Architecture

(1) Prepare for the grid

```
1 class Grid:
```

Copy a grid

```
1 def clone(self):
```

Set Value for one cell.

```
1 def setValue(self, pos, num):
```

Calculate and take down empty cells.

```
1 def EmptyCells(self):
```

Combine 2 cells whose values are same.

```
1 def merge(self, cells):
```

Move operation

```
1 def move(self, dir):
```

Return steps that can take for next.

```
1 def PossibleSteps(self, dirs=vecIndex):
```

(2) Minimax function

```
1 class Play2048(Grid):
```

Return the stage after moved.

```
1 def NextStage(self, stage):
```

How will Max choose his move

```
1   def MAX(self, stage, Alpha, Beta):
```

How will min choose his move.

```
1   def MIN(self, stage, Alpha, Beta):
```

Evaluate the performance.

```
1   def CALCULATE(self, stage):
```

Search

(1) I use Minimax algorithm to perform the game.
Alpha Beta pruning is also applied on it to achieve a better performance.

(2) When searching for a good heuristics, I looked through a lot of resource on the Internet. In the end, I choose Monotonicity, Smoothness, Free Tiles, and the number of cells that can be merged as attributes.
Monotonicity can ensure that the value of the box increases or decreases along the up, down, left and right directions. Squares with larger values should gather in a corner. This will help prevent the blocks with small values from being isolated, and will also keep the panel well organized, so that the blocks with small values are gradually stacked and gradually merged into the blocks with large values.
Smoothness can measure the difference between adjacent grid values and try to reduce the difference.
Free Tails mean the number of the empty cells on the grid. The larger the better.
The number of cells ensure that can be merged is also the larger the better.

When consider the weight of these attributes, I try a lot of time to get the best weight. when adding and reducing the value of weight of Smoothness, I find that there is a value that has a best performance, which is 0.01

Challenges

When calculate what will Max player and Min player do, it is hard to decide when to calculate the leaf node.
Also when processing alpha and beta pruning, it is hard to determine which value should be compared with alpha and beta, and when is an appropriate time to renew alpha and beta.

Weaknesses

It is still not the best performance of 2048. The calculation of the distance between empty cells is missed.
The best weight of monotonicity , and Free Tiles is not found out, so there might be a better weight to get a better performance.