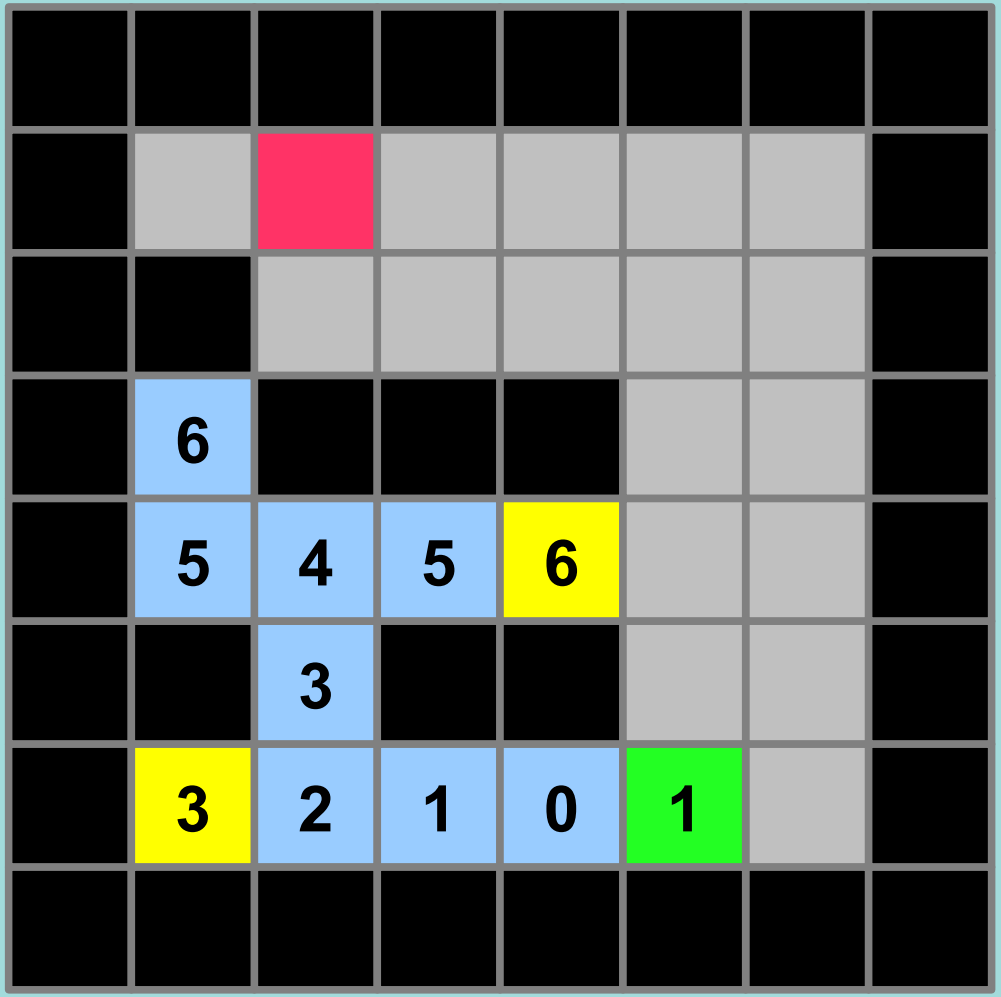
Algorithme de propagation + optimisme

Algorithme A*



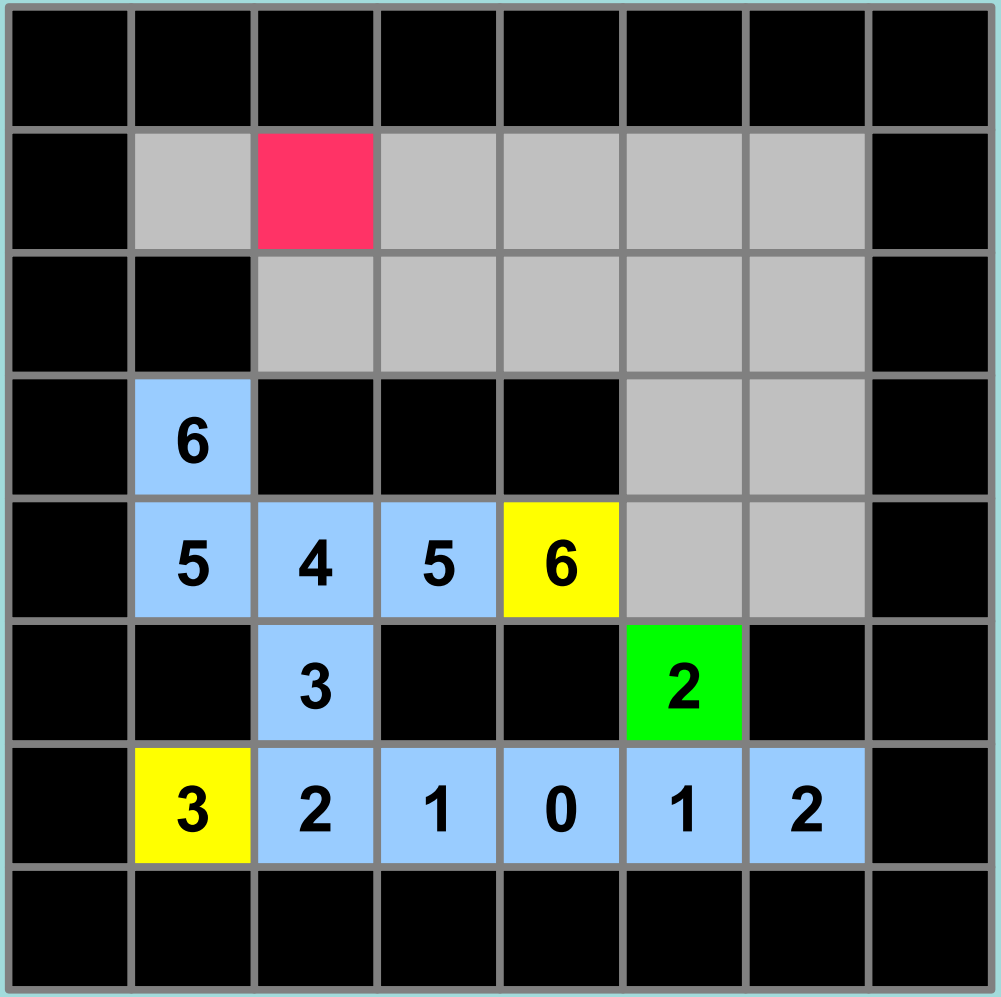
Algorithme de propagation + optimisme

Algorithme A*



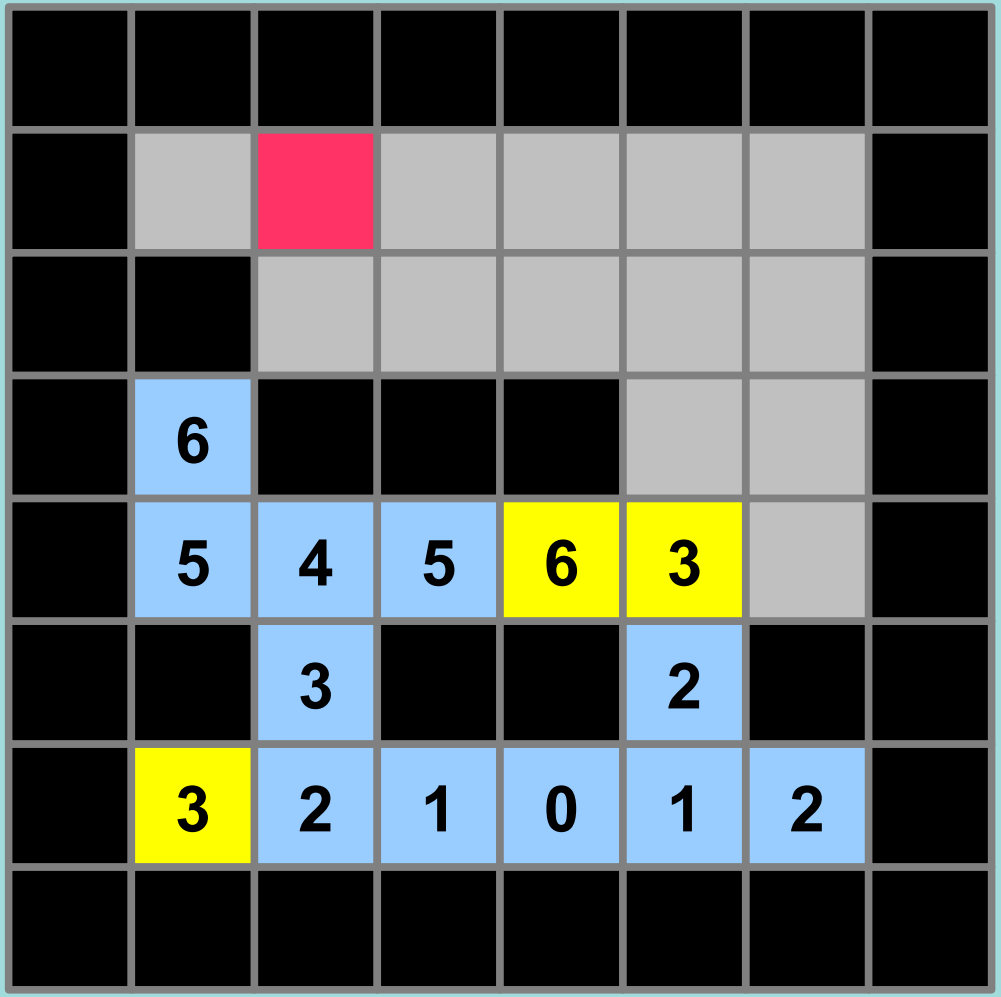
Algorithme de propagation + optimisme

Algorithme A*



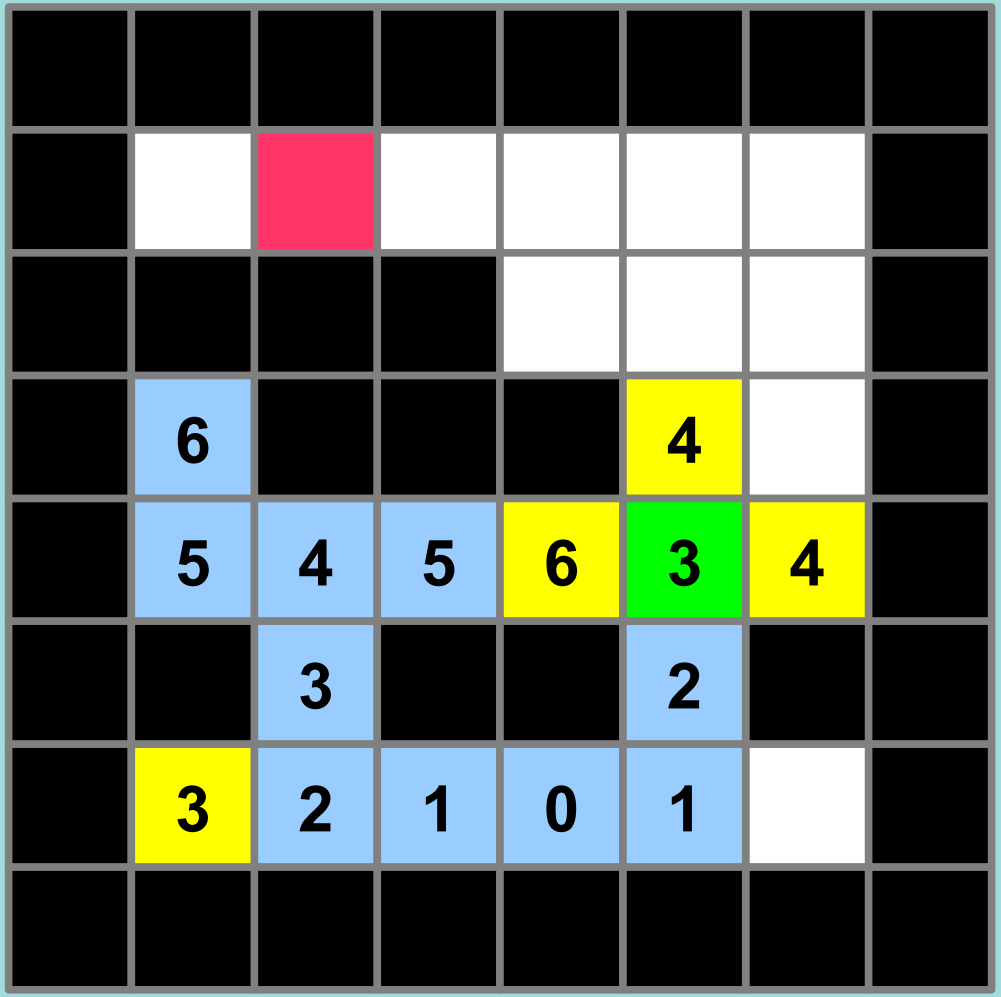
Algorithme de propagation + optimisme

Algorithme A*



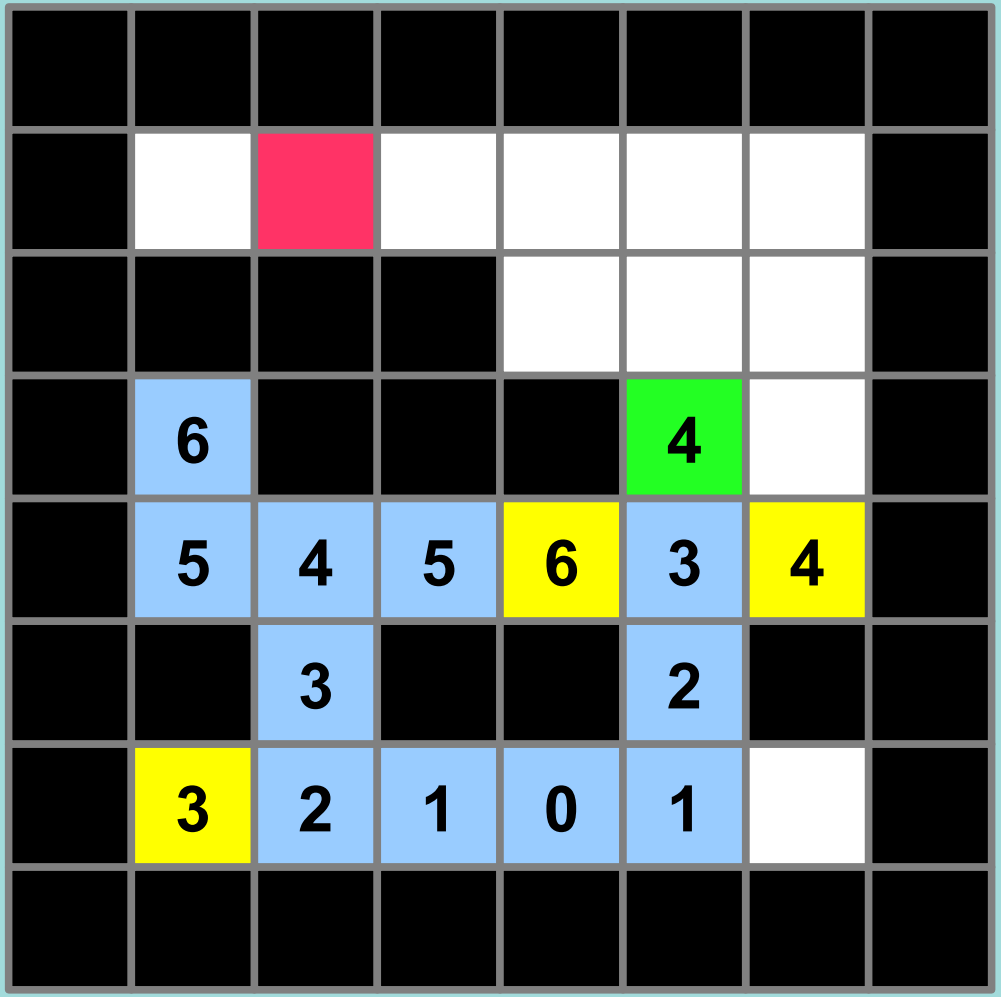
Algorithme de propagation + optimisme

Algorithme A*



Algorithme de propagation + optimisme

Algorithme A*



Algorithme de propagation + optimisme

Algorithme A*

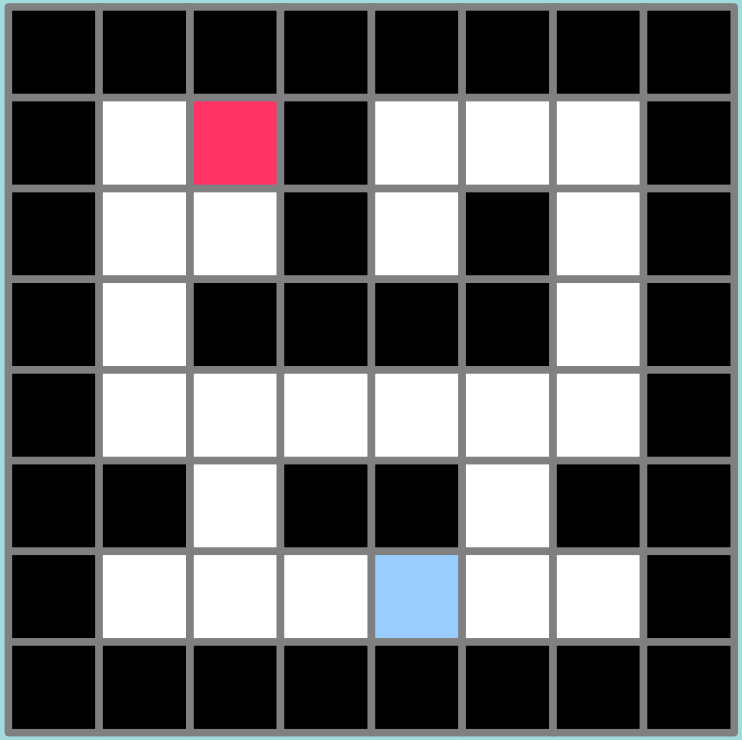
- Démo étudiant DUT

3 Extensions

- Ajoute des couts aux arcs
  - Dijkstra
- Accélérer la recherche
  - Algorithme A*
- Etendre à d'autres cadres
  - Théorie des graphes

Problème labyrinthe

- Structure particulière
  - Cases côtes à côtes
  - Connexité fixée
  - Coûts de 1

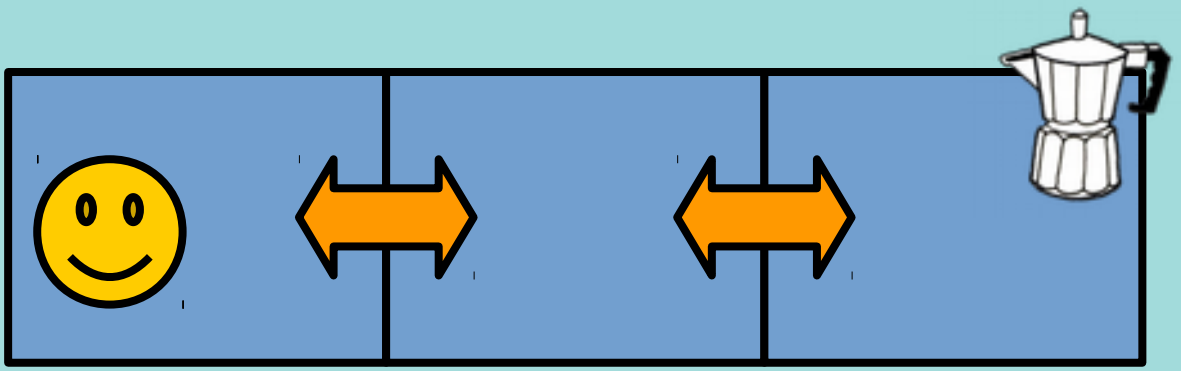


- De manière plus générale
  - Graphe (Noeuds + Arcs)

- Algorithmes identiques
  - Vraie question : Modélisation du probleme

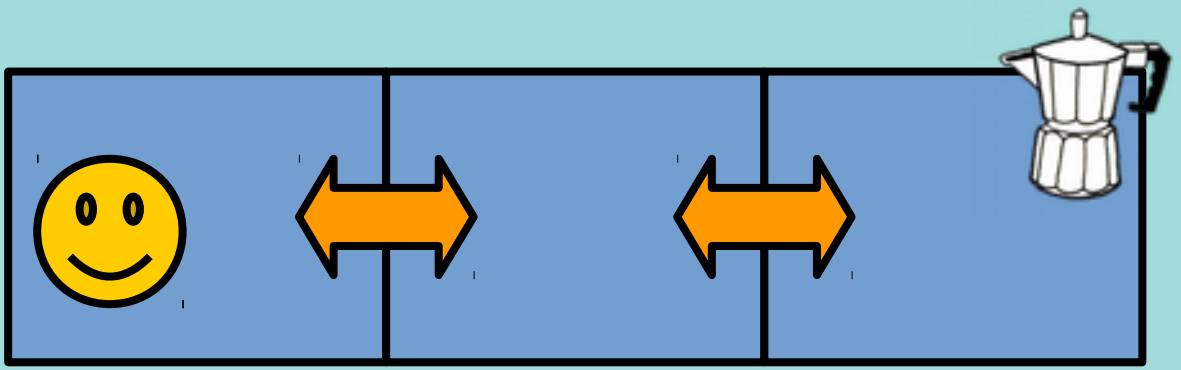
Problème

- Robot qui va chercher un café
  - 3 pièces qui se suivent
  - Cafetière au fond
  - Possibilité de déplacer, prendre, poser



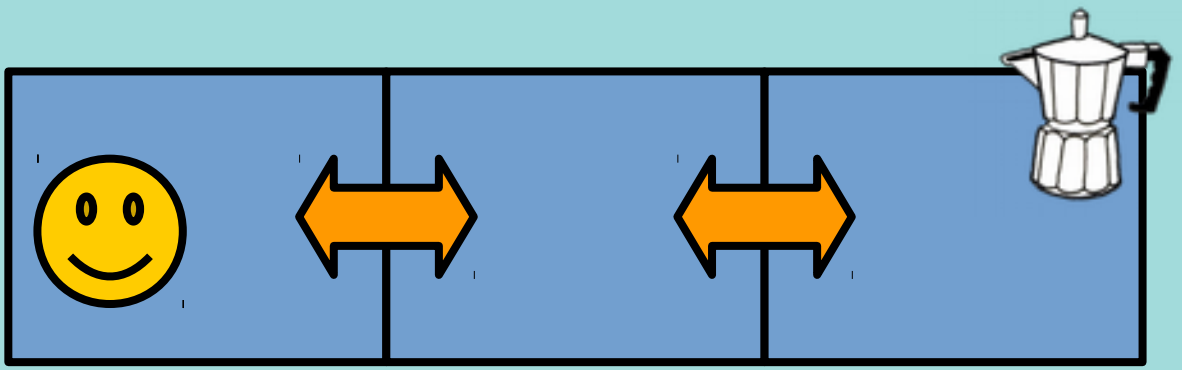
Problème

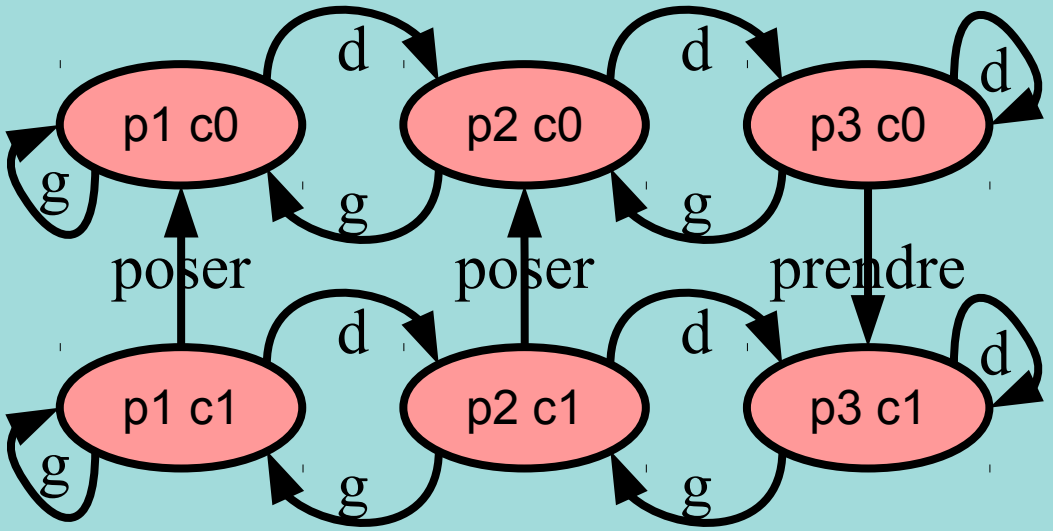
- Robot qui va chercher un café
  - Modélisation du problème



Problème

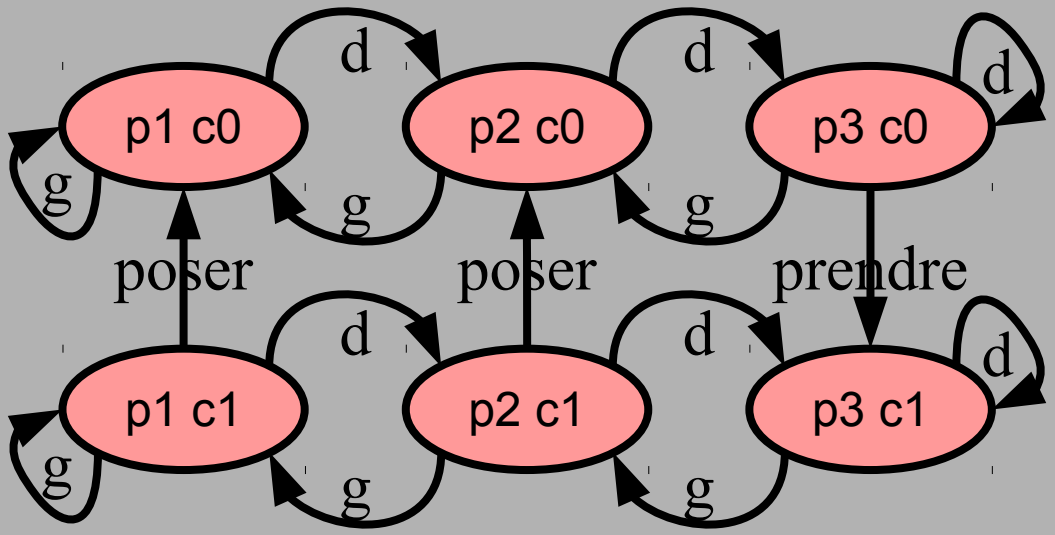
- Robot qui va chercher un café





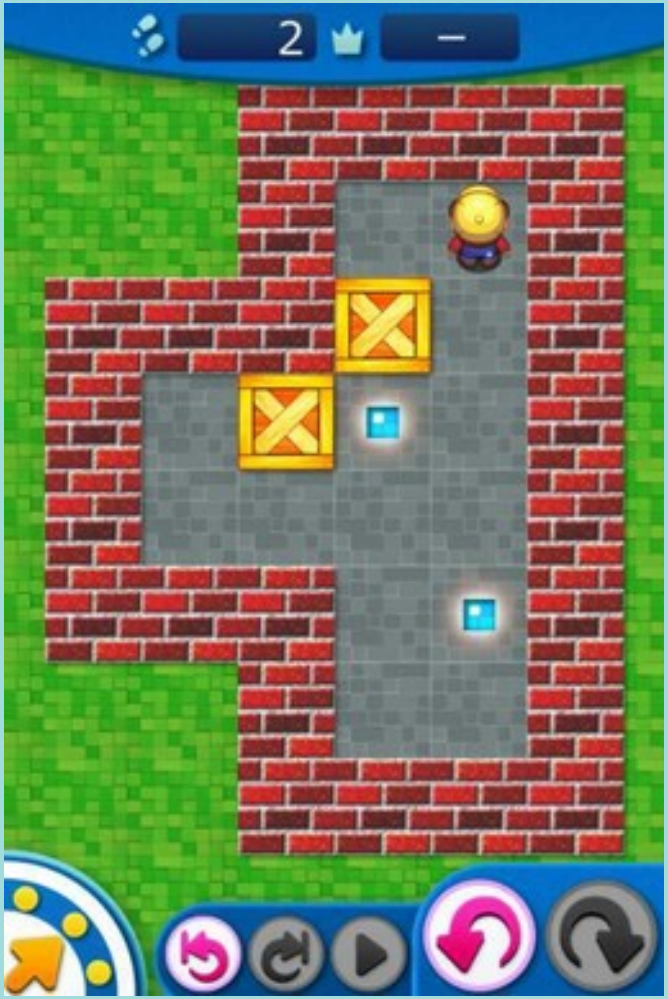
Problème

```
def actions():
    return(['gauche','droite','prendre','poser'])

def etats():
    return([(1,0),(1,1),(2,0),(2,1),(3,0),(3,1)])

def transition(s,a):
    (pos,cafe)=s

    if (a=='gauche'):
        if (pos>1):
            pos=pos-1

    if (a=='droite'):
        if (pos<3):
            pos=pos+1

    if (a=='prendre'):
        if (pos==3):
            Cafe=1

    if (a=='poser'):
        Cafe=0

return(pos,cafe)
```



Probleme : Sokoban

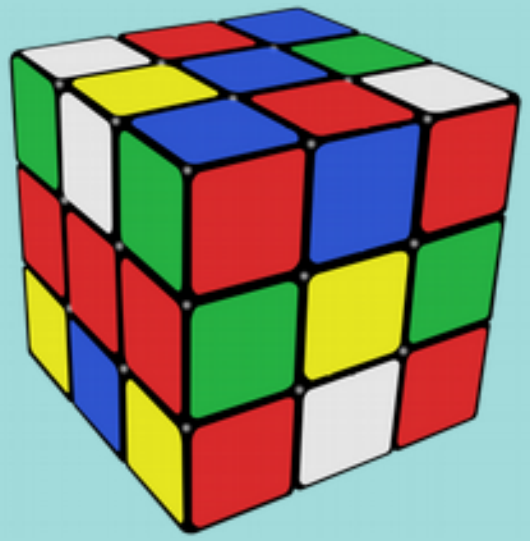
- Sokoban
  - Caisse et murs
  - Amener les caisses
- A faire
  - Modéliser le probleme
  - Appliquer algo
  - Heuristique si A*



Problèmes : puzzle solo

Taquin

Rush hour



Rubick's cube

Problèmes : puzzle solo

Dr Eureka (blue orange)

Et beaucoup d'autres

- Thinkfun / Smartgames

Turnstile