

# Theory of Computer Games 2022

## Report of Project #1

Student name: 劉子齊 Jonathan

Student ID: 311605004

In project 1, our goal is to implement a simple heuristics-based player. For the framework of the game, and the environment, I followed the given sample.

The original given method was to play randomly, but care nothing about the reward. In the following method which I implemented, I applied the backward method and an 8 x 4 tuple network. In my implementation, I calculate the corresponding reward of each operations, which are “slide up”, “slide down”, “slide left”, “slide right”. By calculating reward of each potential operations, I will choose the operation with the highest reward and act to it. The following figure is the implementation of my greedy sliding method.

```
179 class greedy_slider : public random_agent {
180 public:
181     greedy_slider(const std::string& args = "") : random_agent("name=greedy role=slider " + args),
182         opcode({ 0, 1, 2, 3 }) {}
183
184     virtual action take_action(const board& before) {
185         //std::shuffle(opcode.begin(), opcode.end(), engine);
186         board::reward max_reward = -1;
187         int best_op;
188         for (int op : opcode) {
189             board::reward reward = board(before).slide(op);
190             if(op == 0) {
191                 reward = 8 * reward + 1;
192             }
193             if(op == 1) {
194                 reward = 8 * reward + 1;
195             }
196             if(op == 2) {
197                 if(before[3][3] != 0)
198                     reward = 2 * reward + 1;
199             }
200             if(op == 3) {
201                 if(before[0][0] != 0)
202                     reward = 2 * reward + 1;
203             }
204
205             if(reward > max_reward) {
206                 max_reward = reward;
207                 best_op = op;
208             }
209         }
210         if(max_reward > -1) return action::slide(best_op);
211         return action();
212     }
213
214 private:
215     std::array<int, 4> opcode;
216 };
217
```

With my greedy sliding method introduced above, I obtained a way better result than the original strategy with the score of “92.3” in the provided Linux workstation, which is shown in the following figure.

```
[c311605004@tcglinux6 /tcgdisk]$ ls
0356168 0816013 0816137 109550031 110652019 311551059 311553015 311581004 all11574
0810816 0816049 0816169 109550039 111062513 311551069 311553039 311605004 pj-1-code-v1.zip
0810906 0816054 109511105 109550108 310540026 311551124 311554001 31554053 pj-1-judge-v1.zip
0811209 0816067 109550005 109550117 310551150 311551148 311554009 411581005 pj-2-judge-v1.zip
0813367 0816096 109550027 110550088 310555023 311551174 311554053 all11147 threes-judge
[c311605004@tcglinux6 /tcgdisk]$ cd 311605004/
[c311605004@tcglinux6 311605004]$ cd Theory-of-Computer-Games/
[c311605004@tcglinux6 Theory-of-Computer-Games]$ cd Project1/
[c311605004@tcglinux6 Project1]$ make clean
rm threes
[c311605004@tcglinux6 Project1]$ make
g++ -std=c++11 -O3 -g -Wall -fmessage-length=0 -o threes threes.cpp
[c311605004@tcglinux6 Project1]$ ./threes --total 1000 --save greedy.txt
Threes! Demo: ./threes --total 1000 --save greedy.txt

1000    avg = 779, max = 7110, ops = 2073525 (145261115465062)
      12    100%    (5%)
      24    95%    (25%)
      48    70%    (38.9%)
      96    31.1%  (23.8%)
     192    7.3%   (7.2%)
     384    0.1%   (0.1%)

[c311605004@tcglinux6 Project1]$ ./threes-judge --total 1000 --load greedy.txt
Threes! Judge: ./threes-judge --total 1000 --load greedy.txt

1000    avg = 779, max = 7110, ops = 2073525 (145261115465062)
      12    100%    (5%)
      24    95%    (25%)
      48    70%    (38.9%)
      96    31.1%  (23.8%)
     192    7.3%   (7.2%)
     384    0.1%   (0.1%)

Judging the actions... Passed
Judging the speed... Passed, expected 76988 ops
Assessment: 92.3 points
[c311605004@tcglinux6 Project1]$
```