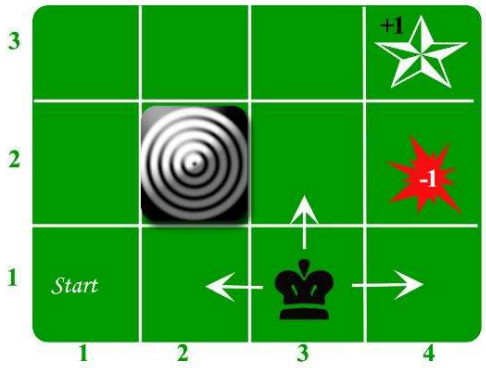
Example 4 The Grid world (迷宮問題, matlab RL toolbax有這個例題)

- An agent lives in the grid. The above example is a 3*4 grid. The grid has a START state(grid no 1,1). The purpose of the agent is to wander around the grid to finally reach the Blue Diamond (grid no 4,3). Under all circumstances, the agent should avoid the Fire grid (orange color, grid no 4,2). Also the grid no 2,2 is a blocked grid, it acts as a wall hence the agent cannot enter it.
- Action：The agent can take any one of these actions: UP, DOWN, LEFT, RIGHT. Walls block the agent path, i.e., if there is a wall in the direction the agent would have taken, the agent stays in the same place. So for example, if the agent says LEFT in the START grid he would stay put in the START grid.
- The move is noisy with 80% of the probability the intended action works correctly, and 20% of the probability the action agent takes causes it to move at right angles. For example, if the agent says UP the probability of going UP is 0.8 whereas the probability of going LEFT is 0.1, and the probability of going RIGHT is 0.1 (since LEFT and RIGHT are right angles to UP).
- **The goal:** to find the shortest sequence getting from START to the Diamond.



- **The agent receives rewards each time step:-**
  - Small reward each step (can be negative when can also be term as punishment, in the above example entering the Fire can have a reward of -1).
  - Big rewards come at the end.
  - The goal is to Maximize the sum of rewards.

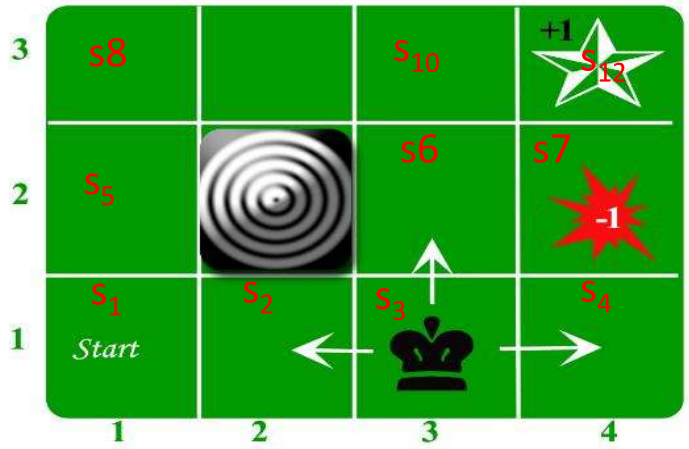
- MDP settings:
  - Actions: "up," "down," "left," "right"
  - Noise of transition probability $P(s'|s;a)$: 0.2,

e.g., up: .8 / .1 . .1 / .0 , right: .1 / 0 . .8 / .1 , right: .1 / 0 .1 .8 ■ ,

etc.

  - γ: 0.9

被擋住



■ 以Up為例，當命令機器人往上走，他只有 80% 的機率會真的走到上面那一格，另外有 10% 的機率會走到右邊那格、10% 的機率會走到左邊那格。

■ 經過 100 次 iteration，會得到在每個 state，往哪邊走會得到最大的 value，如右圖所示之 value 跟 action 方向。此為讓機器人知道他在不同 state 該怎麼選擇 action 的 policy。

計算例:

- Reward for terminal states: R=+1 for s11 and R=-1 for s7
- Reward for the other states: R=0

K=2

K=3

K=3, At s, for action "right"

$$v_3^*(s10) = 0 + 0.9 \sum_{s'} p(s10, right, s') \times v_2^*(s')$$

v_3^*(s10)=0+0.9*(1.0*0.8 + 0.0*0.1 + 0.72*0.1)=0.784

right down Up (碰牆壁留原地)

K=4, At s10, for action "right"

$$V_4^*(s10) = 0 + 0.9 \sum_{s'} p(s10, right, s') \times v_3^*(s')$$

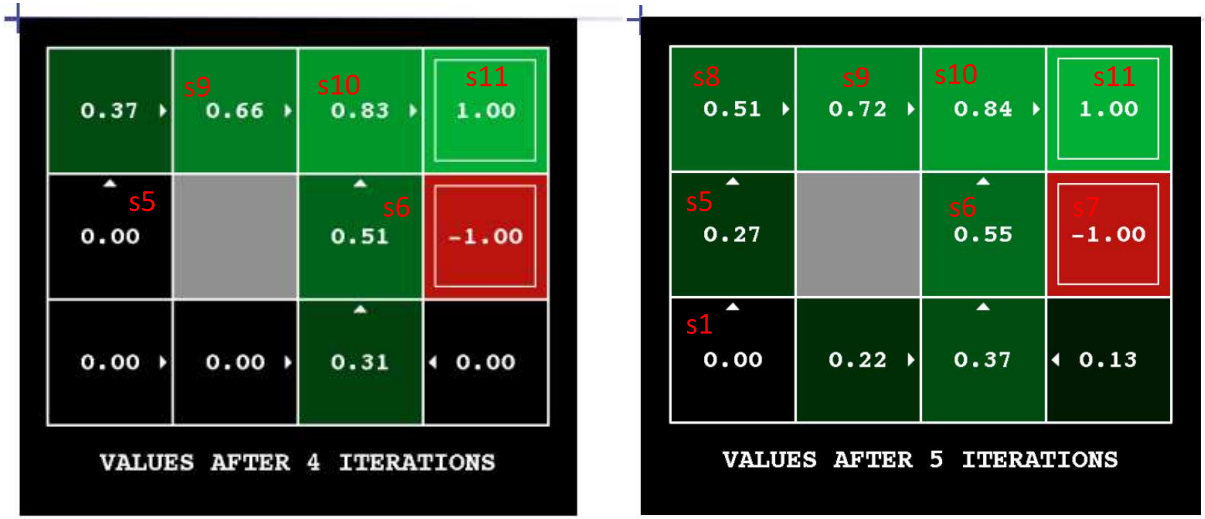
v_4^*(s10)=0+0.9*(1.0*0.8+0.78*0.1+0.43*0.1)=0.828

K=4

Note:以上為針對最佳方向的計算值，正常情況下必須4個方向都要計算，然後取最大的$v_4^*(s10)$，此例題主要為理解算法，了解k=4與k=3的關係

計算例:

- Reward for terminal states: R=+1 for s11, R=-1 for s_7
- Reward for the other states: R=0



$v_4^*(s)$ ➡ $v_5^*(s)$

- 注意terminal states之計算

$V_5^*(s11) = r(s_{11}, a) + 0.9 \sum_{s'} p(s_{11}, a, s') \times v_4^*(s') = 1 + 0 = 1$ (0)

$v_5^*(s7) = r(s7, a) + 0.9 \sum_{s'} p(s_7, a, s') \times v_4^*(s') = -1 + 0 = -1$ (0)

- At s10, the optimal *a* is action "right"

$v_5^*(s10) = 0 + 0.9 \sum_{s'} P(s10, right, s') v_4^*(s')$

$=0.9 \times (1.0*0.8+0.51*0.1+0.83*0.1)=0.84$

- At s6, the optimal *a* is action "up"

$v_5^*(s6) =$

$0 + 0.9 \times [0.83*0.8+(-1.0)*0.1+0.51*0.1]=0.55$

- At s9, the optimal *a* is action "right"

$v_5^*(s9) =$

$0 + 0.9*[0.83*0.8+0.66*0.1+0.66*0.1]=0.716$

- At s5, the optimal *a* is action "up"

$v_5^*(s5) =$

$0+0.9 \times [0.37*0.8+0.0*0.1+0.0*0.1]=0.266$