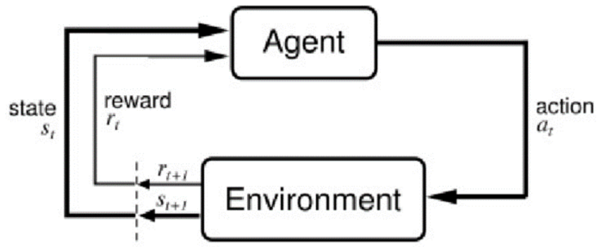
Lab 8: Temporal Difference Learning

Lab Objective:

In this project, you are going to build a AI to play 2048 through reinforcement learning, TD(0). This AI should be able to improve its performance on 2048 through played game sequences/experiences. The output of the AI would be the $Q(s,a)$ of the input state s.

Lab Description:

- Reinforcement Learning is a computational approach to learning from interaction and is focus on goal-directed learning.
- Agent-Environment Interaction Framework
  - Agent: The learner and decision-maker.
  - Environment: The thing it interacts with, comprising everything outside the agent.
  - State: whatever information is available to the agent.
  - Reward: single numbers.



- Temporal Difference Learning(TD-Learning) is a kind of reinforcement learning and is able to adjust weights automatically.
  - The goal of TD-Learning is to predict the actual return R_t
  - The way is adjust the weights is through: $V(s_t) = V(s_t) + \alpha\big(R_t - V(s_t)\big)$
  - **TD(0):** $\boldsymbol{V(s_t) = V(s_t) + \alpha(r_{t+1} + V(s_{t+1}) - V(s_t))}$
- According to [1], because 2048 is a stochastic game (random tile generation after player's move). Training by afterstate is a better choice. Please Check *Methodology* section for details.
- <u>Please hand in your source code and report, and demo to TAs.</u>

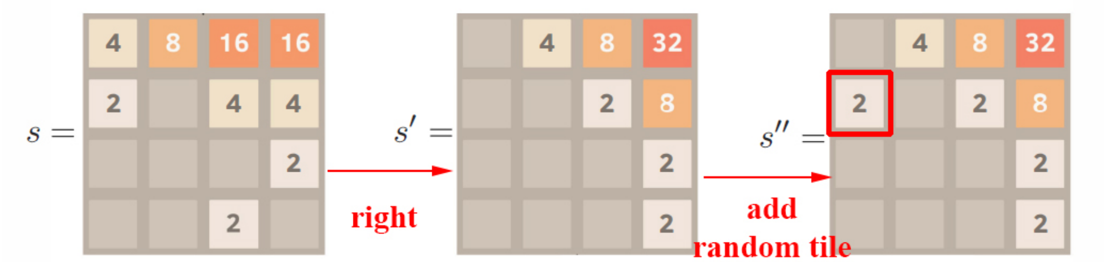
Game Environment – 2048:

- Introduction: 2048 is a single-player sliding block puzzle game. The game's objective is to slide numbered tiles on a grid to combine them to create a tile with the number 2048.
- Actions: **Up**, **Down**, **Left**, **Right**

- Reward: The score is the value of new tile when two tiles are combined.

2048

- A sample of two-step state transition



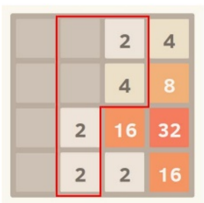
Implementation Details:

- Learning Rate:
  - Start with: 0.0025
  - Train 400k~500k per tuple configuration
- Tuples: 4 tuples
  - 4 (1)

  - 4 (2)

  - 6 (1)

  - 6 (2)



Requirements:

1. Set tuples
2. Finish AI::get_best_move().
3. Finish AI::update_tuple_values().
   - Note: Pay attention to terminal states.
4. Try different tuple settings.

Functions:

- state::move(int *dir*) – move the game board according to *dir* and save the reward in private member variable *reward*.
- state::get_reward() – return the reward when the action applied.
- state::evaluate_score() – evaluate the board score by the value of tuples.
- feature::list().push_back(new pattern<N>(TILE INDEXES)) – set the size and the shape of a tuple
  - Tile index:

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

- Save and Load tuple weights. (Please check code for detail)

Methodology:

A pseudocode of a game engine

function PLAY GAME

$score \leftarrow 0$

$s \leftarrow$ INITIALIZE GAME STATE

while IS NOT TERMINAL STATE(s) **do**

$a \leftarrow \underset{a' \in A(s)}{\operatorname{argmax}}$ EVALUATE(s, a')

$r, s', s'' \leftarrow$ MAKE MOVE(s, a)

$score \leftarrow score + r$

$s \leftarrow s''$

if LEARNING ENABLED **then**

LEARN EVALUATION(s, a, r, s', s'')

return $score$

TD(0)-afterstate

function EVALUATE(s, a)
  $s', r \leftarrow$ COMPUTE_AFTERSTATE(s, a)
  return $r + V(s')$

function LEARN EVALUATION(s, a, r, s', s'')
  $a_{next} \leftarrow \underset{a' \in A(s'')}{\operatorname{argmax}} EVALUATE(s'', a')$
  $s'_{next}, r_{next} \leftarrow COMPUTE\ \ AFTERSTATE(s'', a_{next})$
  $V(s') \leftarrow V(s') + \alpha(r_{next} + V(s'_{next}) - V(s'))$

Scoring Criteria:

- Winning (reach 2048) rate in 1000 games. (50%)
- Report
  - Describe how you implement AI::get_best_move() (15%)
  - Describe how you implement AI::update_tuple_values() (15%)
  - Statistic charts include following data
    - Winning rate and average score of standard tuple setting with 0.0025 learning rate (10%)
    - Winning rate and average score of your tuple setting with learning rate 0.0025 (10%)
  - Discussion (Reinforcement Learning, TD) (5%)

References:

[1] Szubert, Marcin, and Wojciech Jaśkowski. "Temporal difference learning of N-tuple networks for the game 2048." 2014 IEEE Conference on Computational Intelligence and Games. IEEE, 2014.

[2] Kun-Hao Yeh, I-Chen Wu, Chu-Hsuan Hsueh, Chia-Chuan Chang, Chao-Chin Liang, and Han Chiang, Multi-Stage Temporal Difference Learning for 2048-like Games, accepted by IEEE Transactions on Computational Intelligence and AI in Games (SCI), doi: 10.1109/TCIAIG.2016.2593710, 2016.