# TCG Project 1 Report

110550088 李杰穎

September 30, 2022

## 1 Method & Implementation

In the given template, we already have a random slider, which will choose action randomly, regardless of board state. This method gets 65.6 points. Obviously, this is not a good results.

A better method is to try all four actions, and see which action will get higher score. I called this greedy slider. The implementation of vanilla greedy slider is shown as Code 1. This method gets 86.2 point, which is a huge improvement compared with random slider.

To further improve performance, I introduce a technique that I use when playing 2048, which is forbidden one of the directions, i.e. I never slide up when there is no need to do that. This will keep tiles on the bottom of board always be larger than those on top, making game harder to end. The implementation is shown as Code 2. After introducing this technique, the performance get better. This method gets 93.8 point.

Noteworthy, I discover that the forbidden direction actually influence the final score given the judge. After some experiments, I found that the highest score I can get is 94.7 points, when forbidden sliding down.

Finally, I further increase the number of forbidden actions to two, after some experiments, I found that the highest score I can get is 95 points, when forbidden slides up and left, in other word, the agent only slides down and right if those actions is valid. The implementation of this method is shown as Code 3.

Code 1: `class vanilla_greedy_slider` in `agent.h`

```
class vanilla_greedy_slider : public random_agent {
public:
   vanilla_greedy_slider(const
↪  std::string& args = "") : random_agent("name=slide role=slider " + args),
      opcode({ 0, 1, 2, 3 }) {}

   virtual action take_action(const board& before) {
      board::reward max_reward = -1;
      int max_op;
      for (int op : opcode) {
         board::reward reward = board(before).slide(op);
         if(reward >= max_reward){
            max_reward = reward;
            max_op = op;
```

```
            }
        }
        if(max_reward == -1){
            return action();
        } else {
            return action::slide(max_op);
        }
    }

private:
    std::array<int, 4> opcode;
};
```

Code 2: `class forbid_greedy_slider` in `agent.h`

```
class forbid_greedy_slider : public random_agent {
public:
    forbid_greedy_slider(const
↪   std::string& args = "") : random_agent("name=slide role=slider " + args),
        opcode({ 1, 2, 3 }), alter_opcode({ 0 }) {}

    virtual action take_action(const board& before) {
        board::reward max_reward = -1;
        int max_op;
        for (int op : opcode) {
            board::reward reward = board(before).slide(op);
            if(reward >= max_reward){
                max_reward = reward;
                max_op = op;
            }
        }
        if(max_reward == -1){
            max_reward = -1;
            for (int op : alter_opcode) {
                board::reward reward = board(before).slide(op);
                if(reward >= max_reward){
                    max_reward = reward;
                    max_op = op;
                }
            }
            if(max_reward != -1){
                return action::slide(max_op);
            }
            return action();
        } else {
            return action::slide(max_op);
        }
    }

private:
```

```
    std::array<int, 3> opcode;
    std::array<int, 1> alter_opcode;
};
```

Code 3: `class greedy_slider` in `agent.h`

```
class greedy_slider : public random_agent {
public:
    greedy_slider(const std::string& args = "") : random_agent("name=slide role=slider " + args),
        opcode({ 1, 2 }), alter_opcode({3, 0}) {}

    virtual action take_action(const board& before) {
        board::reward max_reward = -1;
        int max_op;
        for (int op : opcode) {
            board::reward reward = board(before).slide(op);
            if(reward >= max_reward){
                max_reward = reward;
                max_op = op;
            }
        }
        if(max_reward == -1){
            max_reward = -1;
            for (int op : alter_opcode) {
                board::reward reward = board(before).slide(op);
                if(reward >= max_reward){
                    max_reward = reward;
                    max_op = op;
                }
            }
            if(max_reward != -1){
                return action::slide(max_op);
            }
            return action();
        } else {
            return action::slide(max_op);
        }
    }

private:
    std::array<int, 2> opcode;
    std::array<int, 2> alter_opcode;
};
```

## 2 Conclusion

After experiments, we found that the method that can get highest points is to forbid two of the actions. This will make larger tiles stay in one of the four corners.