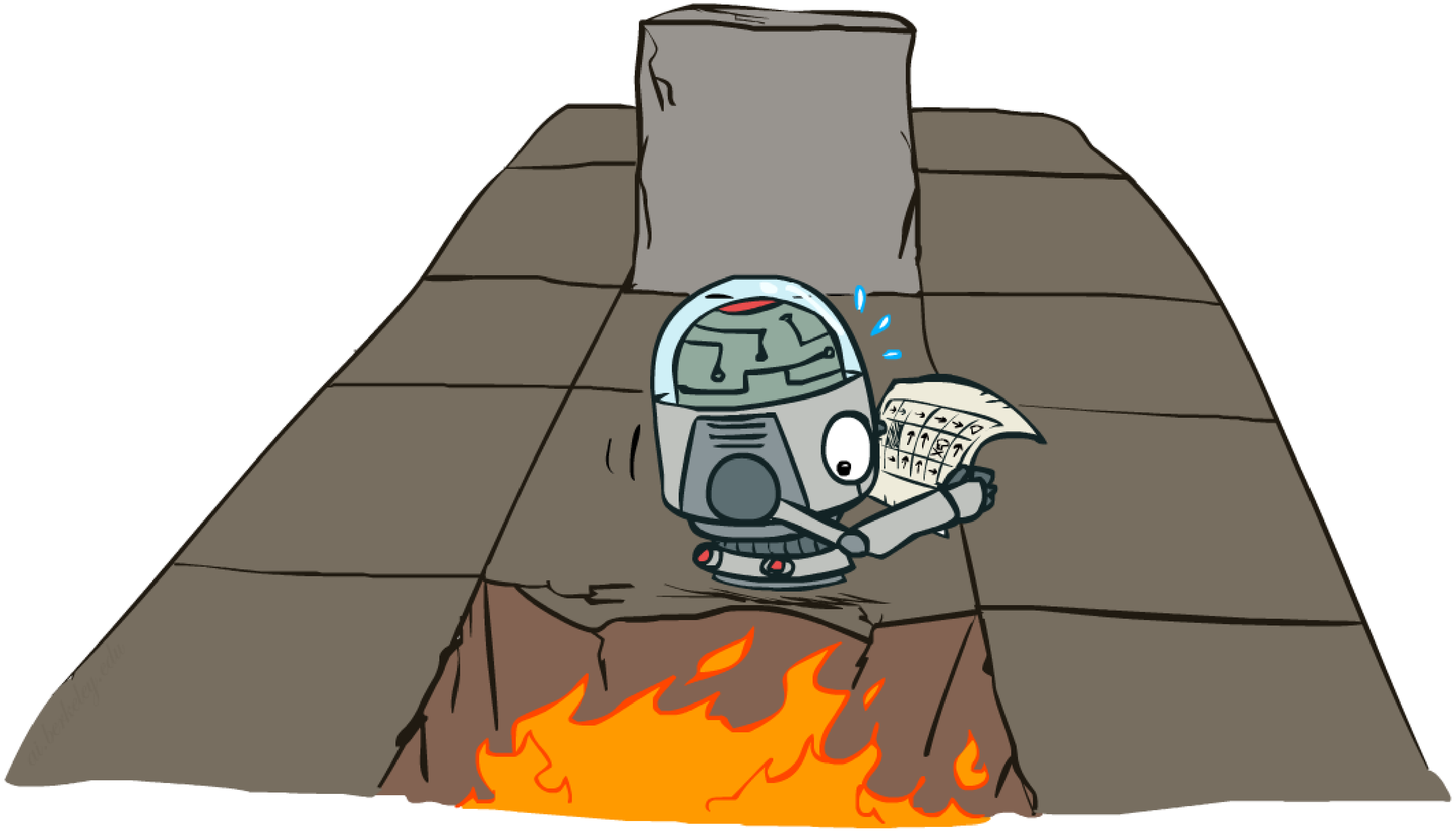
Intelligence Artificielle

Processus de Décision de Markov II

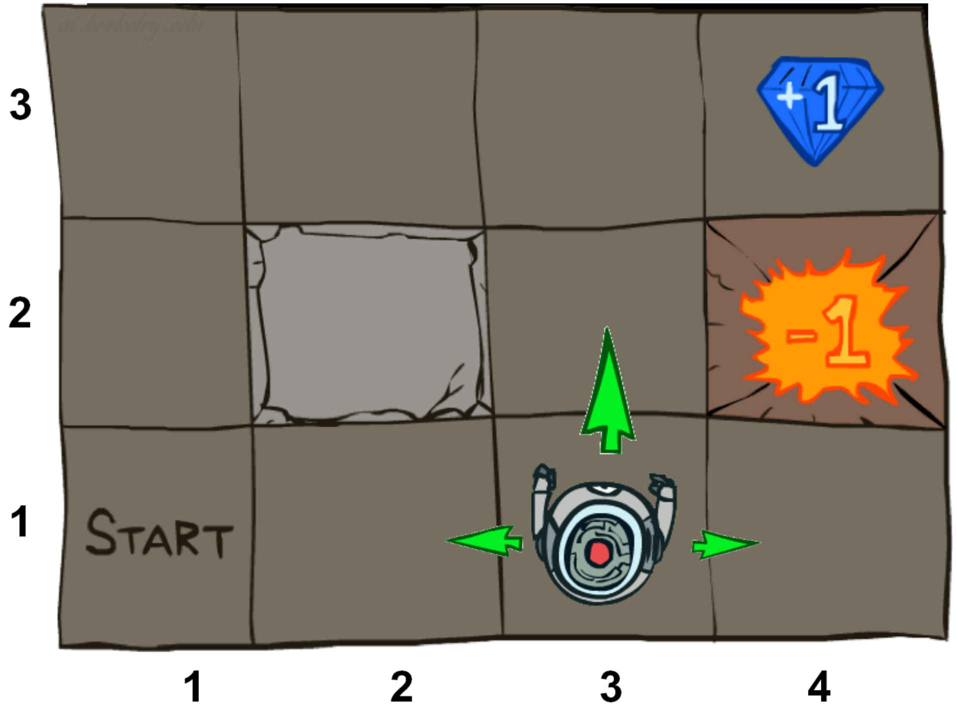


Prof: A.Belcaid --- Ecole Nationale des Sciences Appliquées , Fès

[Slides Crées par Dan Klein et Pieter Abbeel pour le cours CS188 Intro to AI à UC Berkeley]

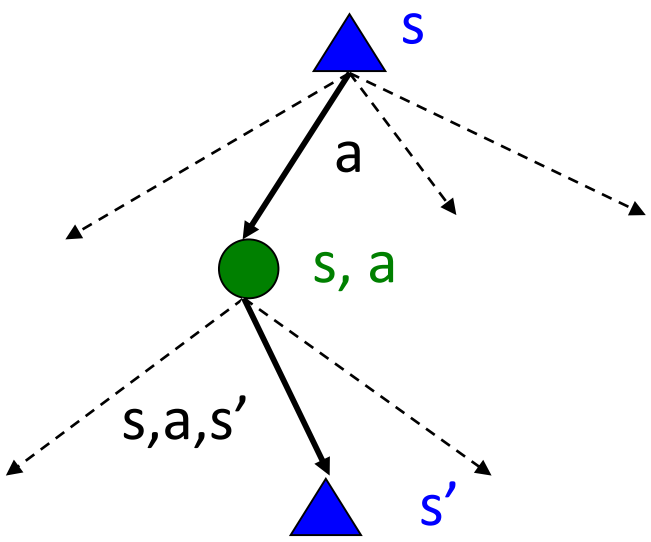
Exemple: Monde Grille

- Un problème labyrinthe
  - L'agent vit dans la grille
  - Murs bloquent le chemin de l'agent.
- Déplacement bruité: Actions ne mènnet pas toujours à leur destination.
  - 80%, l'action '*North*' mène l'agent au **Nord**.
  - 10%, *North* le mène à l'**EST** et 10% **Ouest**
  - Si il y a un mur bloquant son chemin, l'agent garde sa position.
- L'agent recoit une recompense à chaque itération
  - Petite "living" recompense à chaque iteration (peut être négative)
  - Grande recompense à la fin
- Objectif: maximiser la somme des recompense avec remise



Récapitulation: MDPs

- Processus de decision de Markov :
  - Etats S
  - Actions A
  - Transitions P(s'|s,a) (or T(s,a,s'))
  - Récompense R(s,a,s') (et remise γ)
  - Etat de départ s_0



- Elements:
  - Stratégie = Associe chaque état à une action
  - Utilité = Somme des recompense avec remise
  - Valeurs = Espérance de l'utilités d'un noeud max.
  - Valeurs-Q = Espérance de l'utilité d'une état Q (noeud de chance)

Quantités optimales

- Valeur de l'utilité à l'état s:

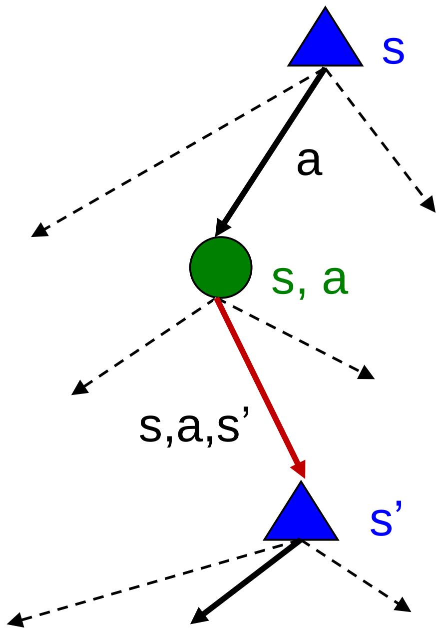
  $V^*(s)$ = espérance de l'utilité en commençant par s.

- Valeur d'un état q(s,a):

  $Q^*(s,a)$ = Espérance de l'utilité en commençcant par s en prennant l'action l'action a.

- La politique optimale:

  $\pi^*(s)$ = Action optimale pour l'état s

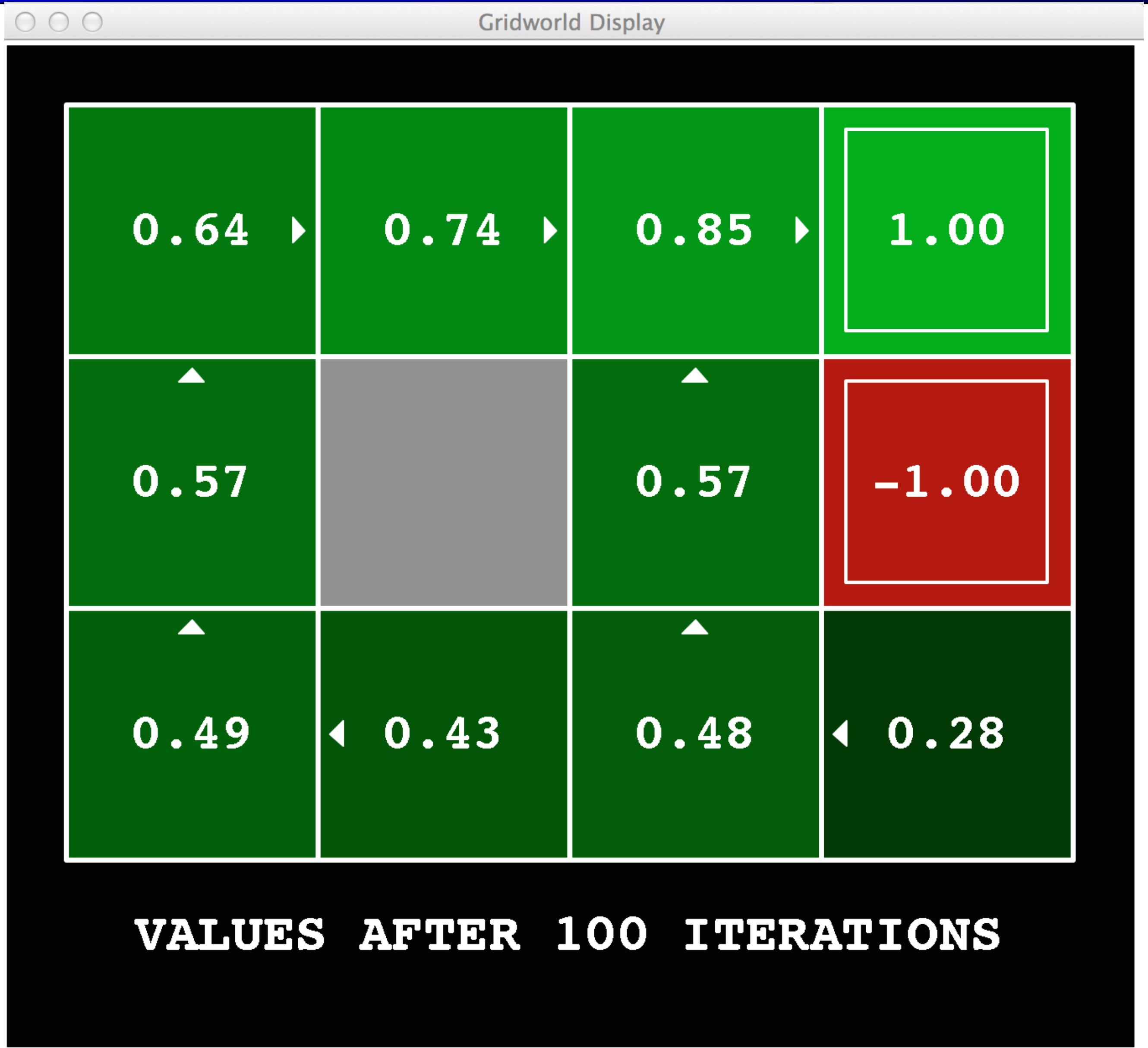


s is a *state*

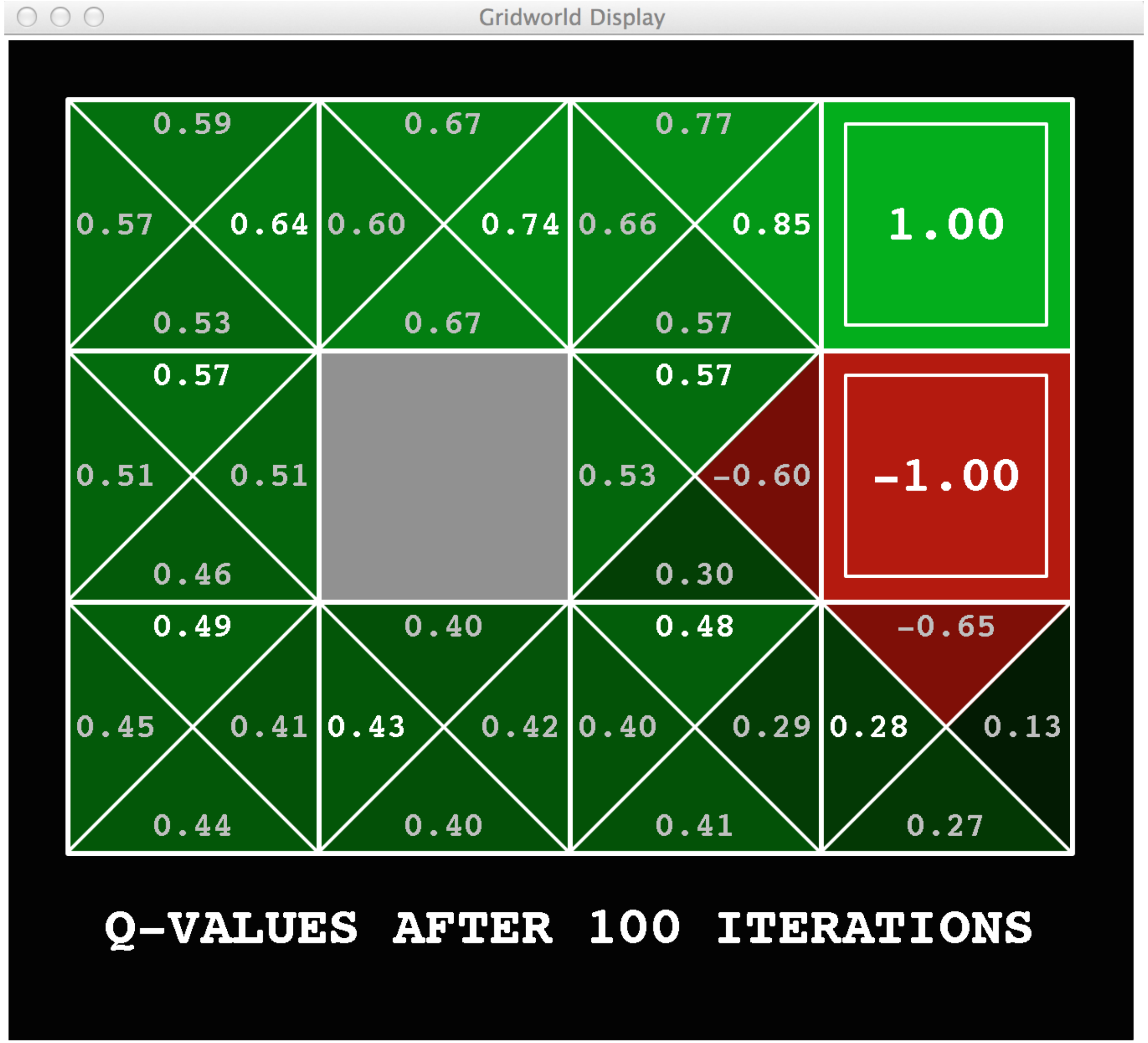
(s, a) is a *q-state*

(s,a,s') is a *transition*

Valeurs V*

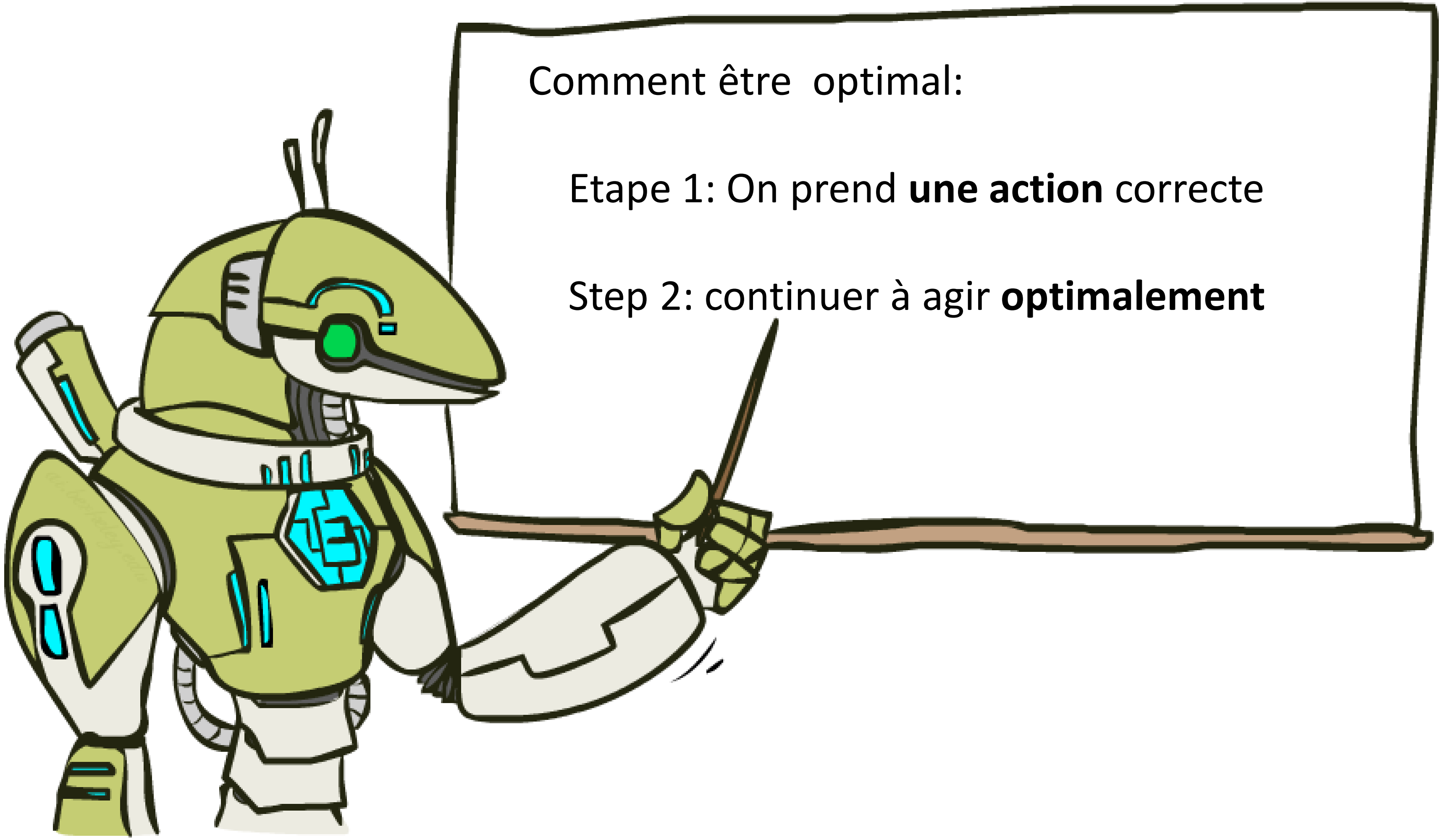

Gridworld Display
0.64
0.74
0.85
1.00
0.57
0.57
-1.00
0.49
0.43
0.48
0.28
VALUES AFTER 100 ITERATIONS

Q* Grille



Gridworld Display
0.59
0.57
0.64
0.53
0.67
0.60
0.74
0.67
0.77
0.66
0.85
0.57
1.00
0.57
0.51
0.51
0.46
0.57
0.53
-0.60
0.30
-1.00
0.49
0.45
0.41
0.44
0.40
0.43
0.42
0.40
0.48
0.40
0.29
0.41
-0.65
0.28
0.13
0.27
Q-VALUES AFTER 100 ITERATIONS

Les équations de Bellman



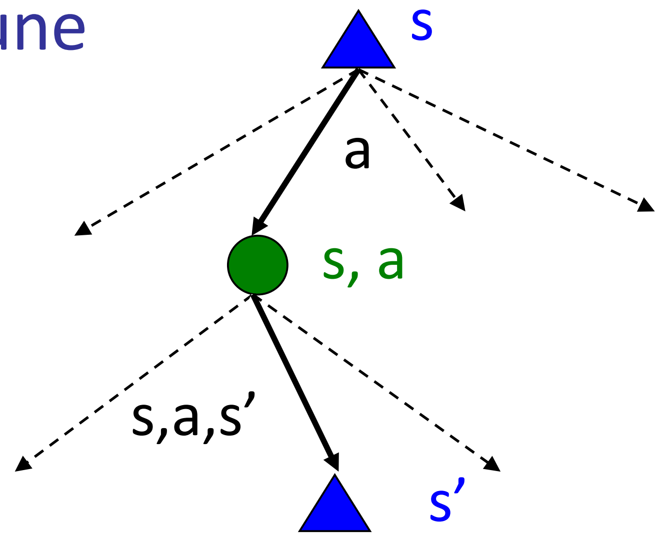
Les Equations de Bellman

- La définition de la fonction d'utilité optimale donne une relation de recurrence entre les valeurs optimales.

$$V^*(s) = \max_a Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s')\right]$$

$$V^*(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s')\right]$$



- Ces équations définnissent des équations de Bellman.

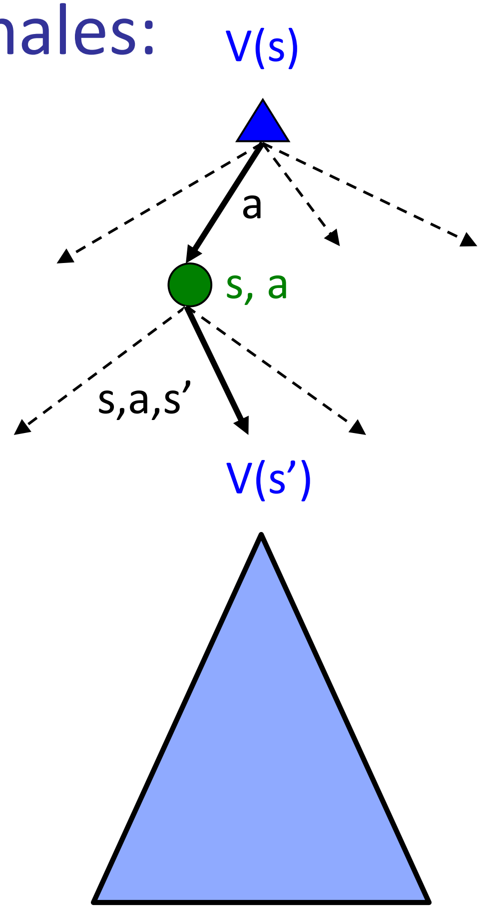
Itération de la valeur

- Les équations de Bellman caractérisent les valeurs optimales:

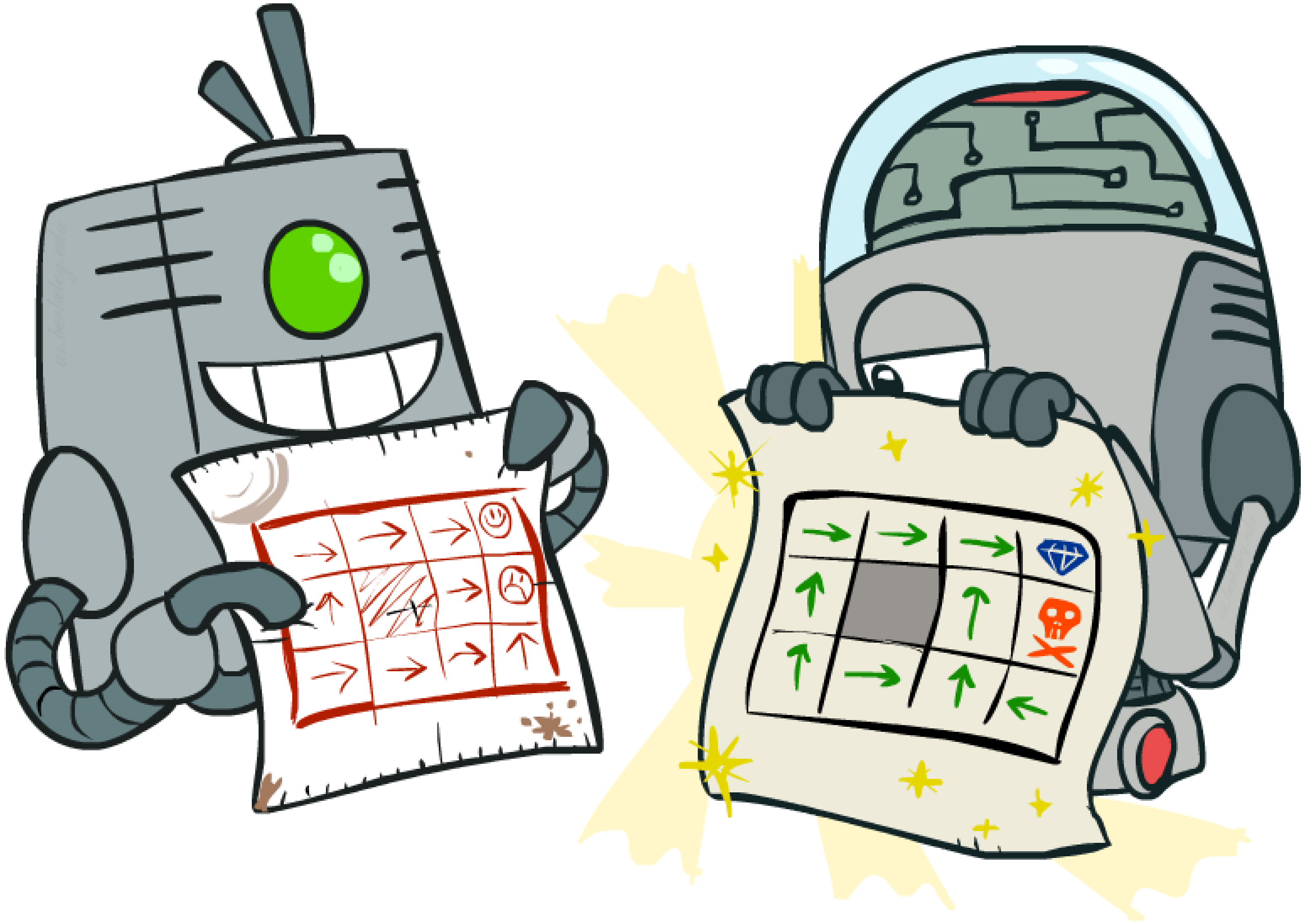
$$V^*(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s')\right]$$

- Un itération de la valeur calcule ces valeurs:

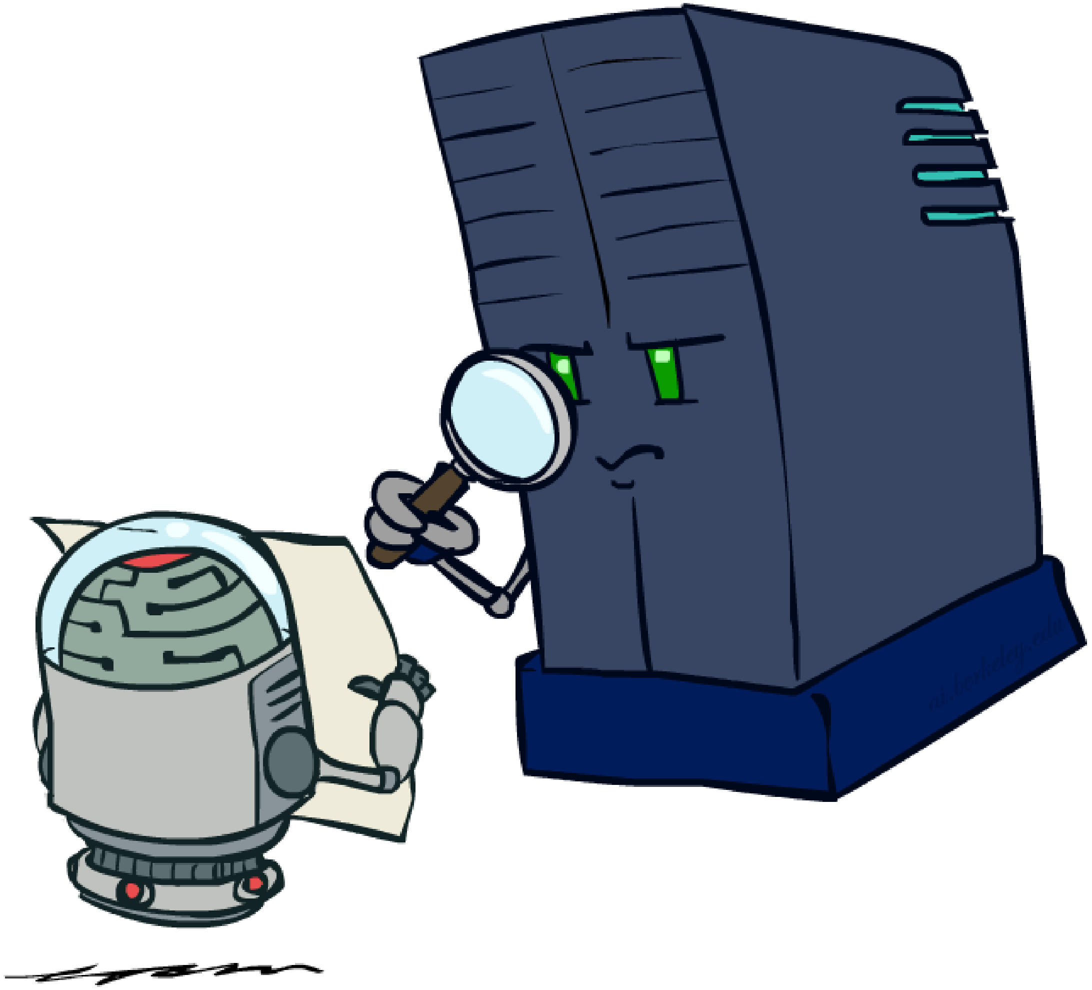
$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s')\right]$$



Méthodes pour les stratégies



Evaluation d’une stratégie



Stratégie fixée

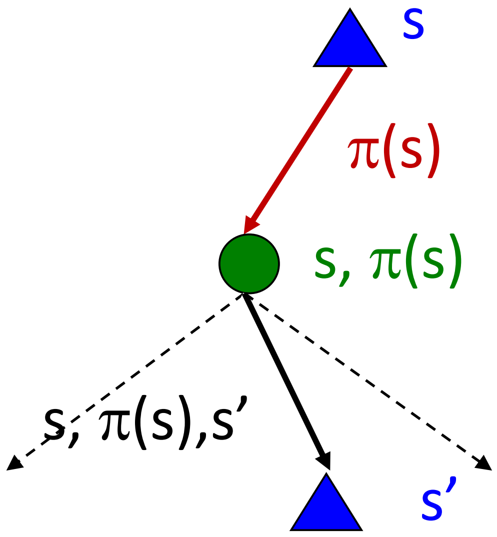
Strétégie optimale

Stratégie fixée π

- L'arbre de Expectimax calcule le max sur **toutes** les actions/
- Pour une stratégie fixée π(s), l'arbre serait plus simple car on considère une seule action par état.

Utilités pour une stratégie fixée

- Une autre operation basique consiste ca calculer l'utilité d'un état selon une stratégie fixée (par forcément optimale).



- Definir de l'utilité d'un état s, selon une stratégie fixée π:

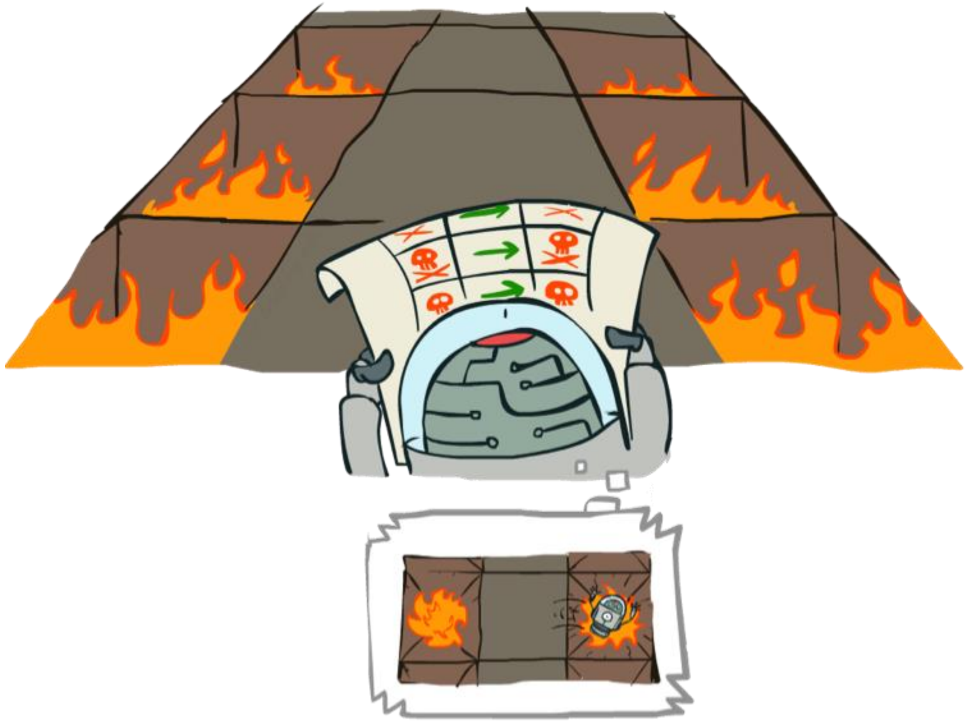
  $V^\pi(s)$ = Espérance de l'utilité en commencant par **s** et en suivant **π**

- Relation de recurrence:

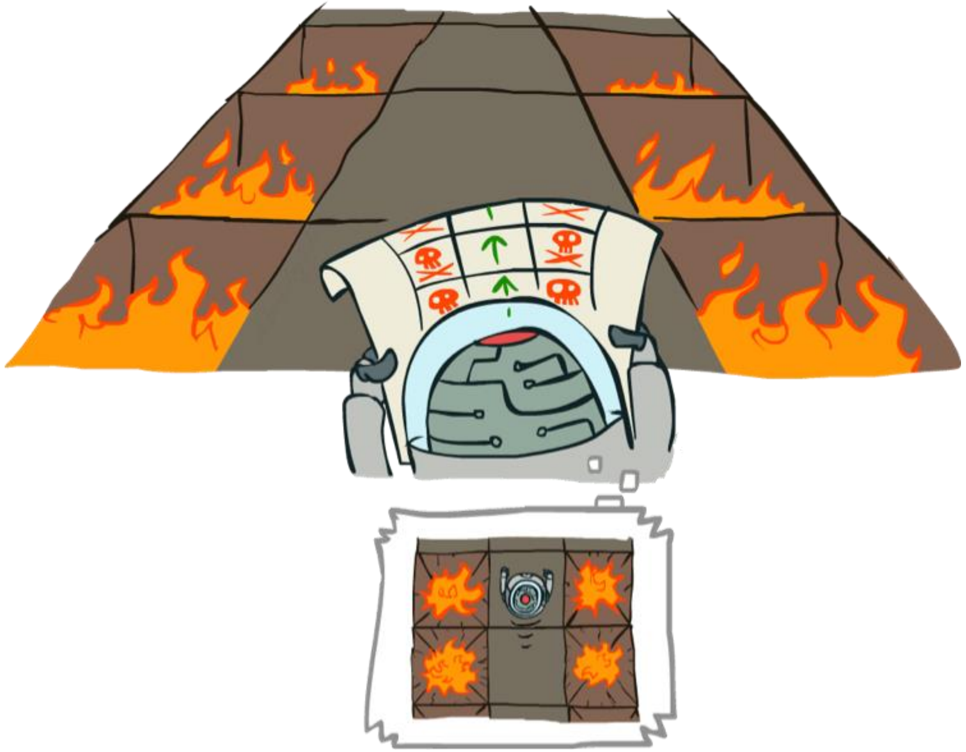
$$V^{\pi}(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^{\pi}(s')]$$

Exemple: Evaluation de la stratégie

Prendre toujours la droite



Allez toujours **devant**



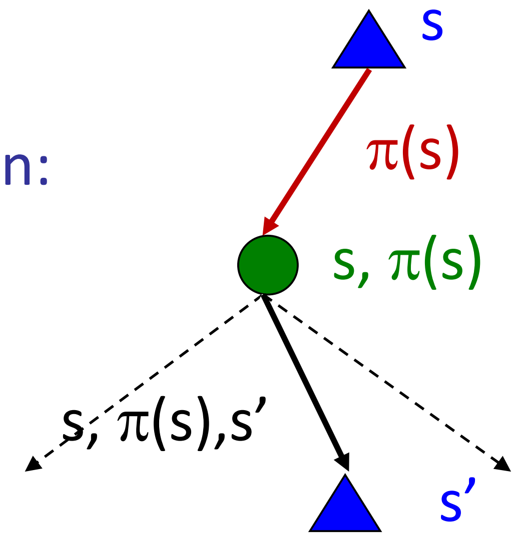
Exemple: Evaluation de la stratégie

Prendre toujours la **droite**

Allez toujours **devant**

Evaluation de la stratégie

- Comment peut on calculer les valeurs **V** pour une stratégie π?
- Idée 1: Convertir les relations recursives des équations de Bellman:

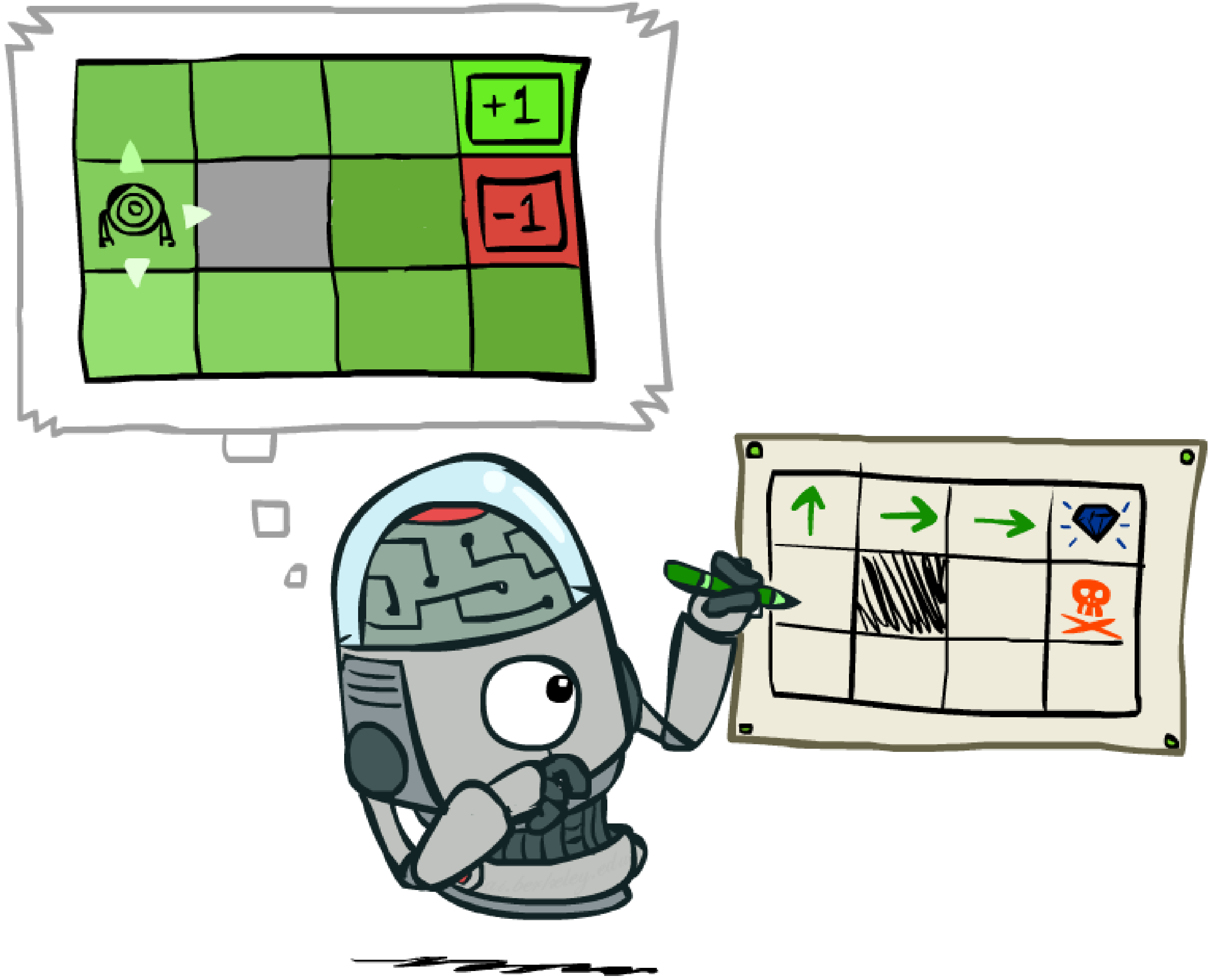


$$V_0^\pi(s) = 0$$

$$V_{k+1}^\pi(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_k^\pi(s')]$$

- Complexité: $O(S^2)$ par itération
- Idée 2: Sans les operations **Max**, les équations de Bellman donnent un **système linéaire.**

Extraction de la stratégie

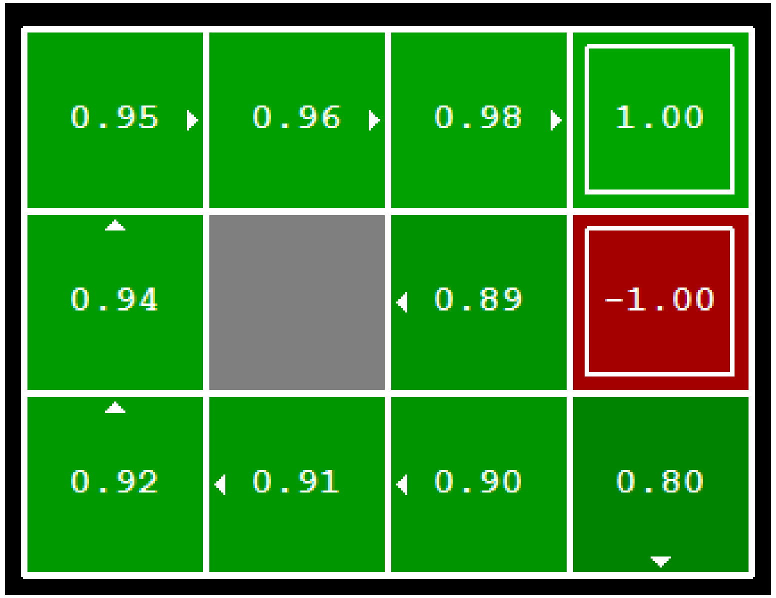


Calculer les actions à partir des valeurs

- Imaginons qu'on possède les valeurs V*(s)

- Comment doit on agir?
  - C'est pas direct!

- Il faut réaliser une itération mini-expectimax

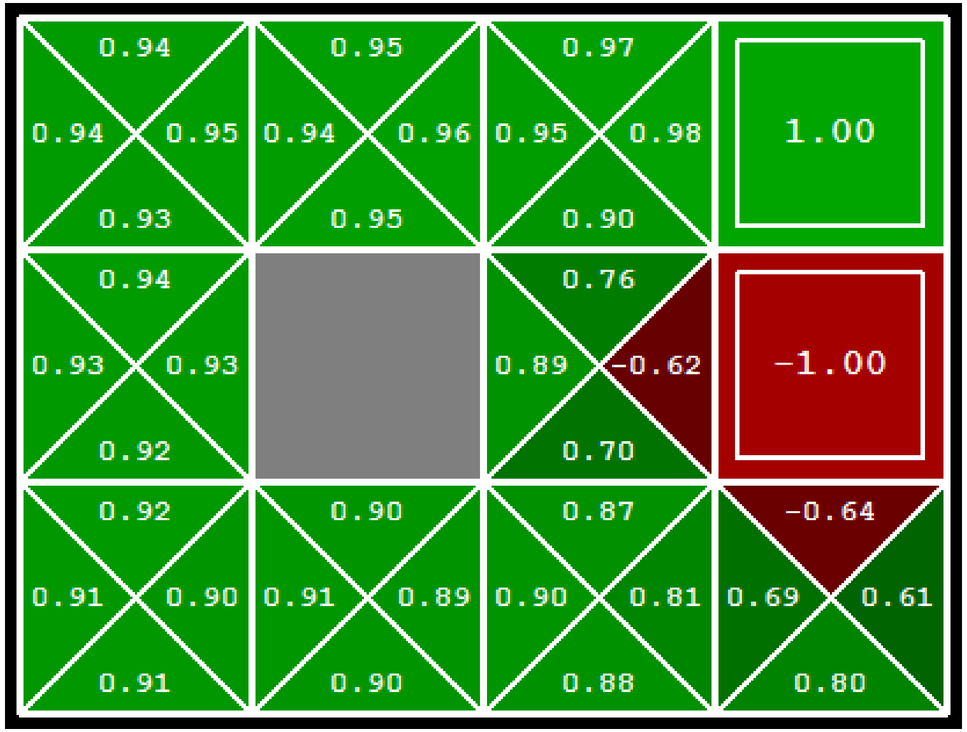


$$\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

- Cette operation est appellée extraction de la stratégie

Cacluler les actions à partir des valeurs Q

- Imaginons maintenant qu'on possède les les valeur Q:



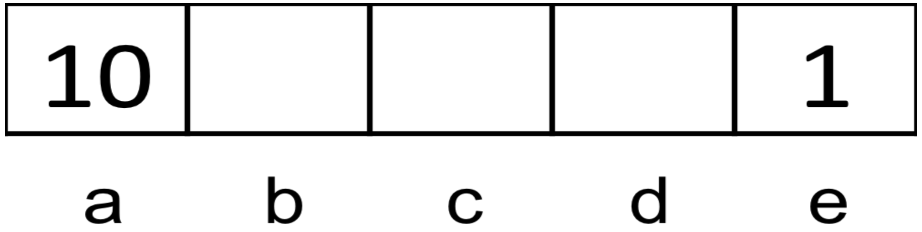
- Comment agir?
  - Décision naturelle!

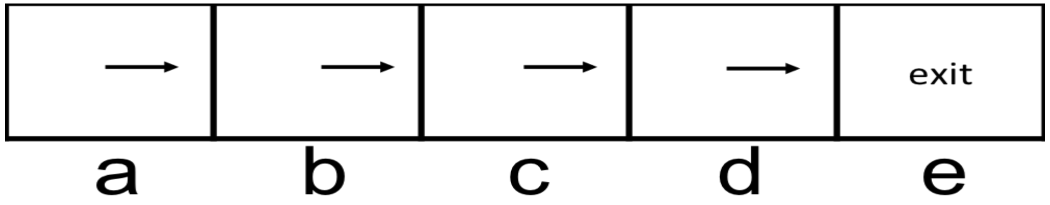
$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

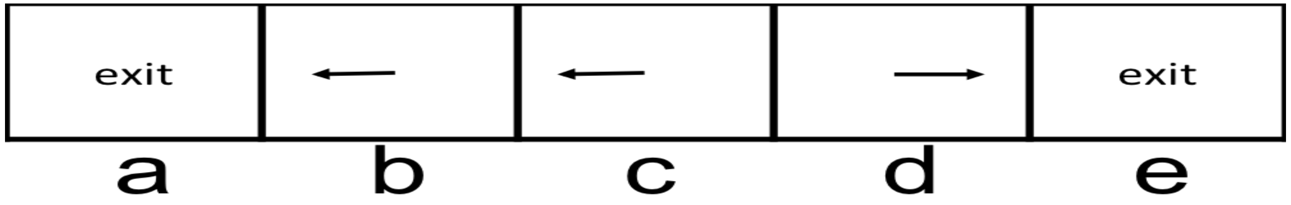
- Lesson importante: Les actions sont très simple à extraire des valeurs Q!

Quiz Evaluation de la stratégie

- On considère le monde grille, où on peut se déplacer vers les deux nœuds voisins. Toutes les actions sont réussies et on considère une remise $\gamma = 1$.

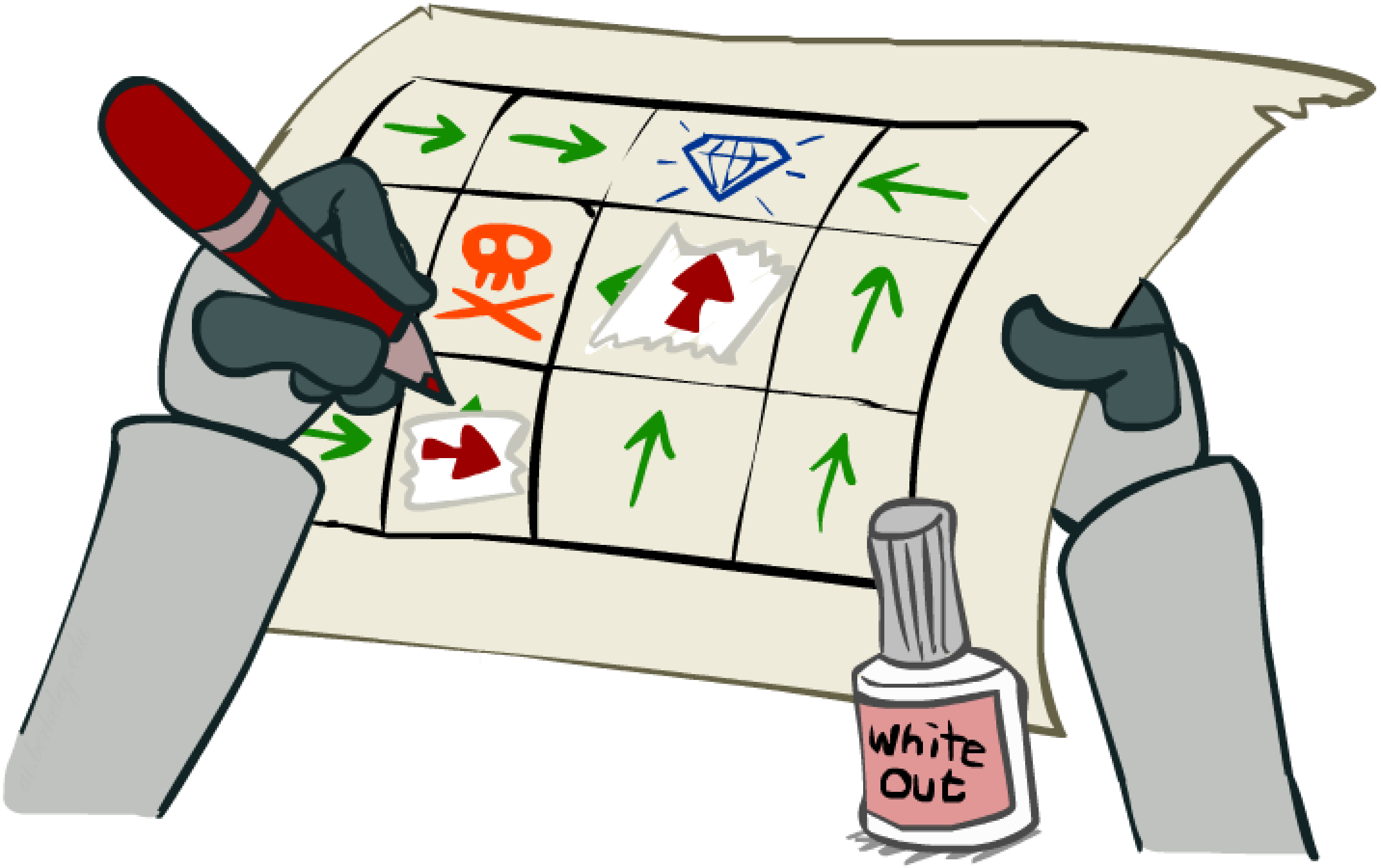






- Donner les valeurs de chaque nœud selon la stratégie π_1 présentée à gauche.
- Donner les valeurs de chaque nœud selon la stratégie π_2 présentée à droite.

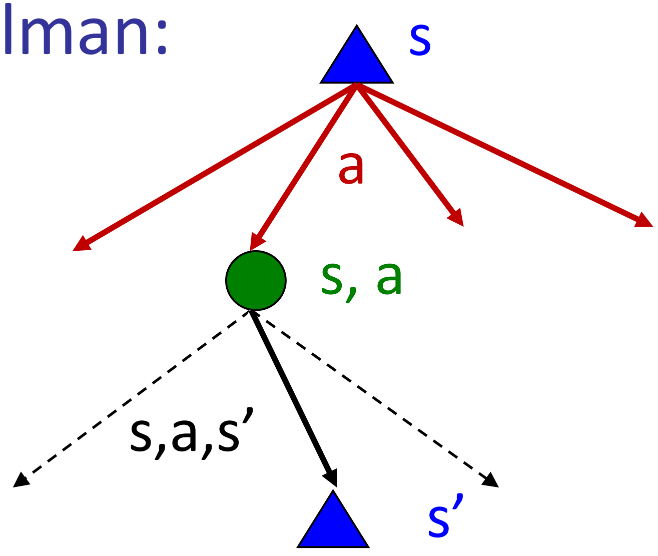
Itération de la stratégie



Problèmes de l'iteration de la valeur

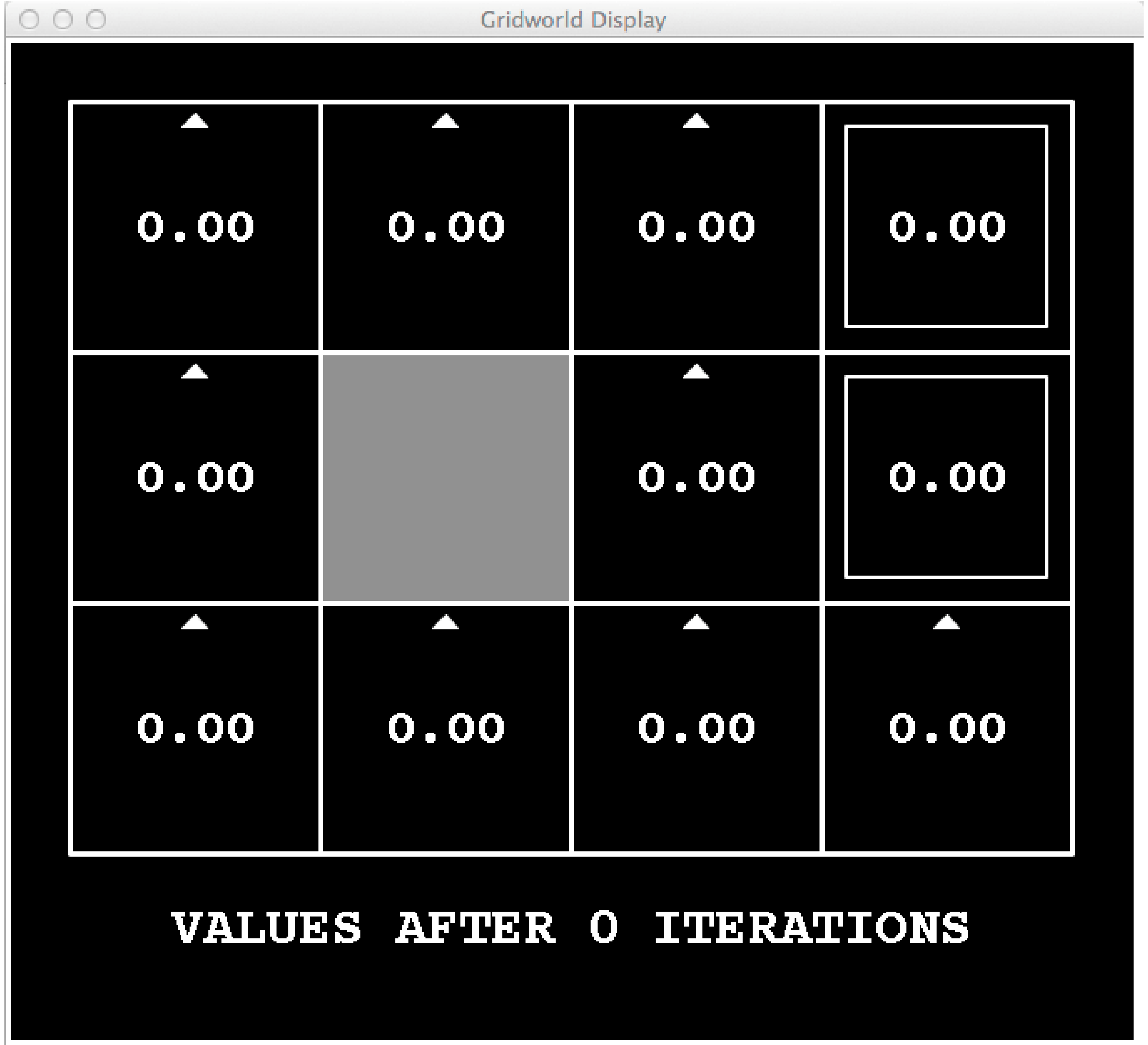
- Itération de la valeur repètent les mises à jour de Bellman:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s') \right]$$



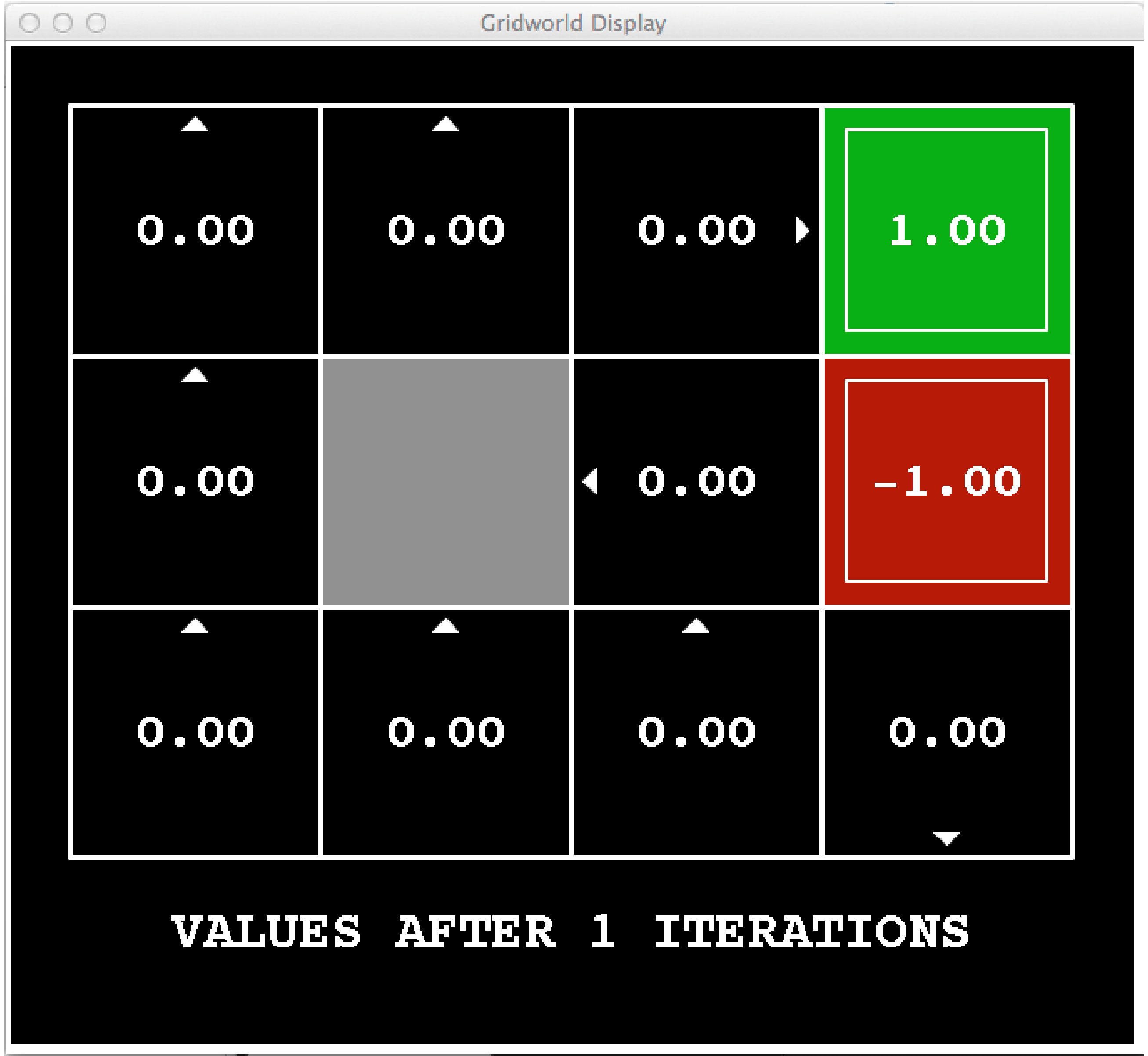
- Problème 1: très lent – $O(S^2A)$ par iteration.
- Problème 2: Le "max" à chaque iteration change rarement.
- Problème 3: La stratégie converge souvent avant les valeurs.

k=0



Bruit = 0.2
Remise = 0.9
Récompense vie = 0

k=1



Bruit = 0.2
Remise = 0.9
Récompense vie = 0

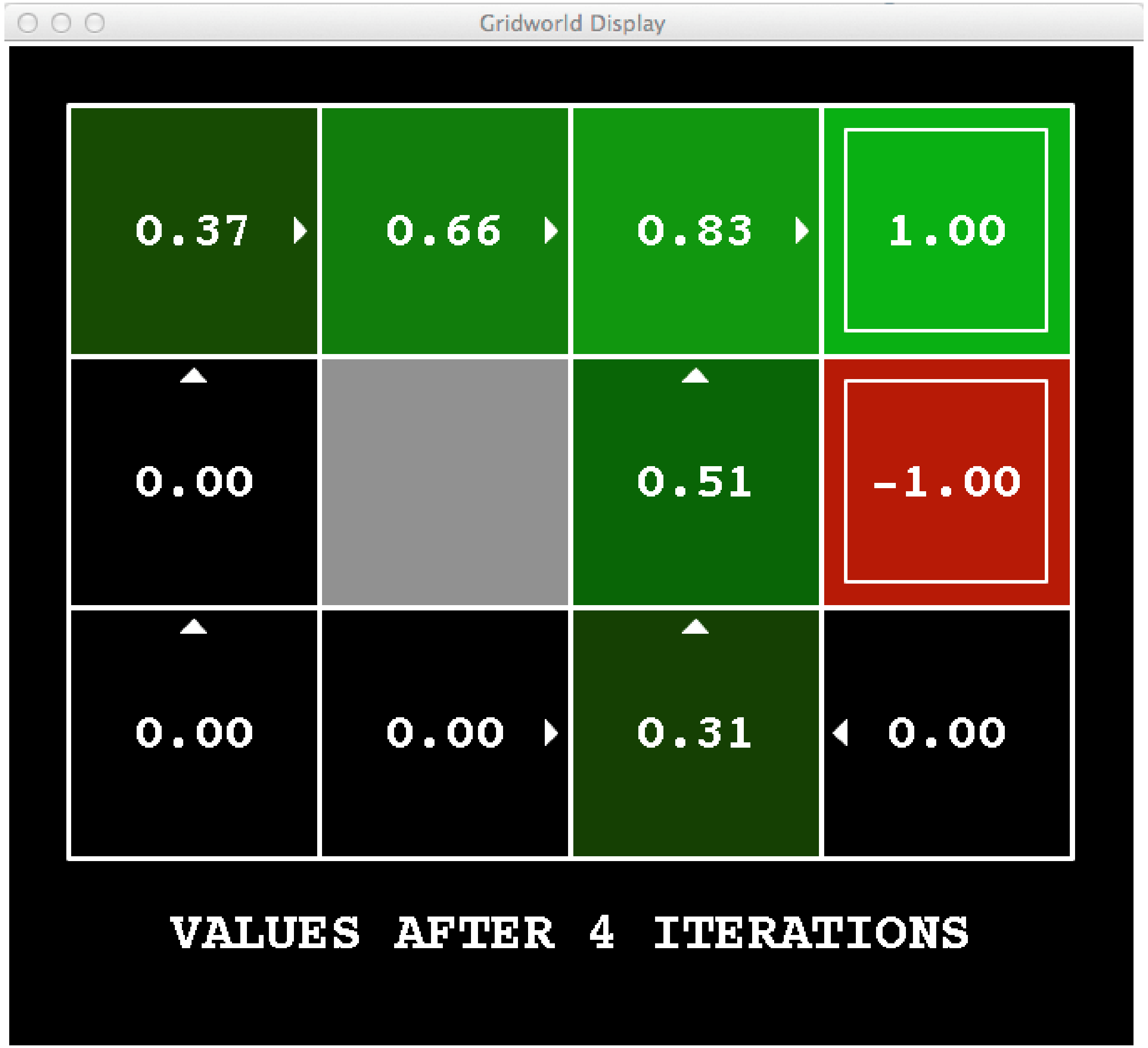
k=2

Noise = 0.2
Discount = 0.9
Living reward = 0

k=3

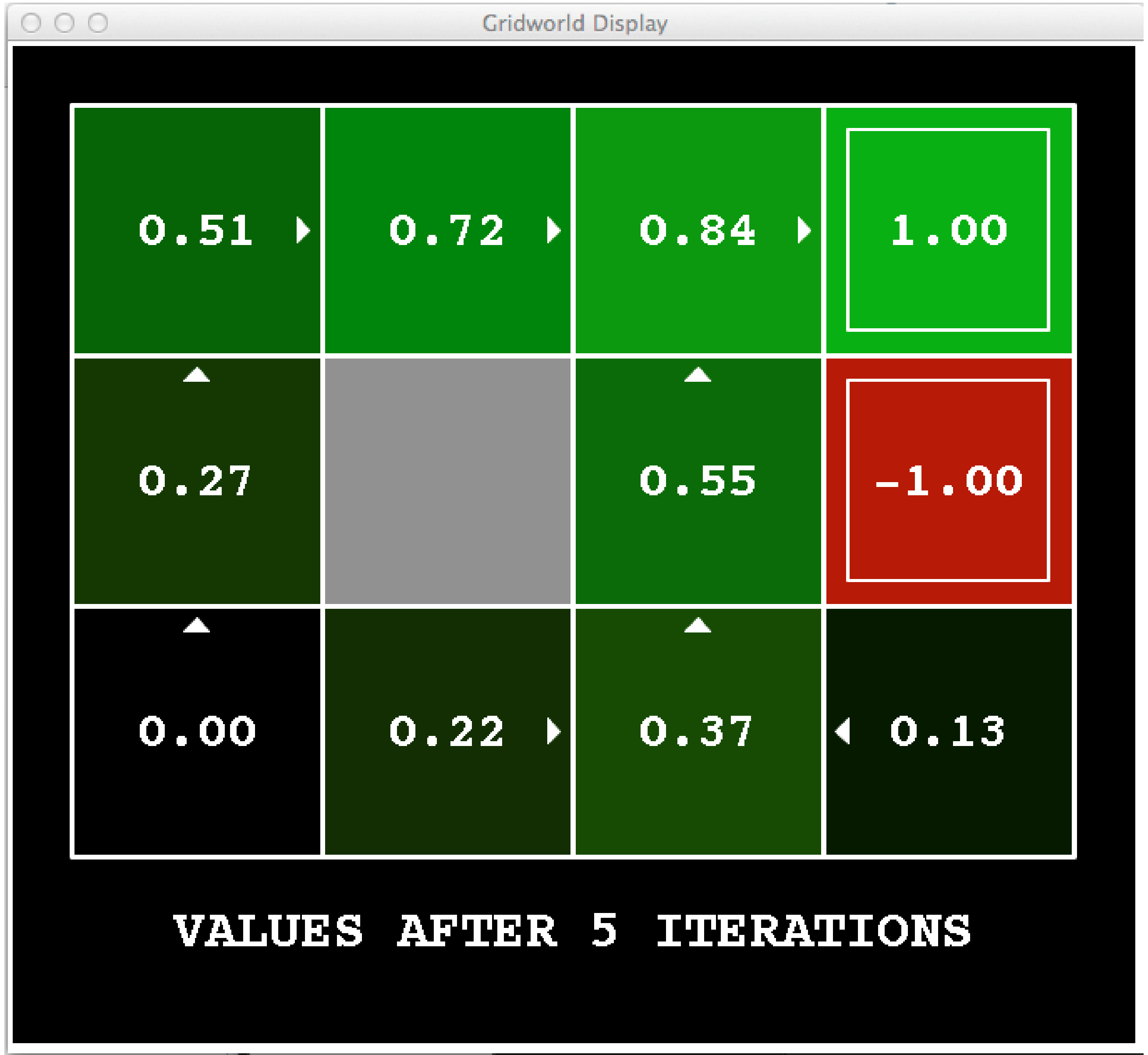
Bruit= 0.2
Remise = 0.9
Récompense vie = 0

k=4



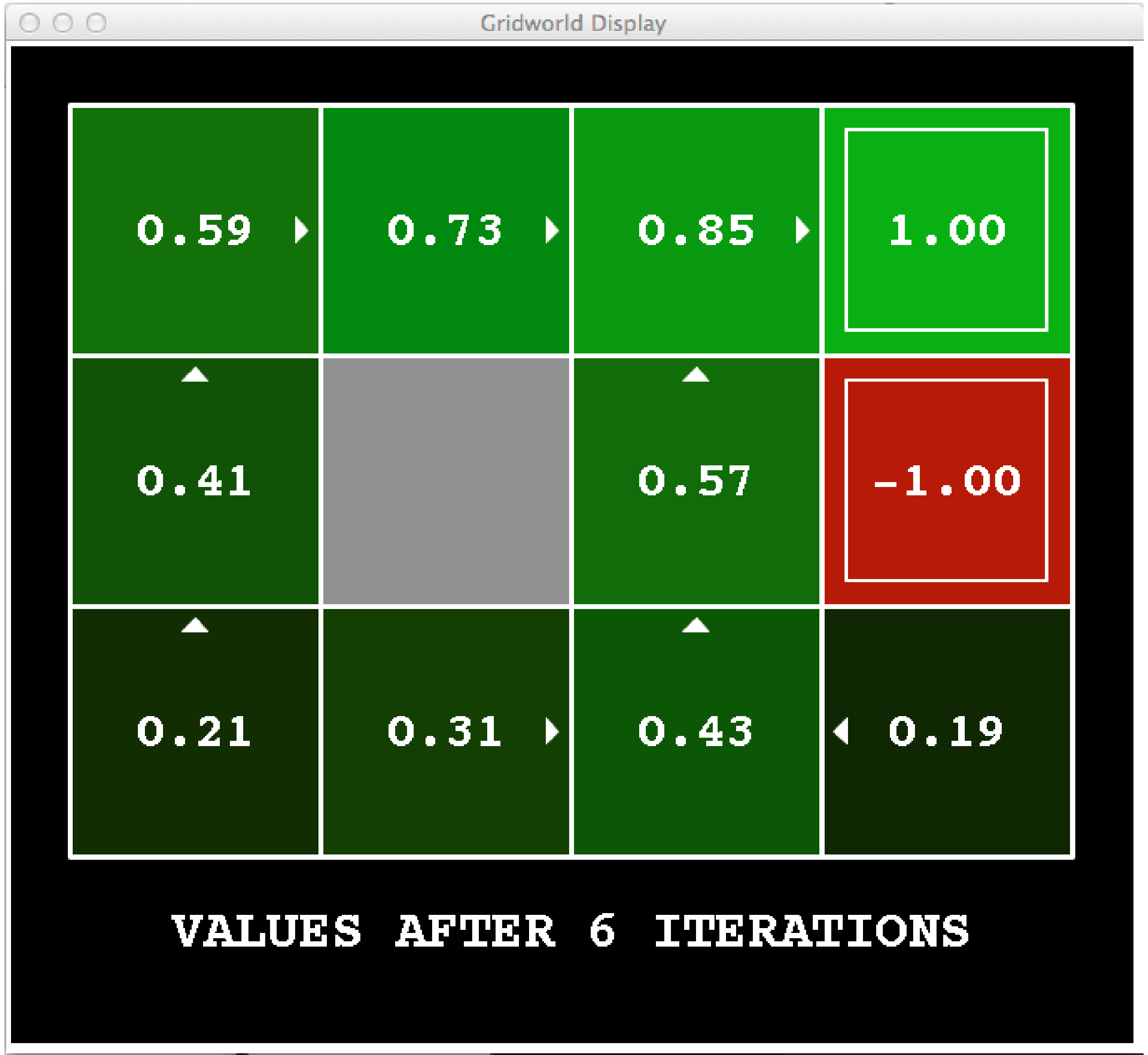
Bruit = 0.2
Remise = 0.9
Récompense vie = 0

k=5



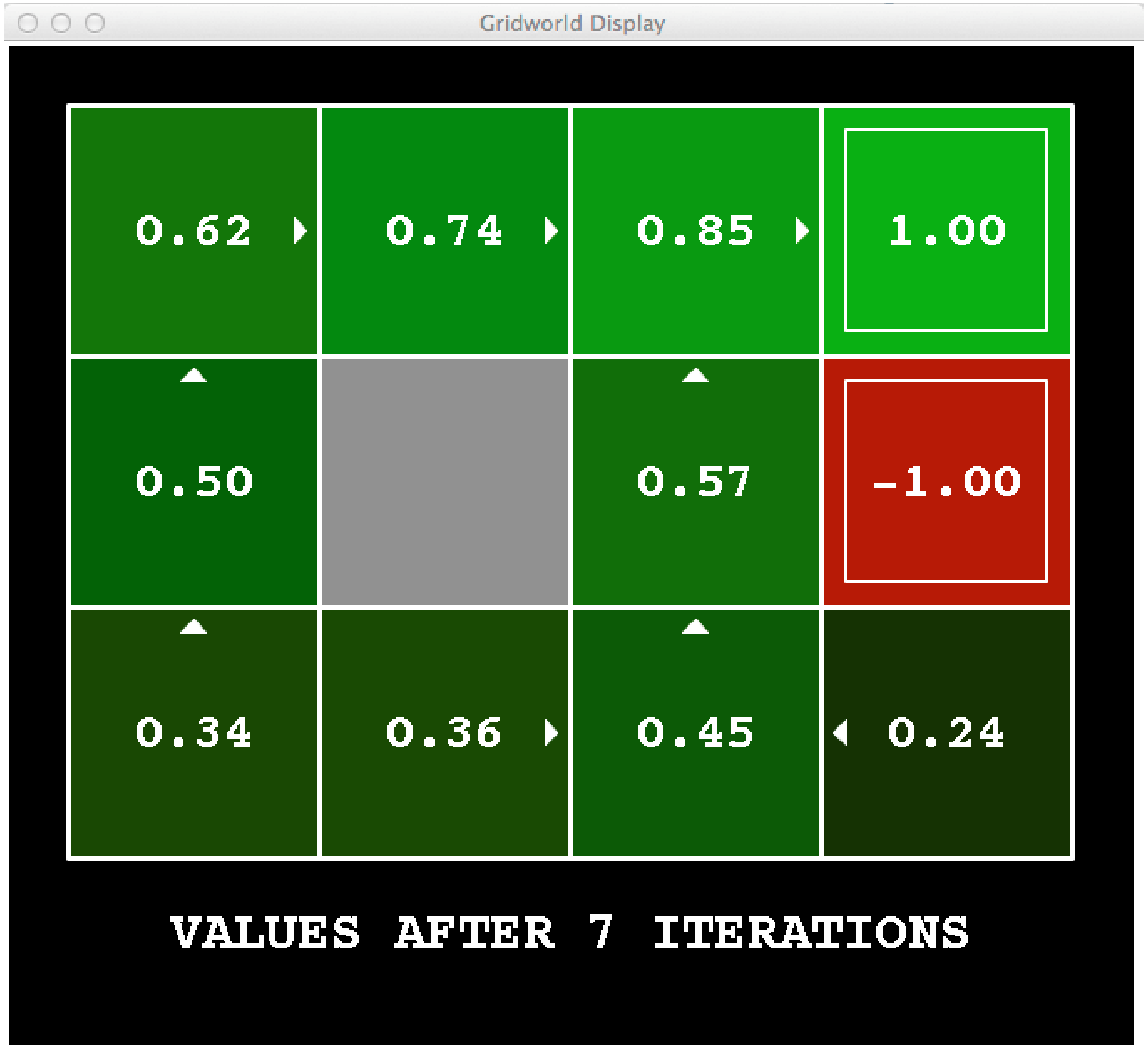
Bruit = 0.2
Remise = 0.9
Récompense vie = 0

k=6



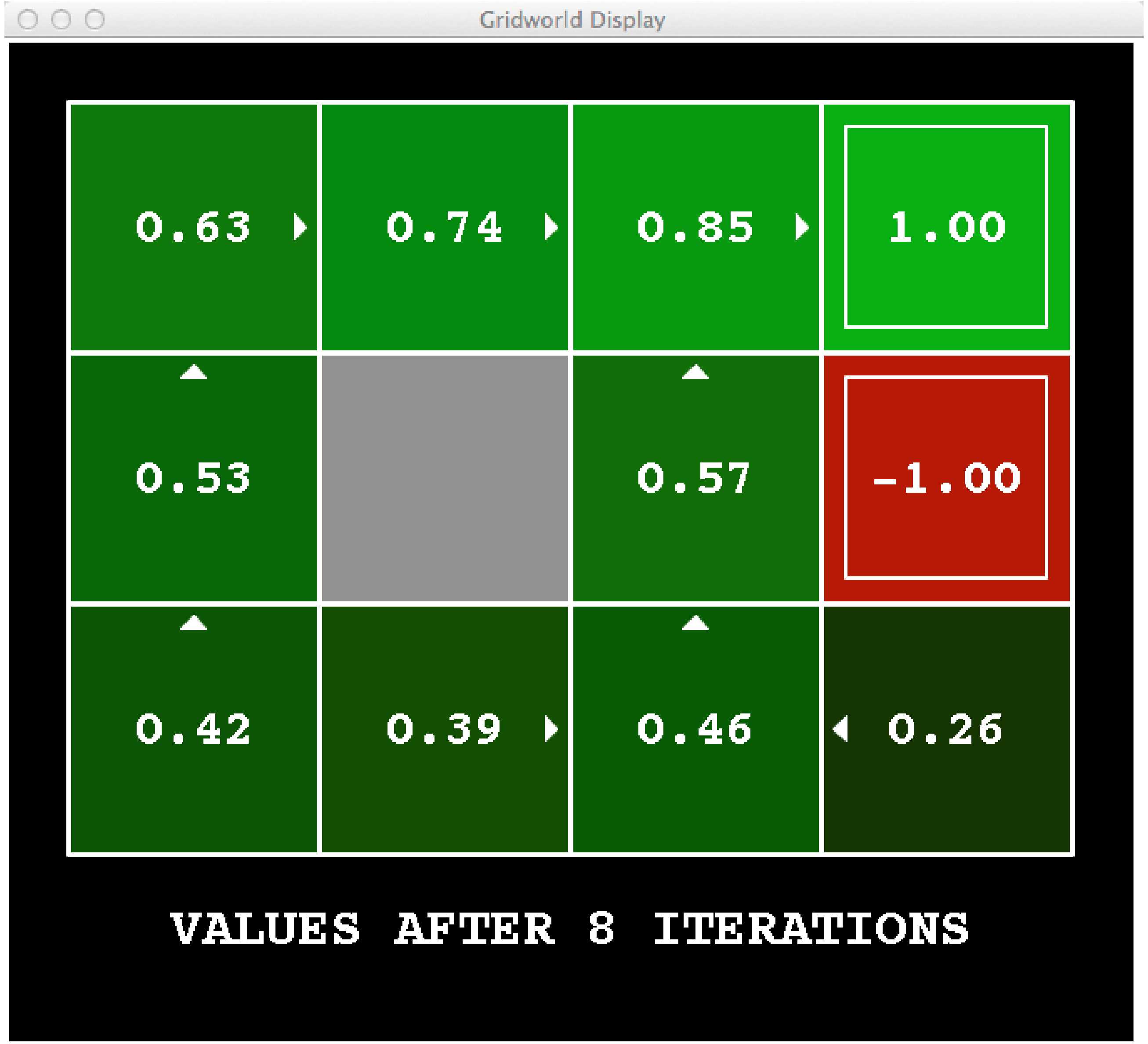
Bruit = 0.2
Remise = 0.9
Récompense vie = 0

k=7



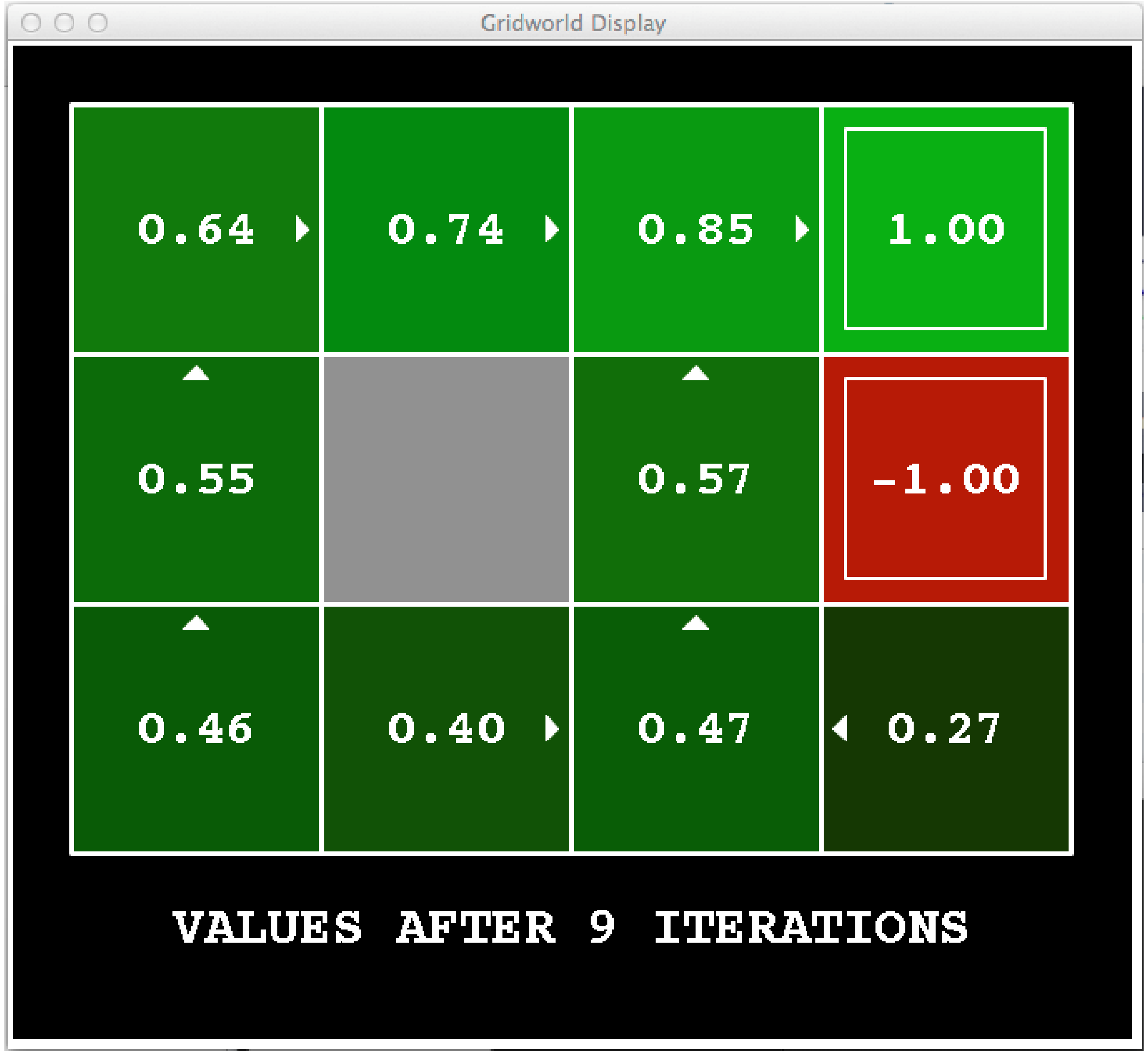
Bruit = 0.2
Remise = 0.9
Récompense vie = 0

k=8



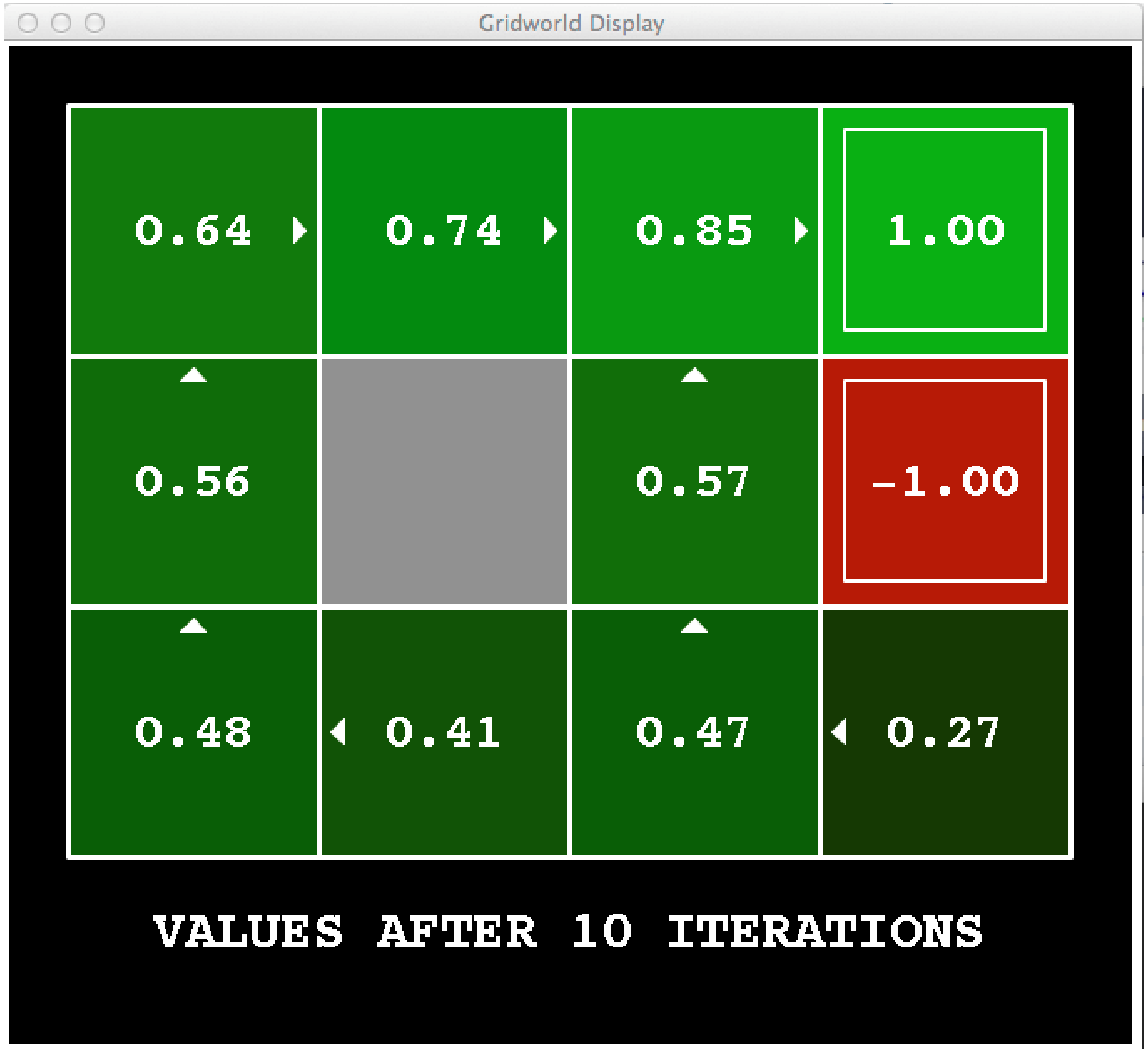
Noise = 0.2
Discount = 0.9
Living reward = 0

k=9



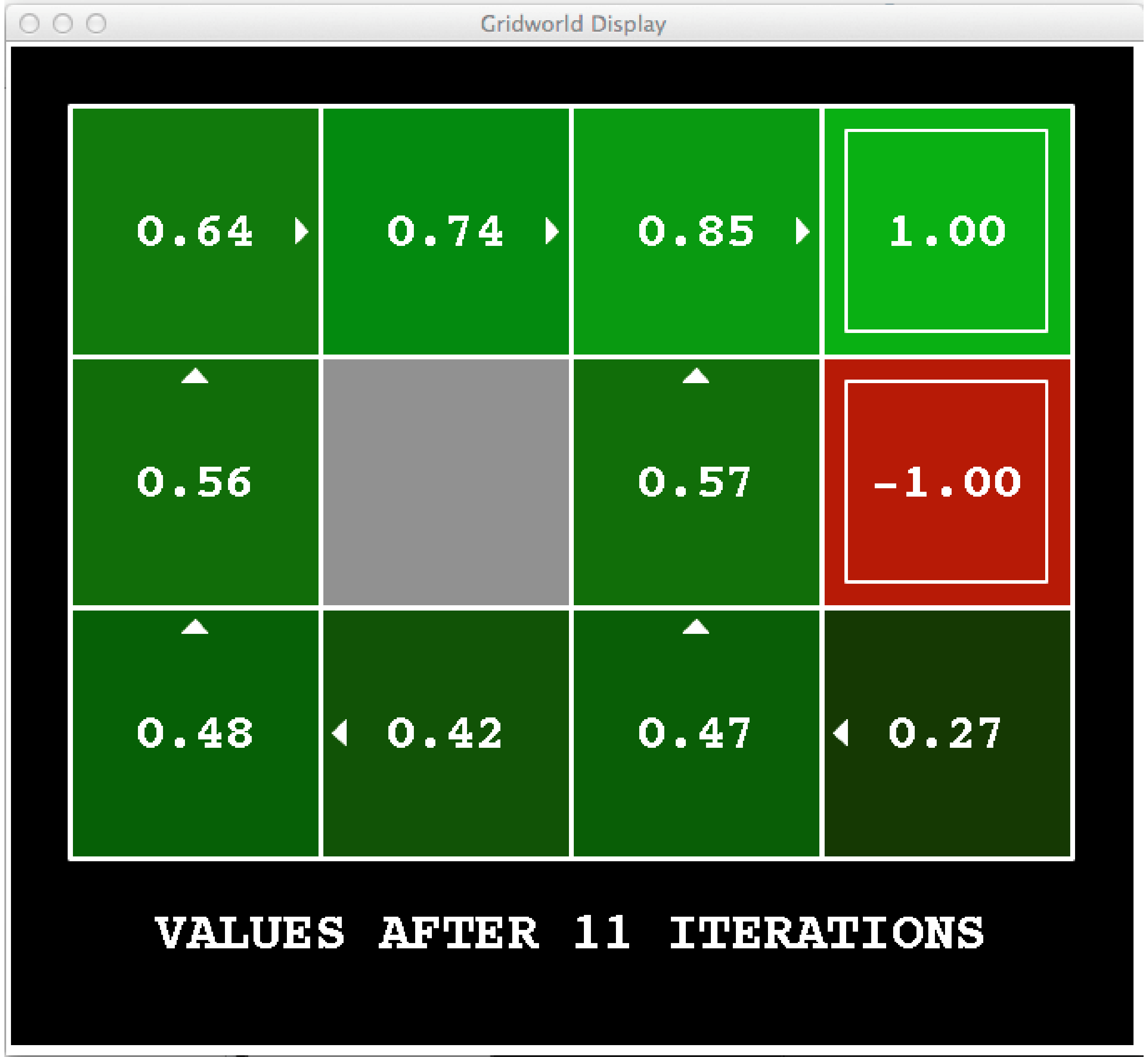
Bruit = 0.2
Remise = 0.9
Récompense vie = 0

k=10



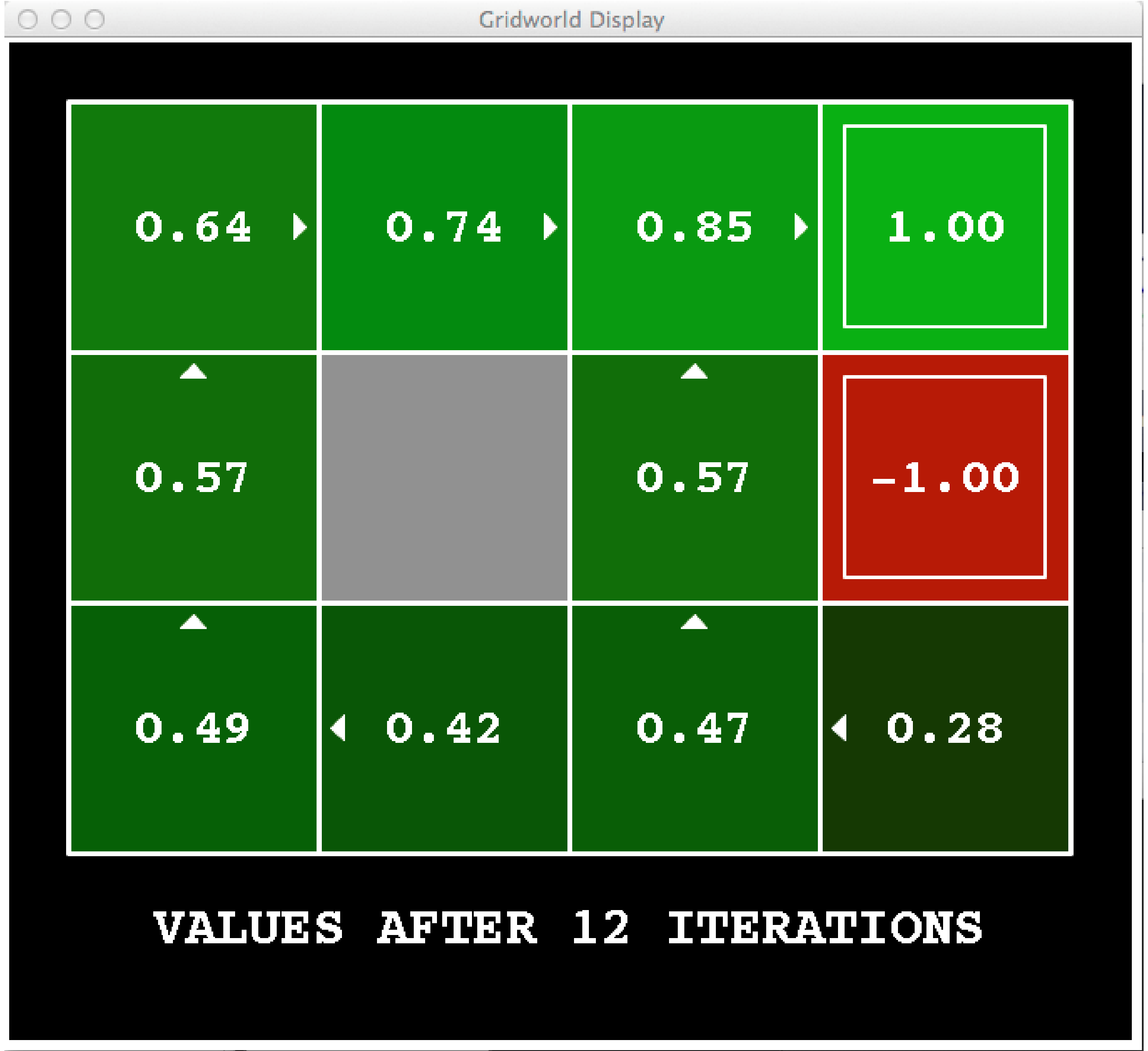
Bruit = 0.2
Remise = 0.9
Récompense vie = 0

k=11



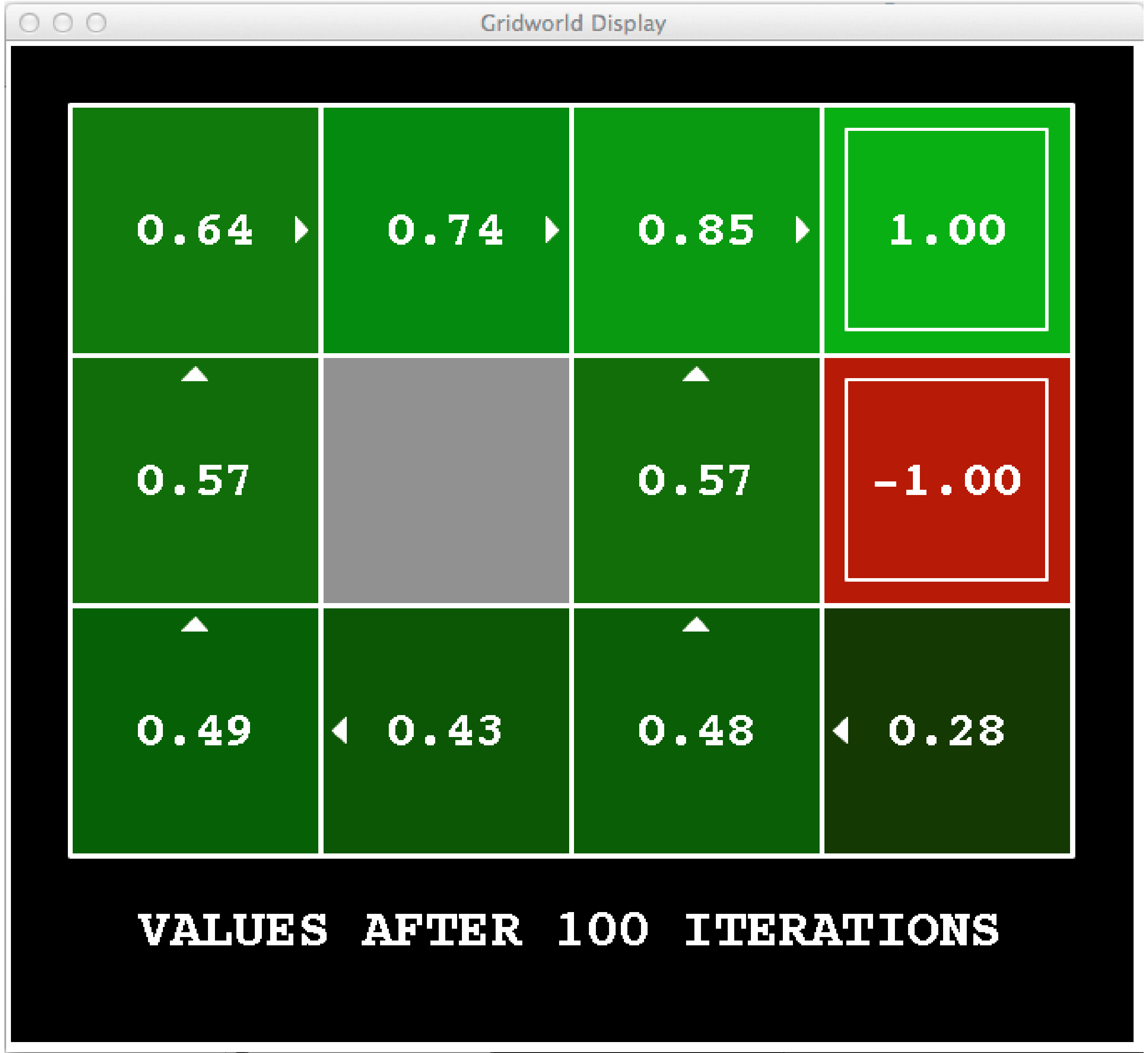
Bruit = 0.2
Remise = 0.9
Récompense vie = 0

k=12



Bruit = 0.2
Remise = 0.9
Récompense vie = 0

k=100



Bruit = 0.2
Remise = 0.9
Récompense vie = 0

Itération de la stratégie

- Approche alternative à l'iteration de la valeur:
  - Step 1: évaluation de la stratégie: calcule les utilités pour une stratégie fixée.
  - Step 2: Amélioration de la stratégie: Mise à jour en utilisant l'exratction de la stratégie
  - Repéter jusqu'à convergence.

- Itération de la stratégie
  - Toujours optimal!
  - Converge plus rapidement sous certains conditions.

Itération de la stratégie

- Evaluation: Pour une **stratégie fixée** π, Calculer les valeurs:
  - Itérer jusqu'à convergence:

$$V_{k+1}^{\pi_i}(s) \leftarrow \sum_{s'} T(s, \pi_i(s), s') \left[R(s, \pi_i(s), s') + \gamma V_k^{\pi_i}(s')\right]$$

- Amélioration: Pour des **valeurs fixées**, Extraire une meilleure stratégie
  - Extraction de la politique:

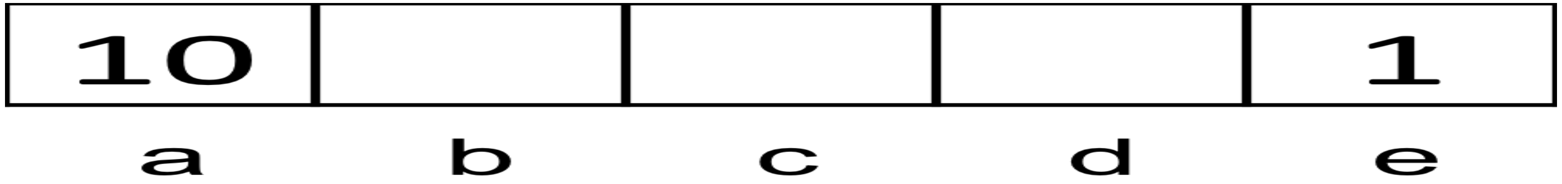
$$\pi_{i+1}(s) = \arg\max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^{\pi_i}(s')\right]$$

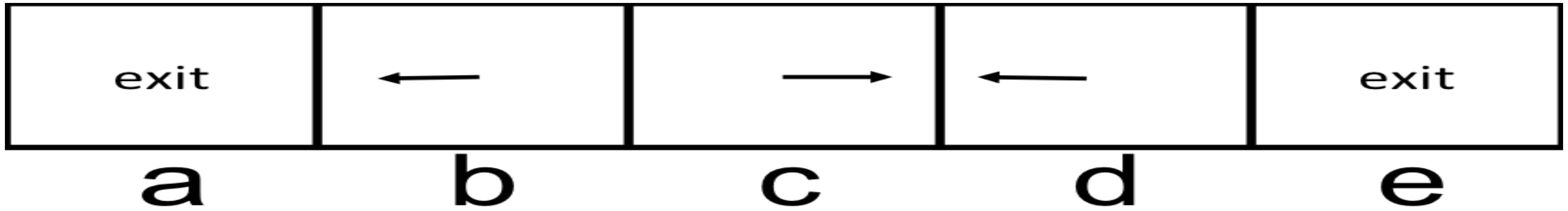
Comparaison

- Les deux algorithmes itértion de la (valeur/ stratégie) calculent la même stratégie.
- Dans l'itération de la valeur:
  - Chaque itération améliore les deux entitées (valeurs + stratégie)
  - Mais on suit pas explicitement la stratégie.
- Pour l'iteration de la stratégie:
  - On réalise plusieurs ameliorations avec une stratégie fixée. Chaque iteration est rapide (résolution d'un système linéaire).
  - Après l'évalution de la stratégie, On choisit une nouvelle stratégie
- Deux programmes dynamiques pour la resolution d'un MDP.

Quiz

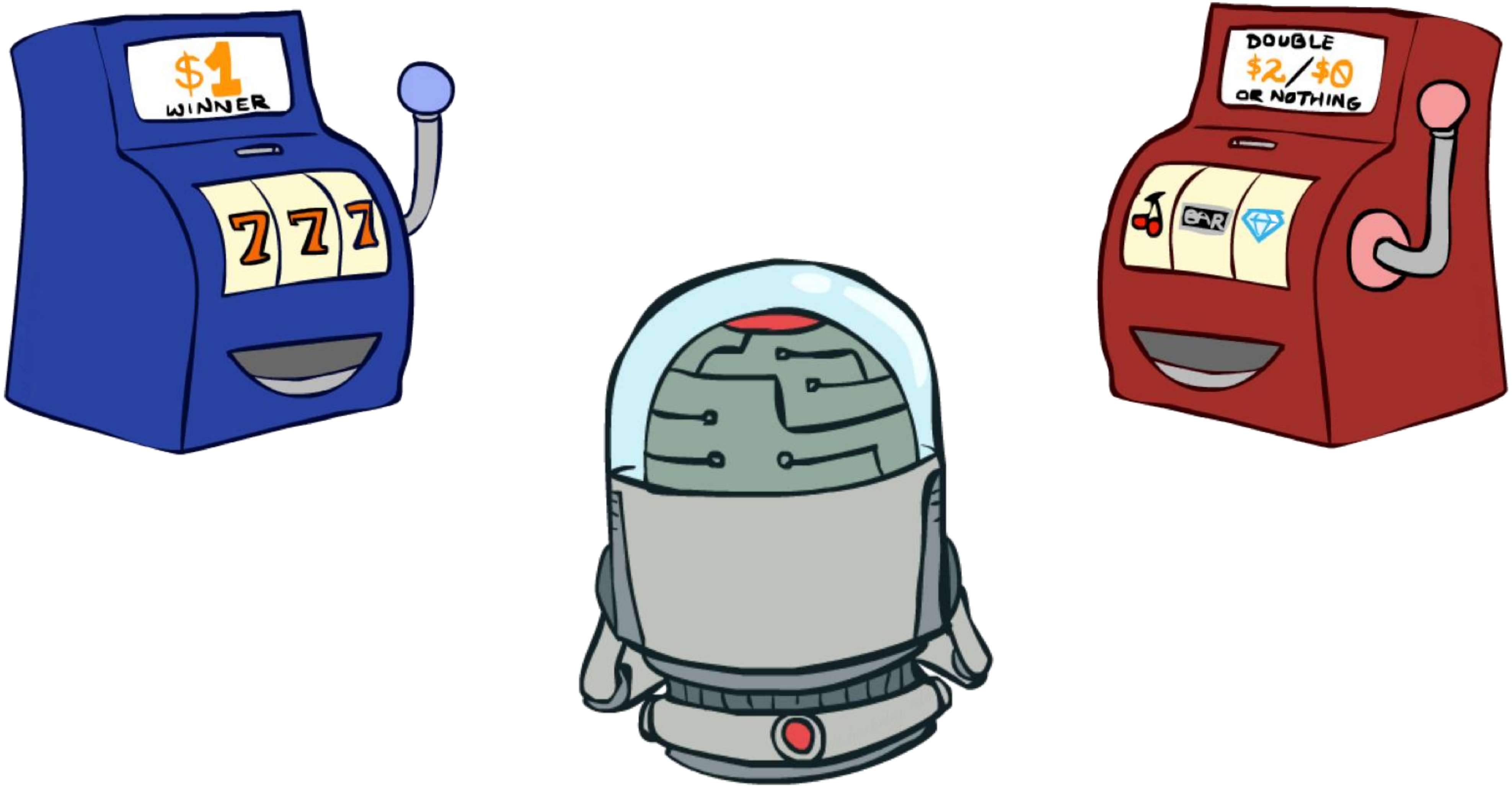
On considère le monde grille, où on peut se déplacer vers les deux nœuds voisins.
Toutes les actions sont réussies et on considère une remise $\gamma = 0.9$





1. Calculer les valeurs de chaque nœud, selon la stratégie présentée.
2. Améliorer la stratégie selon les valeurs calculées.

Bandits

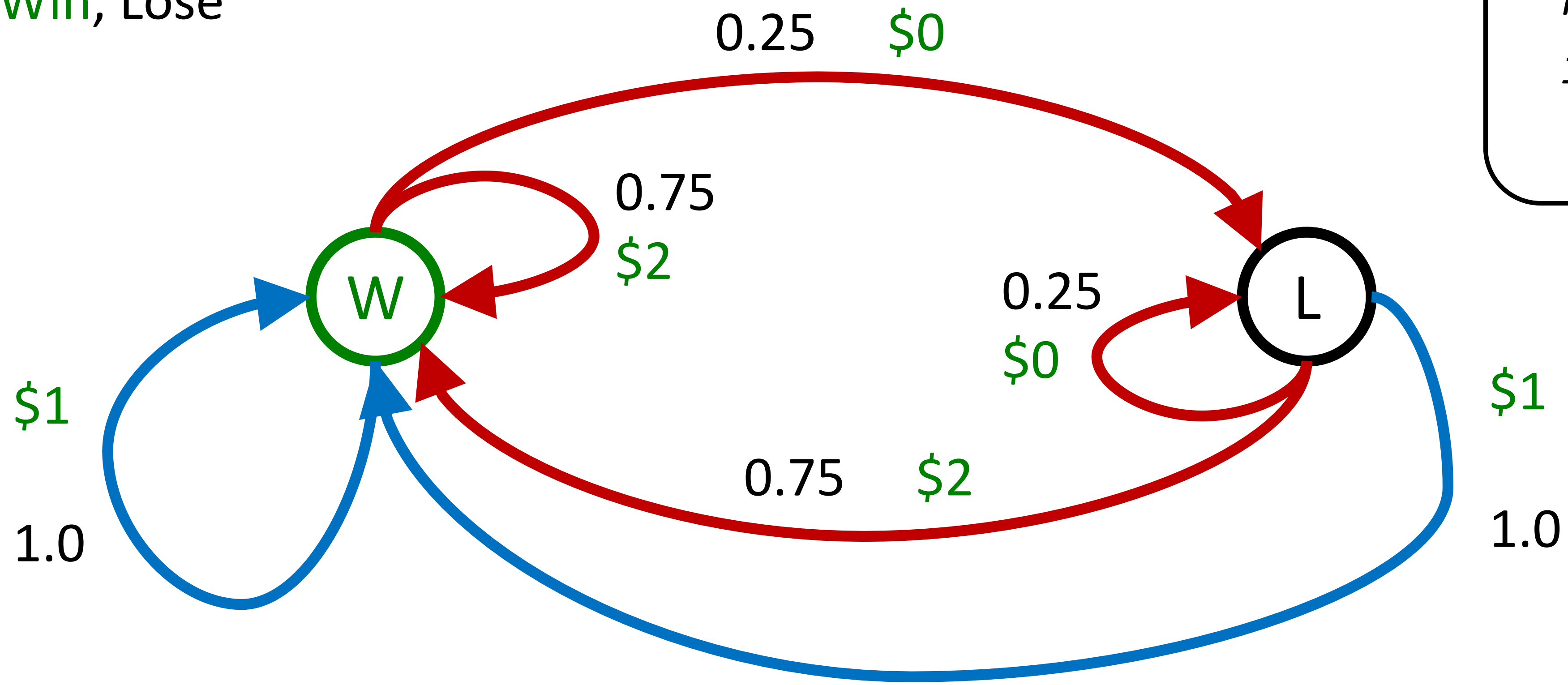


Bandits MDP

- Actions: *Blue*, *Red*
- Etats: Win, Lose

Pas de remise

100 itérations



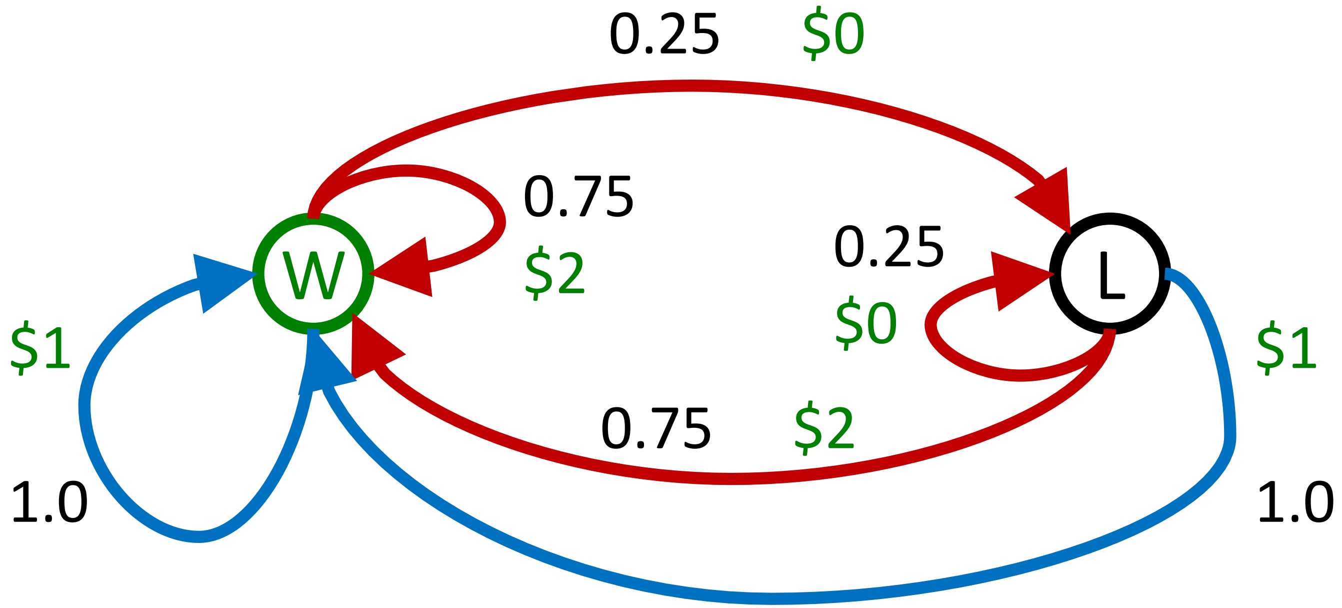
Planification Offline

- Résoudre un MDPs est une planification offline
  - You determine all quantities through computation
  - You need to know the details of the MDP
  - You do not actually play the game!

Pas de remise

100 time steps

	Valeur
Jouer Red	150
Jouer Blue	100



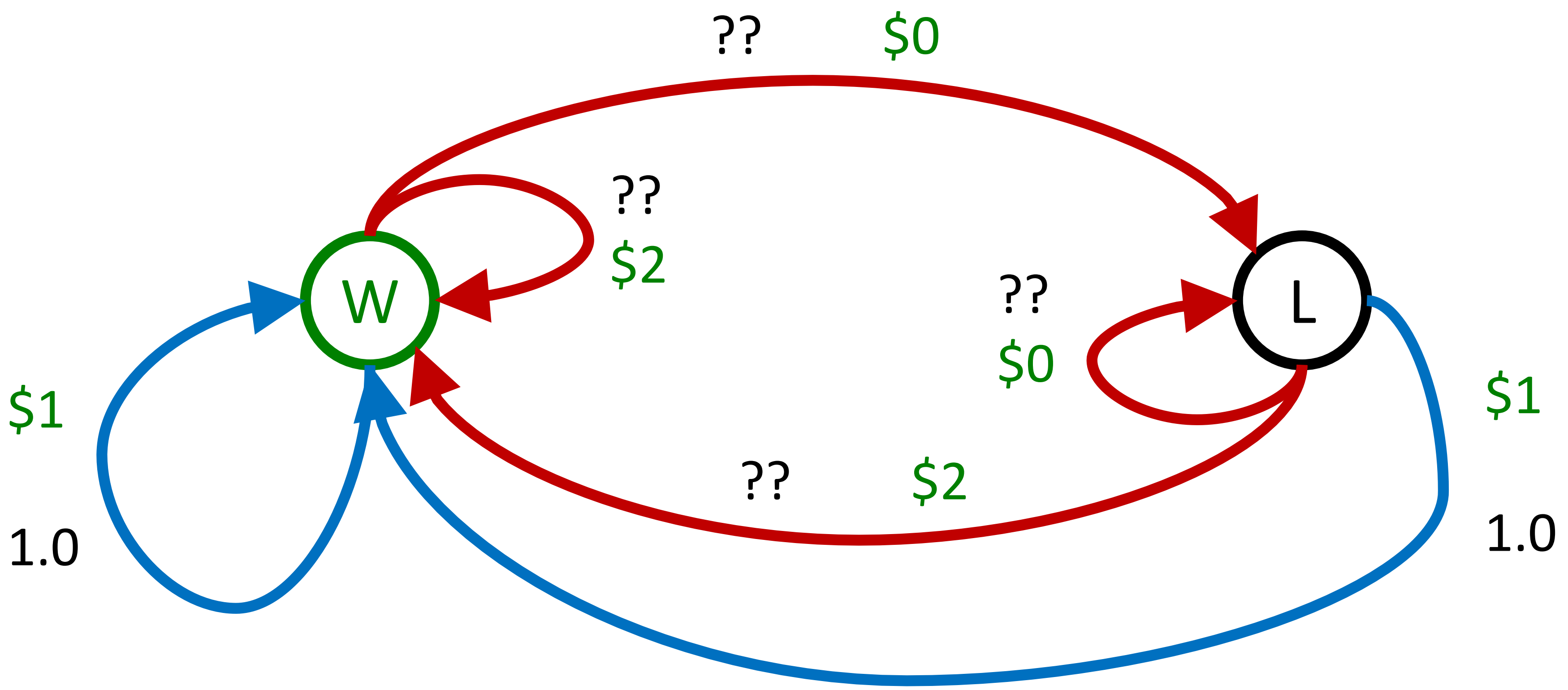
Jouons!

\$2 \$2 \$0 \$2 \$2

\$2 \$2 \$0 \$0 \$0

Planification Online

- Les règles ont changé! Change de gagner pour Red est différente.



Jouons!

$0 $0 $0 $2 $0

$2 $0 $0 $0 $0

Que s’est il passé?

- C’etait pas de la planification, mais un **apprentissage**!
  - Plus précisement, un **apprentissage par reinforcement**
  - On possédait une MDP, mais impossible de résoudre par calcul.
  - On devait prendre des actions pour pouvoir le résoudre.

- Idées Importantes de l’apprentissage par reinforcement.
  - Exploration: Essayer de nouvelles actions pour obtenir plus d’informations.
  - Exploitation: A un moment, on utilize les résultats calculés.
  - Regret: Même avec un apprentissage intelligent on commet des erreurs.