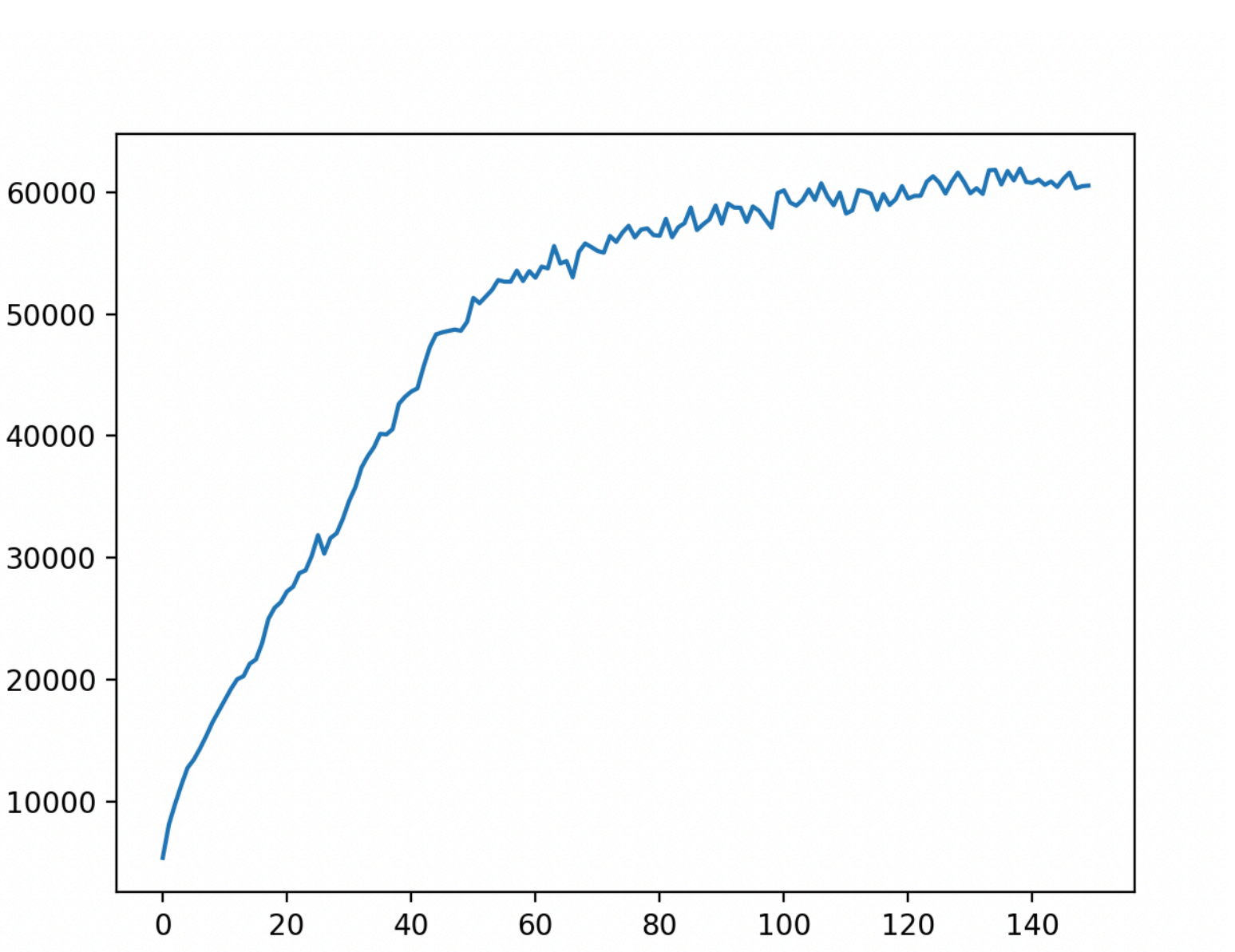
Deep Learning Lab2

周睦鈞

311553060

---

1. A plot shows scores (mean) of at least 100k training episodes



2. Describe the implementation and the usage of n-tuple network

2048 的 board 有 16 格，若想要計算所有 state 的 V(S) 話，state 的數量高達 16^{16} 會花費大量的記憶體

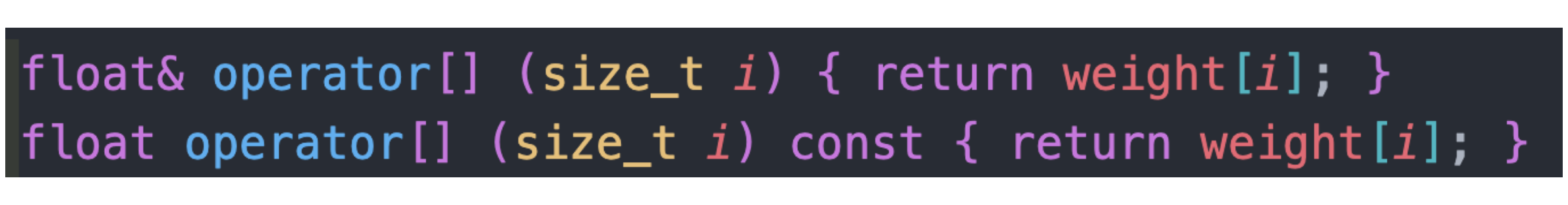
因此我們使用 n-tuple network 讓每一種 pattern 的 8個 isomorphic pattern 來代表目前 board 的盤面狀況，並且利用該盤面狀況去 look up weight table 來得到該 pattern 所代表的總值 ，最後將所有 pattern 的總值都加起來就可以代表該 state 的 V(S) 值

```
for (int i = 0; i < 8; i++) {
    board idx = 0xfedcba9876543210ull;
    if (i >= 4) idx.mirror();        You, 2 週前
    idx.rotate(i);
    for (int t : p) {
        isomorphic[i].push_back(idx.at(t));
    }
}
```

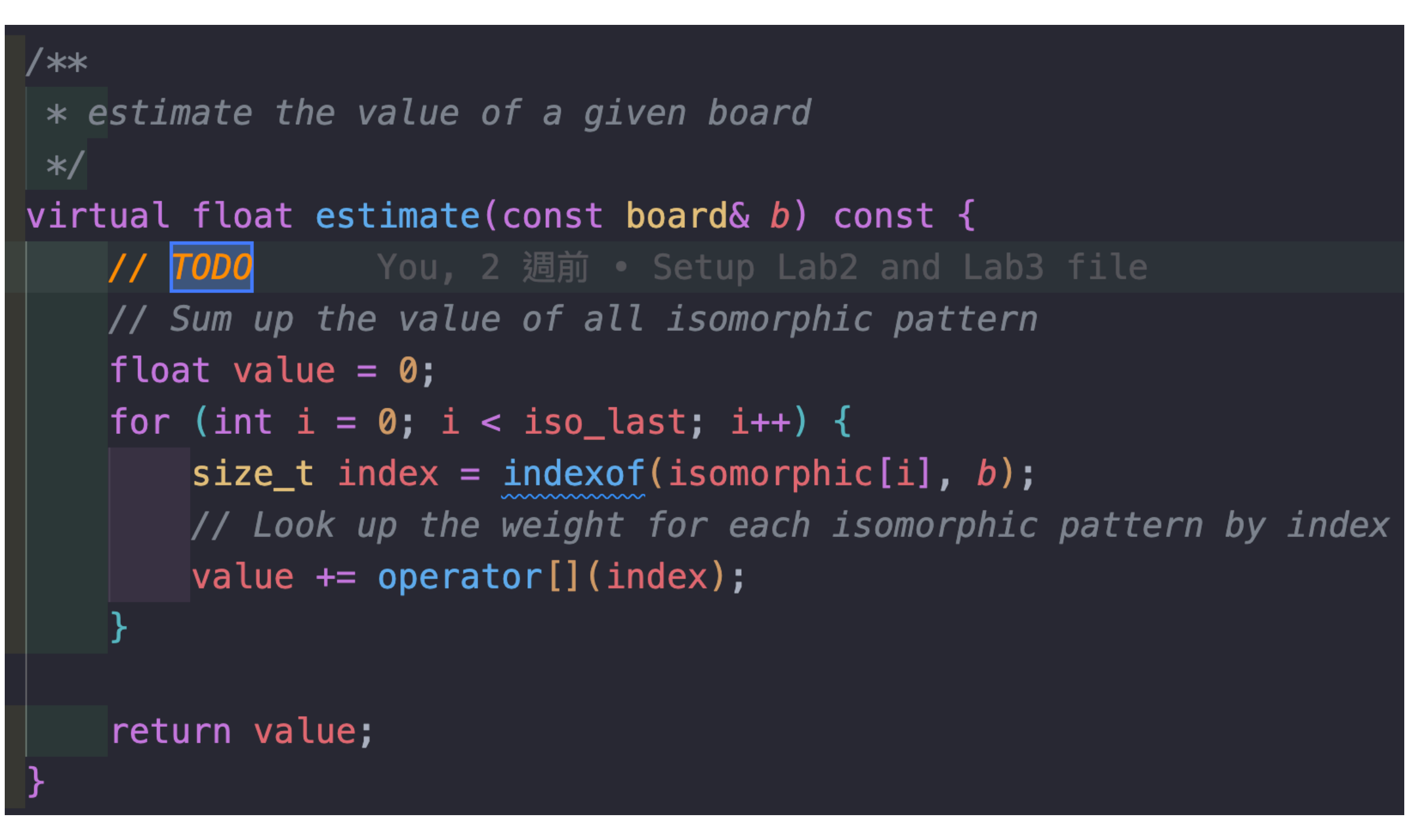
上面的 Code 會取得 pattern 的所有 isomorphic pattern 各自在 board 上的 index

```
size_t indexof(const std::vector<int>& patt, const board& b) const {
    // TODO
    // Return b.at(patt[len - 1]) | b.at(patt[len - 2]) | ... | b.at(patt[1]) | b.at(patt[0])
    // as an index for look up table
    size_t index = 0;
    for (size_t i = 0; i < patt.size(); i++) {
        index |= b.at(patt[i]) << (4 * i);
    }

    return index;
}      You, 2 週前 • Setup Lab2 and Lab3 file
```

利用 indexof function 可以取得 isomorphic pattern 對應 board 上的 index 取得一組值 ，該值後續可以當作我們要在 weight table lookup 的 key



```
float& operator[] (size_t i) { return weight[i]; }
float operator[] (size_t i) const { return weight[i]; }
```



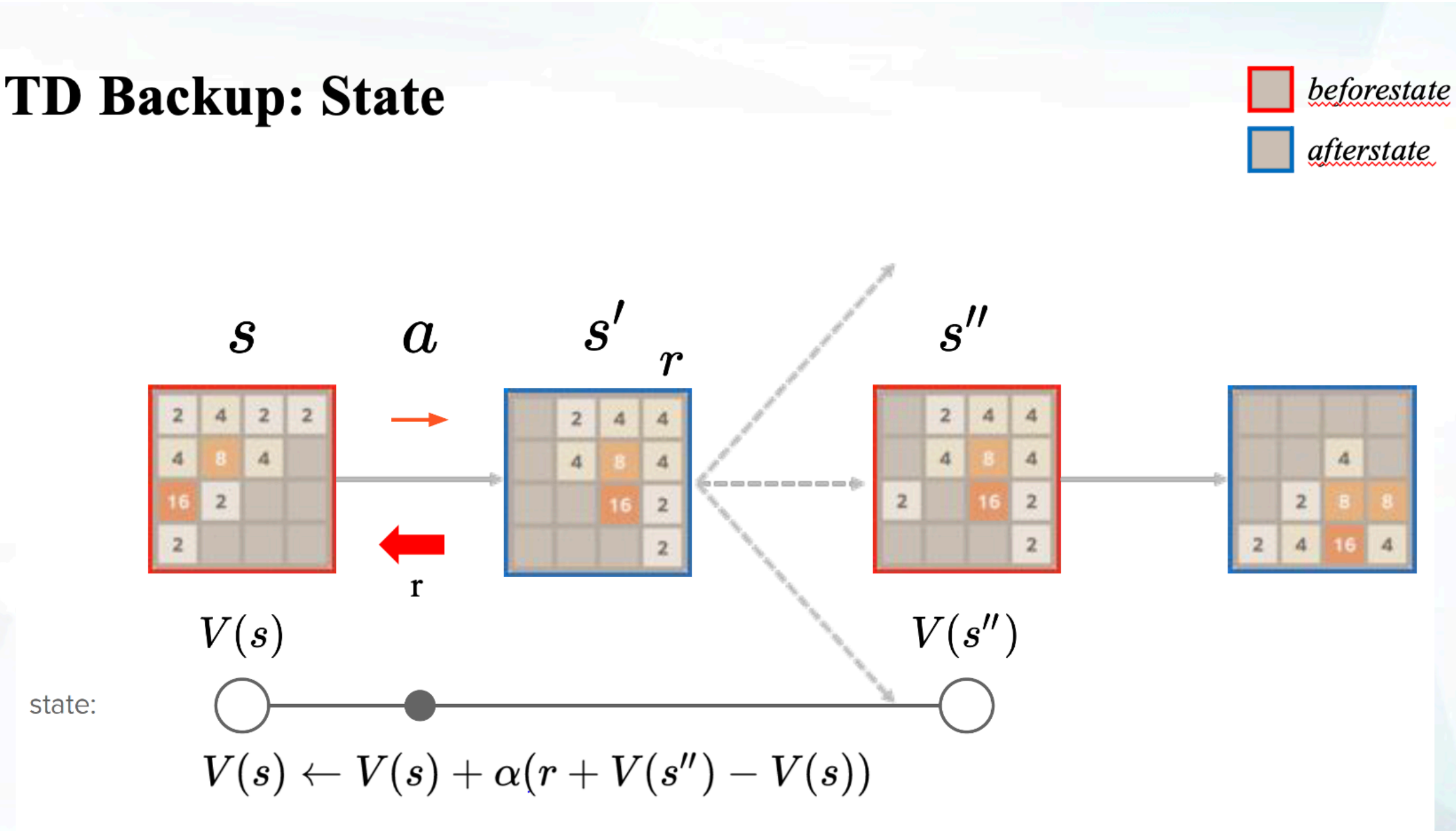
```
/**
 * estimate the value of a given board
 */
virtual float estimate(const board& b) const {
    // TODO      You, 2 週前 • Setup Lab2 and Lab3 file
    // Sum up the value of all isomorphic pattern
    float value = 0;
    for (int i = 0; i < iso_last; i++) {
        size_t index = indexof(isomorphic[i], b);
        // Look up the weight for each isomorphic pattern by index
        value += operator[](index);
    }

    return value;
}
```

上面的 Code 會得到 pattern 的 weight , 方法是透過該 pattern 的所有 isomorphic pattern 利用 key - value 對應關係將各自的 index 去 weight table lookup 最後將所有查到的 weight 加總其來

3. Explain the mechanism of TD(0)

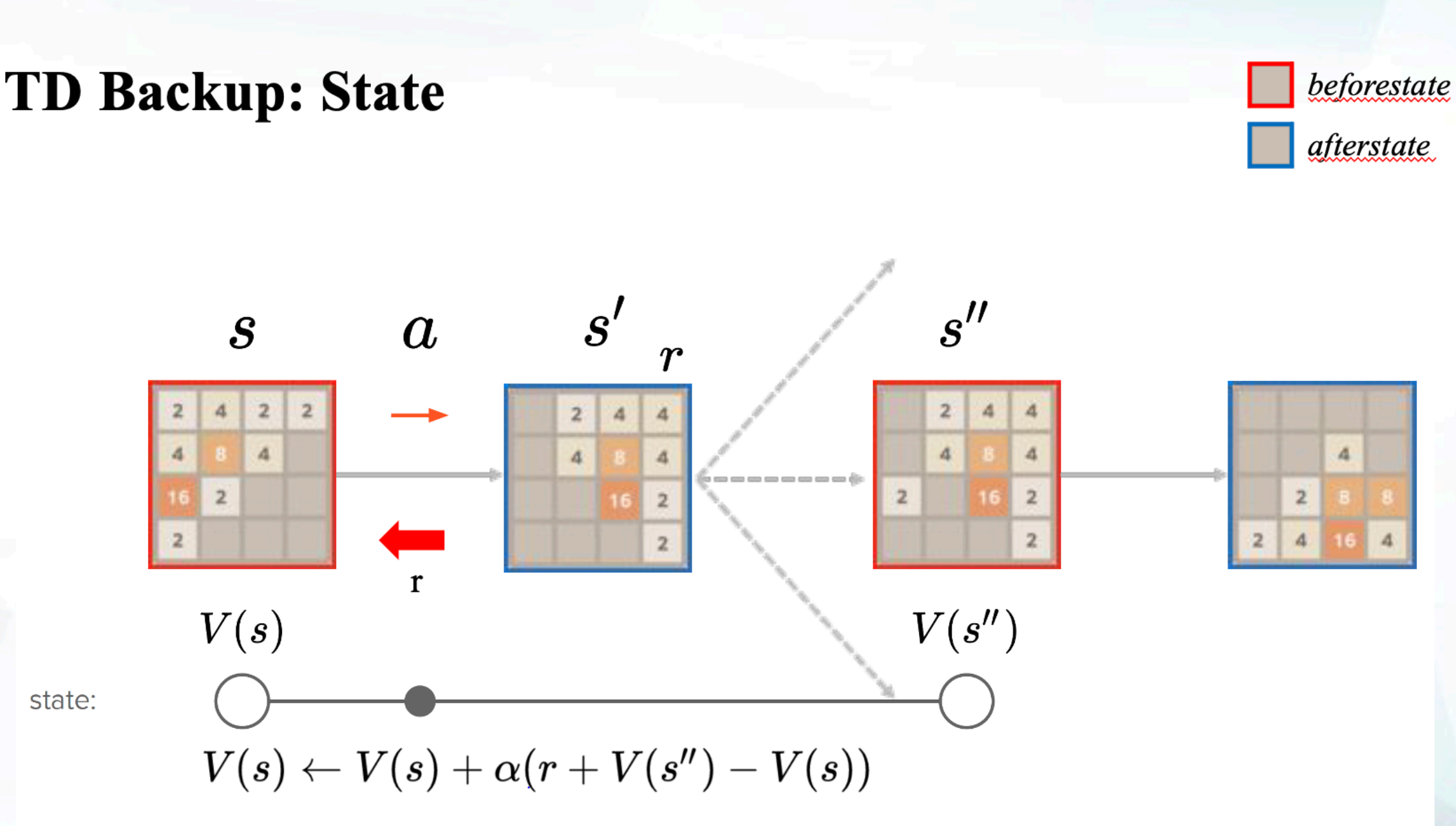
TD(0) 的機制是會先進行一場遊戲，當遊戲結束後會從倒數第二個 state 開始往回更新，一直重複上述動作直到收斂

TD(0) 在更新 state 的時候，是利用該 state 和下一個 state 去計算 TD_error 並且更新當前的 state



4. Describe your implementation in detail including action selection and TD-backup diagram

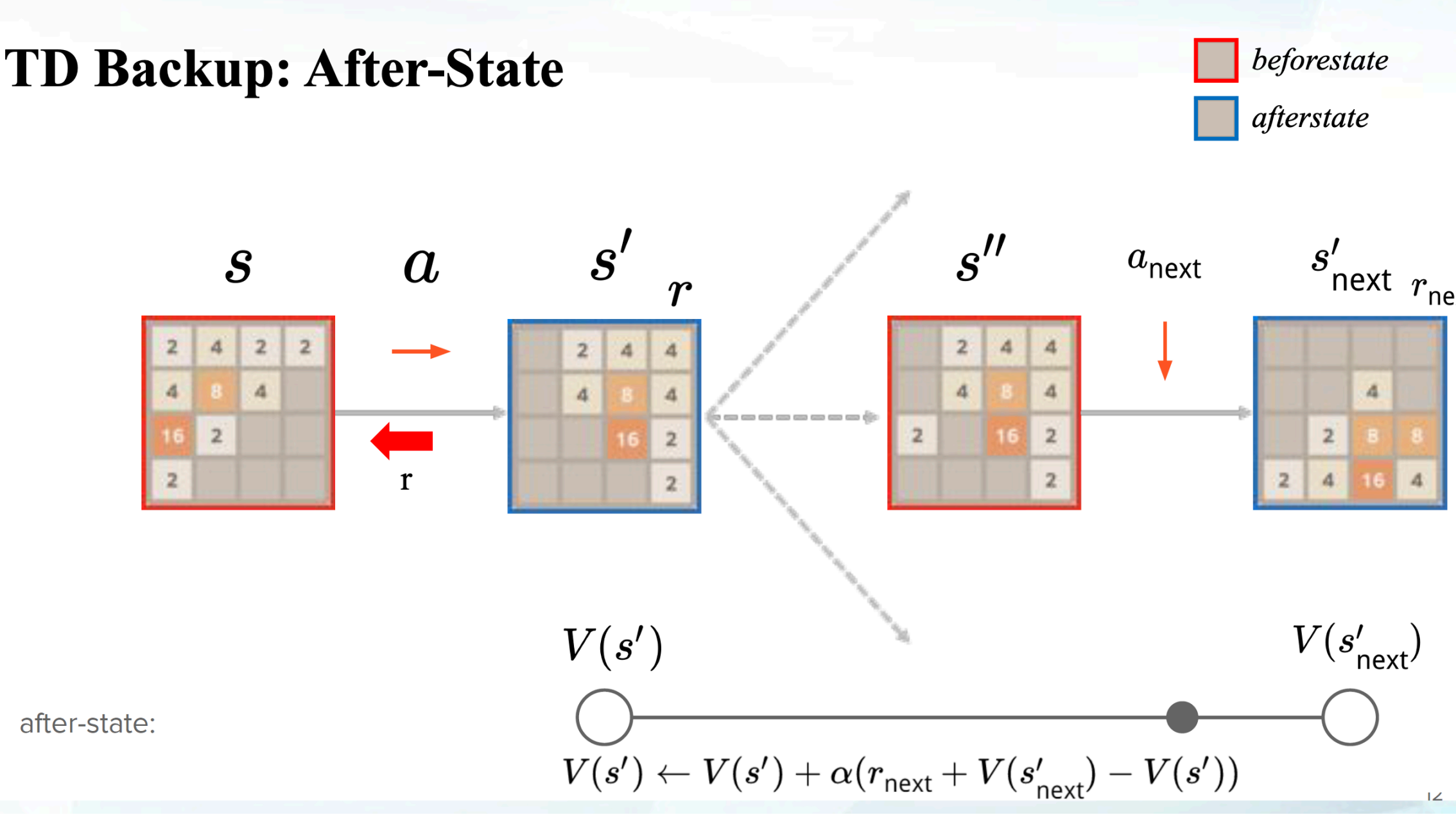
a - 1. TD-backup diagram (Before state):



利用 Before state 做 backup: 當我們在該 state 經過移動後，會得到 reward value , 同時也會有新有 tile popup 形成新的 before state

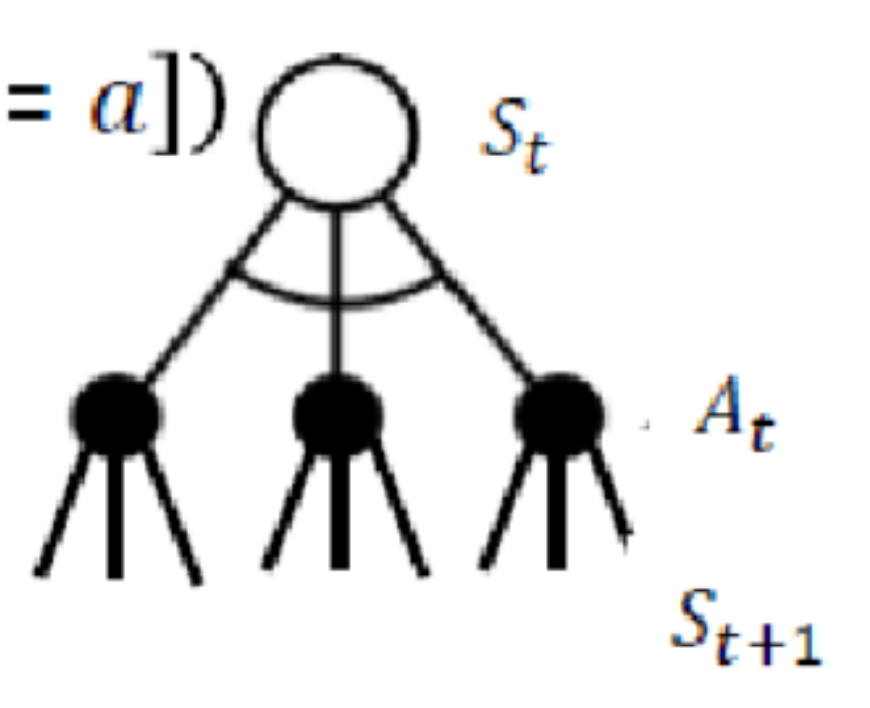
更新的時候我們會利用 TD_error 來更新當前的 state , 也就是我們期望 reward 加上下一個 before state 的 value function 當作 TD_target , 並且用來更新當前 before state 的 value function

a - 2. TD-backup diagram (After state):



利用 After-state 去做 backup : 方法和 before state 差不多，差別在於我們在做更新的時候不考慮環境 popup 新的 tile ，利用當前的 after state 去和下一個 after state 去做更新 , 也就是我們期望下一個 move 的 reward 加上 下一個 after state 的 value function 當作 TD_target , 用來更新當前的 after state value function

b. Action selection diagram



由於我們使用 before state 來 backup , 因此在經過移動後，會因為產生新的 popup tile (2 或 4) 因此會有多個 S_{t+1}

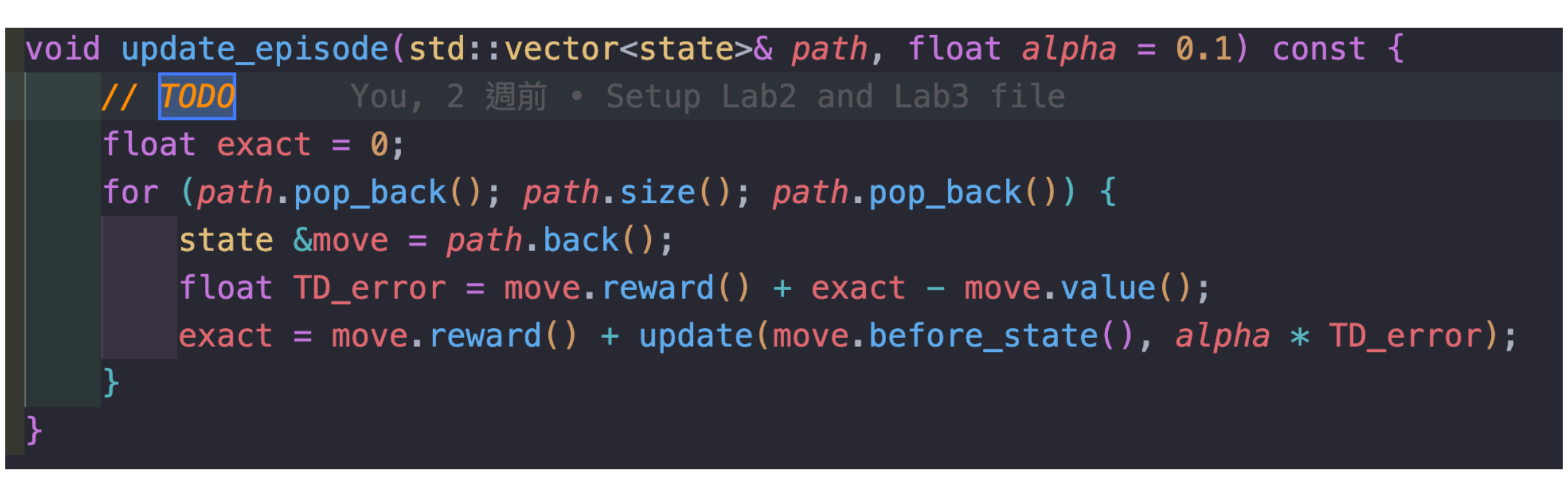
所以在計算哪一個 move 是最好的話，我們需要模擬該 move 的期望值也就是下一個 before state 的加權平均 value function 是最高的

c. Implementation Detail

```
state select_best_move(const board& b) const {
    state after[4] = { 0, 1, 2, 3 }; // up, right, down, left
    state* best = after;
    float best_total = -std::numeric_limits<float>::max();

    for (state* move = after; move != after + 4; move++) {
        if (move->assign(b)) {
            // TODO      You, 2 週前 • Setup Lab2 and Lab3 file
            // Find empty tile
            board after_state = move->after_state();
            int space[16], num = 0;
            for (int i = 0; i < 16; i++) {
                if (after_state.at(i) == 0) {
                    space[num++] = i;
                }
            }

            // Set value for before state
            move->set_value(estimate(move->before_state()));
```

```
            // Avg before state value: R + V(S')
            float total = move->reward();
            for (int i = 0; i < num; i++) {
                board *tmp = new board(uint64_t(after_state));
                // before state for new popup tile: 2
                tmp->set(space[i], 1);
                total += 0.9f * estimate(*tmp) / num;

                // before state for new popup tile: 4
                tmp->set(space[i], 2);
                total += 0.1f * estimate(*tmp) / num;

                delete tmp;
            }

            if (total > best_total){
                best = move;
                best_total = total;
            }
        } else {
            move->set_value(-std::numeric_limits<float>::max());
        }
        debug << "test " << *move;
    }
    return *best;
}
```

如上圖的 action diagram，需要先計算 move 後的 after state , 並且需要知道在該 board 下有哪些地方是空的可以填入新的 tile , 因此會把能填入新的 tile 的 board index 記錄下來，之後透過加權平均的方式去計算各種可能的 S_{t+1} 之期望值以及該 move 後所得到的 reward



```
void update_episode(std::vector<state>& path, float alpha = 0.1) const {
    // TODO      You, 2 週前 • Setup Lab2 and Lab3 file
    float exact = 0;
    for (path.pop_back(); path.size(); path.pop_back()) {
        state &move = path.back();
        float TD_error = move.reward() + exact - move.value();
        exact = move.reward() + update(move.before_state(), alpha * TD_error);
    }
}
```

在做 TD backup , 會將這個 episode 走過的 path 從倒數第二個 move 開始，計算該 move 的 state 和下一個 state 的 TD_error , 並且更新該 state 的期望值，之後就可以利用 state 的期望值，再算出 TD_target 也就是 exact 來對上一個 state 做一樣的更新

```
/**
 * update the value of a given board, and return its updated value
 */
virtual float update(const board& b, float u) {
    // TODO      You, 2 週前 • Setup Lab2 and Lab3 file
    float u_avg = u / iso_last;
    float value = 0;

    // Update all isomorphic pattern with average TD-error
    for (int i = 0; i < iso_last; i++) {
        size_t index = indexof(isomorphic[i], b);
        operator[](index) += u_avg;
        value += operator[](index);
    }

    return value;
}
```

當我們拿到 TD_error 時，我們需要對 weight table 做更新，我們會將 error 平均分給 8 個 isomorphic pattern ，所以需要先知道各個 isomorphic pattern 所對應的 index , 並且利用 key - value 關係去 weight table 找到該 isomorphic pattern 代表的 weight 做更新並且回傳更新過後的 weight 總和

5. Improvement or discuss

```
tdl.add_feature(new pattern({1, 4, 9, 14}));
tdl.add_feature(new pattern({0, 5, 8, 10}));
tdl.add_feature(new pattern({1, 6, 9, 11}));
tdl.add_feature(new pattern({1, 4, 5, 6, 9, 13}));
```

除了助教提供的兩種 6-tuple pattern , 我額外新增了三種不同的 pattern , 發現增加以後可以讓 performance 從原本的 88% 可以上升到 92%左右

其中第一行是一個歪歪的拐杖，第二和第三行是類似 ”入” ，最後一行的是一個十字架的圖案