ECE 277, FALL 2020
GPU Programming

Department of Electrical and Computer Engineering
University of California, San Diego

LAB 2: Reinforcement learning: Q-learning (Single Agent)

This lab requires to design an agent to interact with the environment using the reinforcement learning algorithm in Figure 1. Specifically, you need to design a single thread agent to maximize rewards from the mine game environment using Q-learning. The agent should interact with the given mine game environment in Figure 2.

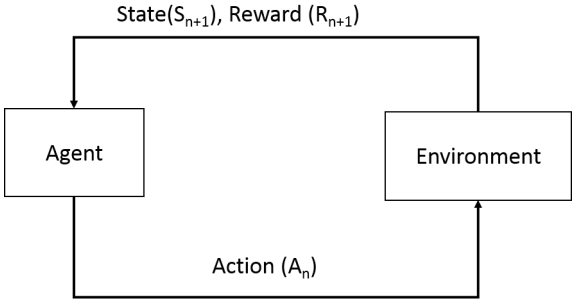

Figure 1: Reinforcement learning.

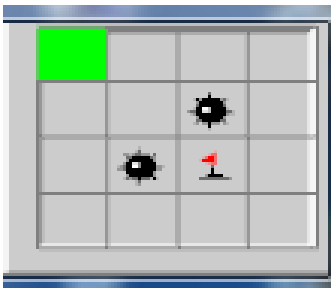

Figure 2: 4x4 mine game environment.

- Action: (right:0, down:1, left:2, up:3)
- State: (x, y) current position of an agent in the coordinator of (0,0) at the top-left corner
- Reward: flag: +1, mine: -1, otherwise: 0
- Every episode restarts from (0,0) after the agent reaches one of mines or a flag.

You should not modify any given codes except CMakeLists to add your codes. You only need to add your agent code to the lab project.

You have to use CUDA to program a single agent with

```
Agent_update,<<<1, 1>>>, Agent_action<<<1, 1>>>,
 Agent_init<<<1, 1>>>, etc.
```

Q table should be initialized using a CUDA kernel (not cudaMemSet, which is very slow.)

Interface pointers of all the extern functions are allocated to the Device (GPU) memory (not CPU memory). The below function is an informative RL environment routine to show when and how agent functions are called

```
extern void agent_init();
extern float agent_adjustepsilon();
extern short* agent_action(int2* cstate);
extern void agent_update(int2* cstate, int2* nstate, float* rewards);

int qlearningCls::learning(int *board, unsigned int &episode, unsigned int &steps)
    if (m_episode == 0 && m_steps==0) {// only for first episode
        env.reset(m_sid);
        agent_init(); // crate and initialize Q table + self initialization
    }else {
        active_agent = checkstatus(board, env.m_state, flag_agent);

        if (m_newepisode) {
            env.reset(m_sid);
            float epsilon = agent_adjustepsilon(); // adjust epsilon
            m_steps = 0;
            printf("EP=%4d, eps=%4.3f\n", m_episode, epsilon);
            m_episode++;
        }else {
            short* action = agent_action(env.d_state[m_sid]);
            env.step(m_sid, action); // current state buffer
            agent_update(env.d_state[m_sid], env.d_state[m_sid ^ 1], env.d_reward);

            m_sid ^= 1;
            episode = m_episode;
            steps = m_steps;
        }}
    m_steps++;
    env.render(board, m_sid);
    return m_newepisode;
```

The provided parameters are just for reference.

$$\gamma = 0.9, \quad \alpha = 0.1, \quad 0.1 \leq \epsilon - \delta\epsilon \leq 1.0, \quad \delta\epsilon = 0.001$$

Submit only agent.cu file into the assignment.
Programming language: CUDA