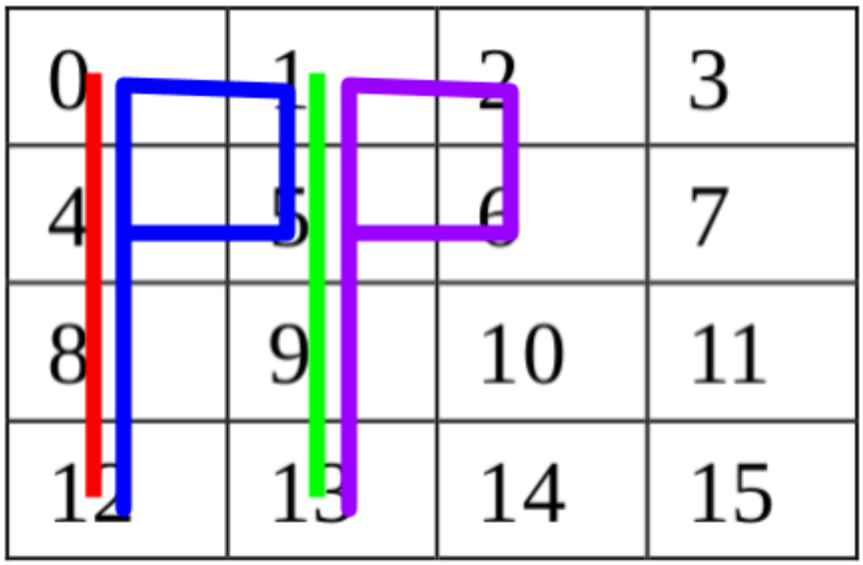
1. **Describe how you implement AI::get_best_move() :**
   simple algorithm:
   In four directions, validate each direction. if valid, update the state's score with bigger result and direction; If invalid, just skip to next direction. Therefore, the function will return the best direction depend on its afterstate's score.
2. **Describe how you implement AI::update_tuple_values() :**

$$\textbf{TD(0):}\ V(s_t) = V(s_t) + \alpha(r_{t+1} + V(s_{t+1}) - V(s_t))$$

   nothing but to implement the function above:
   error = (t+1) reward + (t+1) score - (t) score
   alpha = 0.0025
   (t) V update toward (t+1) V with speed alpha.
   (note that in last experience, the reward should be 0.0)
3. **Statistic Charts**
   a. **standard tuple**
      Winning Rate: 95.36% (train 100k times)
      Average Score: 54472.7



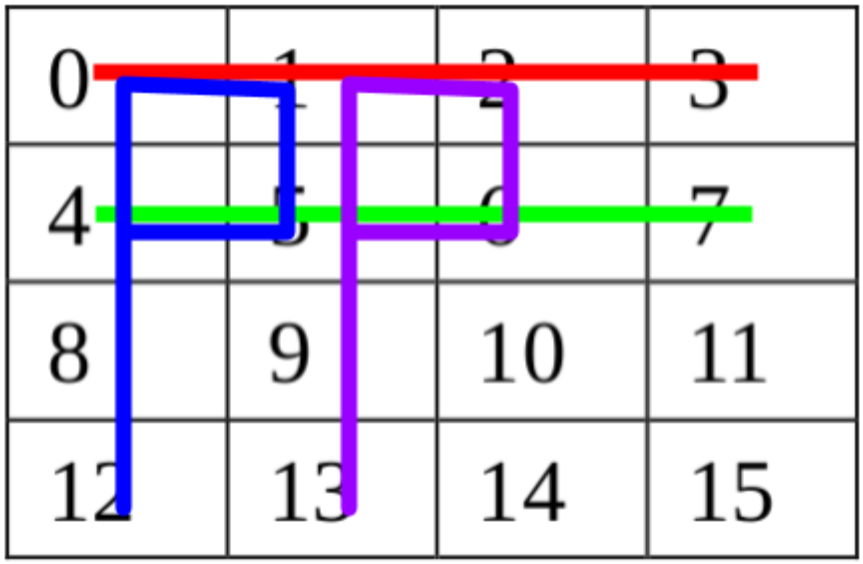
```
mean = 54472.7   max = 112440
512     0.4%    (0.4%)
1024    3.3%    (3.7%)
2048    31.5%   (35.2%)
4096    64.2%   (99.4%)
8192    0.6%    (100%)
```

```
jay_lab@jay:~/Desktop/dlc/lab6$ ./2048_Test
4-tuple pattern 048c initialized, size = 65536 (64K)
4-tuple pattern 159d initialized, size = 65536 (64K)
6-tuple pattern 01458c initialized, size = 16777216 (16M)
6-tuple pattern 12569d initialized, size = 16777216 (16M)
4-tuple pattern 048c is loaded from ori_100k.weight
4-tuple pattern 159d is loaded from ori_100k.weight
6-tuple pattern 01458c is loaded from ori_100k.weight
6-tuple pattern 12569d is loaded from ori_100k.weight
Success Rate:   0.9536
```

b. your tuple

Winning Rate: 95.94% (train 100k times)

Average Score: 53404.9




```
mean = 53404.9   max = 128944
512     0.3%    (0.3%)
1024    4.8%    (5.1%)
2048    35%     (40.1%)
4096    59%     (99.1%)
8192    0.9%    (100%)
```

```
jay_lab@jay:~/Desktop/dlc/lab6$ ./2048_Test
4-tuple pattern 0123 initialized, size = 65536 (64K)
4-tuple pattern 4567 initialized, size = 65536 (64K)
6-tuple pattern 01458c initialized, size = 16777216 (16M)
6-tuple pattern 12569d initialized, size = 16777216 (16M)
4-tuple pattern 0123 is loaded from new2_100k.weight
4-tuple pattern 4567 is loaded from new2_100k.weight
6-tuple pattern 01458c is loaded from new2_100k.weight
6-tuple pattern 12569d is loaded from new2_100k.weight
Success Rate:   0.9594
```

4. Discussion

I also try with other tuple settings, but most of them performs really bad. In one of my designed tuple, 4 patterns with 7 tiles for each, I train them with 100k iterations, which only result of about 50% winning rate. I am not quite sure why the result is so bad, maybe the amount of parameters is too large to train, which need more iterations to converge.

And, in this assignment, the most challenge part should be tracing the code, I can't read them all without well comments. Thanks to TA's detailed explaination on this assignment or I could not complete it.