TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology

ITI40LT
Stanislav Frolov 123971IAPB

SOLUTION STRATEGIES FOR A GAME 2048.

Bachelor's thesis

Supervisor: Marko Kääramees
PhD
Associate Professor.

Tallinn 2016

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

ITI40LT
Stanislav Frolov 123971IAPB

MÄNGU 2048 LAHENDAMISSTRATEEGIAD

Bakalaureusetöö

Juhendaja: Marko Kääramees
PhD
Dotsent

Authors declaration of authority

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Stanislav Frolov
10.12.2016

Abstract

The objective of this thesis is to suggest multiple working algorithms for a game named 2048, which can complete the objective of the game. To explore already existing solutions created by individuals from around the world, employ these ideas while creating personal AI algorithms.

The second objective of the work is to analyse different solution methods, choose the best methods and specify what the key parameters of each method are.
The comparison and analysis of the methods proposed is provided and it is shown how it is related to the work done by others.

Taking into consideration the value of each cell, aswell as each cells weight attribute turned out to be a required aspect for a winning algorithm. Methods which held one corner in highest priority aswell as methods considering all corners of equal priority performed with a similar win rate. Methods with all direction calculation performed almost 3 times slower than the three direction methods.

The thesis is in English and contains 60 pages of text, 6 chapters, 11 figures, 3 tables.

Annotatsioon

Selle lõputöö eesmärk on välja pakkuda mittu töötavaid algoritmi mängu jäoks nimega 2048, mis suudab selle mängu eesmärki saavutada. Lahti kirjutada juba eksisteerivad lahendused mis on loodud üksikisikudega maailmast, võtta kasutusele need ideid et luua personaalseid tehisintellekti algoritmid.

Sekundaarsed eesmärgid on analiseerida erinevaid lahendus meetodid, valida kõige parimaid nendest ja täpsustada missugused tähtsad parameetrid igal meetodil on. Võrdlus ja analüüs pakutavast meetodist on ette antud, on näha kuidas see on seotud tööga tehtud teistega.

Iga asukoht mängu väljal, ning selle kaal välja tuli nõutavaks aspektiks võidava algoritmi jäoks. Meetodid, mis hoidsid ühte nurga suuremal prioriteedil samuti need meetodid millel kõik nurgad on võrdse prioriteediga, omandasid sarnase võidu šanssi. Meetodid kõige suuna arvutamisega töötasid peaaegu kolm korda aeglasem kui meetodid kolme suuna arvutamisega.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 60 leheküljel, 6 peatükki, 11 joonist, 3 tabelit.

Table of contents

Contents

List of pictures

List of tables

Introduction

Explanation of game 2048 taken from Wikipedia [10]: 2048 is a game played on a 4 by 4 cells field. Player can move all cells in any direction, each cell stops moving if it touches the edge of the game field or another cell with a value different from its own. When two cells with the same value collide: they merge into a new cell with double the value of the previous one. After each move a player makes, a new cell with value 2 or 4 is randomly placed on an empty cell. When a player cannot make any more moves, the game is lost. When a cell with 2048 is made the game is considered won.

Example how game is played. The figure 1 illustrates in initial state:

Figure 1 Initial state

Moving to the right from previous game field results in:

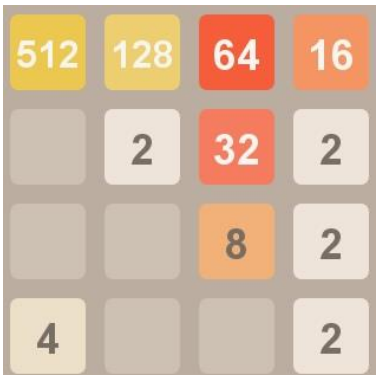

Figure 2 Move to the right

Next move upward:

Figure 3 Move up

And left:

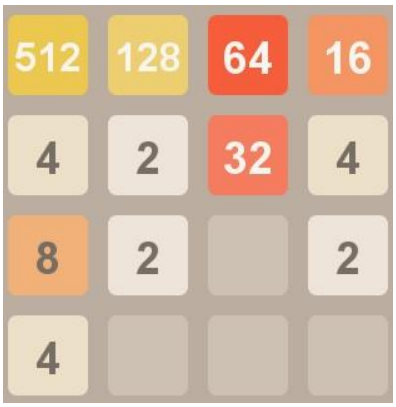

Figure 4 Moving left

Most important objective for this thesis is to program AI(Artificial Intelligence) which can win in 2048. Secondary objectives are creating different strategies, comparing them to each other. Strategies used by other individuals will be used as ideas and comparison for personally created AI.

Java programming language is used to implement the methods of solving the task.

In chapter 1 terms used by algorithms are explained, solutions used by others are described. Chapter 2 presents algorithms used as solutions, explanations how they work.

Chapter 3 contains results produced by all created methods, comparisons between most successful methods. Chapter concludes the thesis, mentions its objectives and what was done, and describes results achieved by created strategies. Appendix has all the field state calculation examples for each method developed during course work.

1 Background.

The chapter describes which ideas and algorithms already existed before this thesis was started. 2048 is a simple but popular game for which people were interested to write their own AI strategies for winning this game. Terms and alpha-beta/minimax algorithms used as basis will be explained with picture examples in this section.

1.1 Minimax algorithm.

Minimax algorithm can used for building an AI (Artificial Intelligence) for two-player board games with alternating moves. It is a strategy where maximum node represents best move to make for highest gain while minimum is the worst move which opponent can make. Algorithm goes through all possible move variations this way.

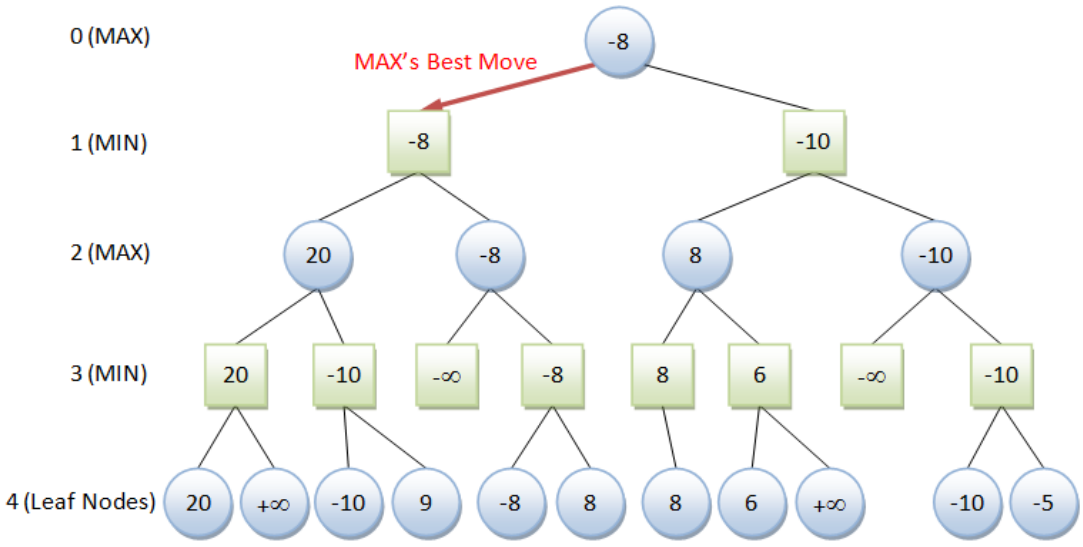

Figure 5 Minimax [7].

1.2 Alpha beta algorithm.

According to the explanation from [6] this strategy adds to the minimax in a way that it can leave out some branches unchecked when a certain condition is met. There are two additional values: **alpha** which is best case maximum value a node has, **beta** is best case minimum value. When a maximum node above a minimum node has a higher alpha value than the child node beta value, then all other unchecked branches from the

minimum node are ignored. And vice versa when a minimum node above a maximum node has a lower beta value than the child node alpha value, rest of child node branches are ignored.

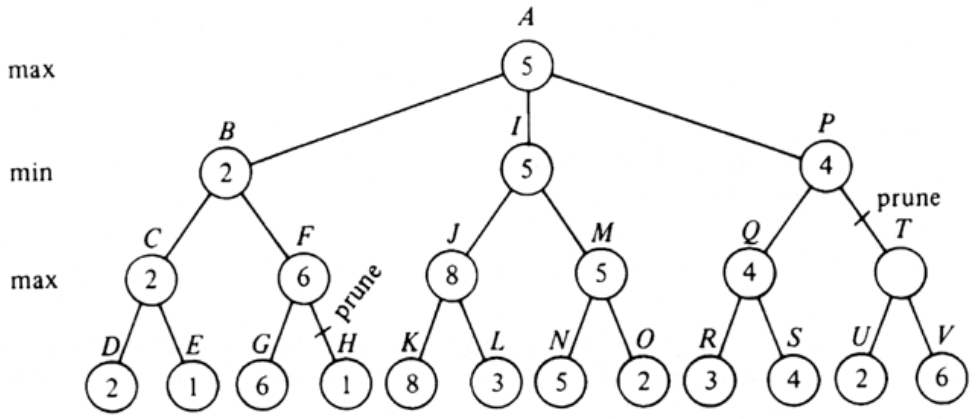

Figure 6 Alpha Beta [8].

1.3 Expectimax search algorithm.

As it is explained on Stack overflow forum [1], algorithm is using a "expectimax" search, meaning a recursive search alternating between (all possible variations of random tile spawning and the value of probability of each possible board situation) as in "expectation", and choosing the move which has the highest score value as in "maximization".

Two simple heuristics are used as mentioned on Stack overflow forum [1] at the start of development: bonus for a number of empty spaces and higher values positioned on the edge.

Additional heuristics were implemented by Petr Morávek [1]: 1) a score penalty for non-monotonic rows or columns, the higher values in such row or column the higher penalty would be. 2) Counting the amount of possible merges and amount of empty spaces.

Implementation of this strategy is explained on Stack overflow [1] as: the board of this game is encoded as a single 64 bit integer. Bit shift operations extract a single row or column. Move results for each row or column are stored in a table, when a move is made, then 4 lookups to that table produce the result of moving in a direction. A

example of an entry "1122 -> 0023" shows a row of “2244” turning into “0048” if moving right.

Scoring uses table lookup as well. Table contains heuristic scores for each row or column variation, the total score for a board situation is the sum of table values for all rows/columns.

1.4 Clustering.

When cells with similar values are close to each other, it is easier to merge cells with equal values and hard to lose the game. That means the board is clustered, if values are not in the before mentioned positions, then it is a worse situation. A clustered board is preferred. As explained on [2]

1.5 Monotonicity.

Monotonicity meaning, as described on [5] when tiles decrease in value the further it is from a corner. Largest valued cell should be in a certain corner, and all other cells should be clustered around the large tile, according to value. The farther from the largest tile you get, the lower each tile should be in value.

If a cell has 0 values: it is not compared to the other nodes, only values which are equal or bigger than 2 are compared with each other for monotonicity.

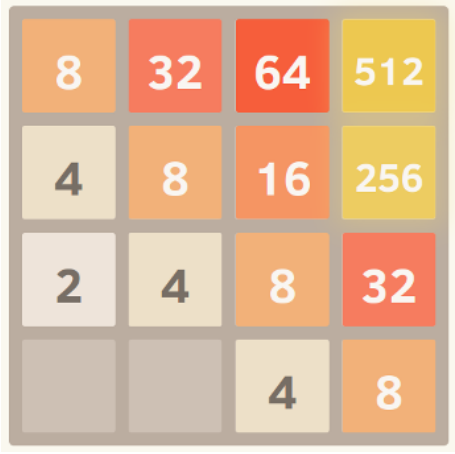

Figure 7 Example of monotonicity [9].

1.6 Smoothness.

Smoothness is commented on [1] that the closer in value are next to each other cells the smoother the board is.

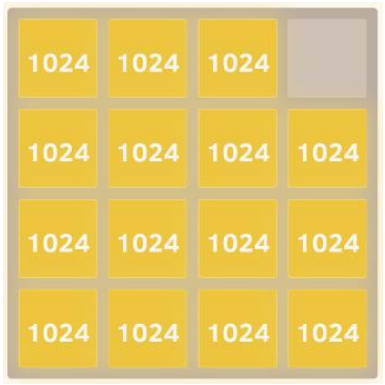

Figure 8 Perfectly smooth grid example [9].

1.6 Clustering, monotonicity and smoothness algorithm.

Cells with similar values are kept close to each other to make merging them easier. As Vasilis Vryniotis [2] suggests that cells with higher values should appear on the sides or corners of the board, not in the middle of it.

There are two players taking turns in this game, first player plays the game making moves in directions, second player is considered the random element of block spawning after each move by player one.

This strategy uses clustering score, monotonicity, smoothness and the amount of free cells on the board heuristics.

Performance of algorithm: depth of 3 search takes less than 0,05 seconds with 20% win rate, 5 depth takes 1 second with 40% win rate, 7 depth takes 27-28 seconds with 70-80% win rate.

1.7 Weight matrix and probability algorithm.

This strategy created by Yiyuan Lee [3] uses recursion to employ depth search for the best move.

Here the chance of a random block appearing in each field influences the value for each move.

To calculate the score at one game state a function is used: sum of all probabilities to create each possible set of game states multiplied by maximum numerical score of a game state.

Recursion stops calculating in two possibilities: 1) when a situation is reached that AI can no longer make a move. 2) When a recursion depth limit is reached.

And a weight matrix is used here so that as mentioned by Yiyuan Lee [3] “the bigger tile is pushed near any one corner and the smaller tiles away from that corner.”

Explanation about weight matrix taken from [3] was “An optimization search carried out using randomly generated, diagonally monotone decreasing weight matrices produces the following matrix as the most optimal.”

$$\begin{bmatrix} 0.135 & 0.121 & 0.102 & 0.0999 \\ 0.0997 & 0.088 & 0.076 & 0.0724 \\ 0.0606 & 0.0562 & 0.0371 & 0.0161 \\ 0.0125 & 0.0099 & 0.0057 & 0.0033 \end{bmatrix}$$

Results of this strategy were: with depth of 6 a 4096 tile appeared more than 40% of the time, depth 8 produced 8192 tile more than 30% of the time.

2 Solution.

In order to add best move depth search, minimax or alpha beta algorithm without pruning is used as a basis. Also, instead of the biggest/smallest value selection for each branch - the average value for each branch on each level is calculated because of random element presence.

2.1 General Algorithm.

As mentioned on [4] not to make a move in the direction opposite of a largest file, for example down for an upper left or right tile.

This strategy uses depth search to choose the best move based on the average score for each direction (left, right, up).

As explained on [5], it is required for the highest value block to be in an upper left corner and cell values are required to be smaller the further away from the corner they are. The way score for each direction separately is calculated: a move is made into that direction, then for each empty space a value of 2 or 4 is placed in two separate loops and a move is made in each direction counted as a separate call to method which returns the total score of the board when a certain depth is reached. This way on each depth level the average value of each direction is calculated and returned to the previous depth.

The matrix below shows for each **field** location what x and y location values it has.

$$\begin{bmatrix} x=1\,y=1 & x=2\,y=1 & x=3\,y=1 & x=4\,y=1 \\ x=1\,y=2 & x=2\,y=2 & x=3\,y=2 & x=4\,y=2 \\ x=1\,y=3 & x=2\,y=3 & x=3\,y=3 & x=4\,y=3 \\ x=1\,y=4 & x=2\,y=4 & x=3\,y=4 & x=4\,y=4 \end{bmatrix}$$

Exceptions

These exceptions are added to the algorithm ignoring best score of a direction and are used by certain methods.

1. When any cell in first row except the most left one is empty and there are cells available which can fill that space, move in the upper direction is made.

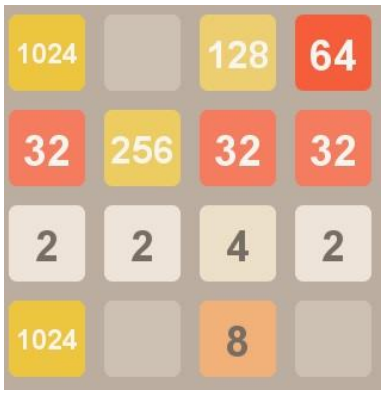

Figure 9 Condition for exception 1 example

2. When the upper- left cell is empty and there are filled cells on the right in the first row, move to the left is made.

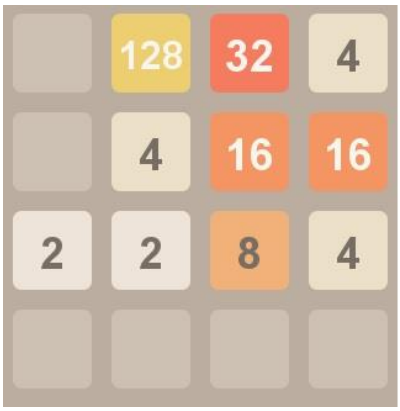

Figure 10 Condition for exception 2 example

```
While (game is not won or lost) {
  For (each direction [left; right; up]) {
    If (can move in direction) {
      direction_value = Calculate_score (direction, depth = 1)
  Make move in direction with the highest direction_value.
}
Function Calculate_score (direction, depth) {
    If (game is won) return infinity;
    If (cannot make a move) return –infinity;
    If current depth is smaller than target depth {
      If move in direction is possible {
        Move in direction
        For each empty_space {
          empty_space = 2
          For (each direction [left; right; up]) {
            If (can move in direction) {
              Total_score += Calculate_score (direction, depth + 1)
              call_method_number += 1
              Restore board to state after move was made.
      If move in direction is possible {
        Move in direction
        For each empty_space {
          empty_space = 4
          For (each direction [left; right; up]) {
            If (can move in direction) {
              Total_score += Calculate_score (direction, depth + 1)
              call_method_number += 1
              Restore board layout to state after move was made.
      Total_score = Total_score / call_method_number
      Return Total_score
    } else {
      Return current_field_score
```

Figure 11 three direction algorithm pseudo code

Different methods for calculating the current field score at the final depth of search are described in the following

2.2 3-direction methods.

Only 3 directions up, left, right are used until any of those is possible. Down only when other directions are not possible. Weight matrix preferring the higher value fields in one corner are used.

Distance – distance from corner with highest value. This value is used as a penalty in equations the further a cell is located from upper left corner.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{bmatrix}$$

Free space – amount of cells having no **field** values at its location.

Field – cell containing a numerical value on a certain location.

Weight – numeric value of a location displaying its importance. Location weight values are as followed in the provided matrix.

$$\begin{bmatrix} 135 & 121 & 102 & 99 \\ 72 & 76 & 88 & 99 \\ 60 & 56 & 37 & 16 \\ 12 & 9 & 5 & 3 \end{bmatrix}$$

Merge – meaning when two cells having equal numbers on their location, are combined with each other they create a new cell with double the value.

Board – as in game board, containing cells on a 4 by 4 game field.

Monotonicity score – is the sum of all values coming from the monotonic equation for each row and columns which are monotonic, when their cell values are descending from the highest priority corners side

Different scores are based on the following data:

2.2.1 Method 1.

$$\begin{bmatrix} 135 & 121 & 102 & 99 \\ 72 & 76 & 88 & 99 \\ 60 & 56 & 37 & 16 \\ 12 & 9 & 5 & 3 \end{bmatrix}$$

Weight matrix

For each possible merge, **merge score** equation is defined:

$$field[x][y] * weight\ [x][y] * free\ space * \frac{1}{distance[x][y]}$$

Mentioned above equation takes x and y values of a cell with smaller or equal value when both cells are in the same row or column, because two cells can be merged with each other. For the **field** it does not matter which of the two cells to use in the equation because their values are equal.

In this method to the **current_field_score** I am adding value from equation named **board score** as following equation:

$$\sum_{x=1}^{4}\sum_{y=1}^{4} field[x][y] * weight[x][y]$$

For each row if it is non-monotonic, **monotonic score** equation is used:

$$\sum_{x=1}^{4} field[x][y] * weight[x][y]$$

Where Y – Row number.

For each column if it is non-monotonic, **monotonic score** equation is used:

$$\sum_{y=1}^{4} field[x][y] * weight[x][y]$$

Where X – column number.

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = merge score + board score – monotonic score

This method uses both exceptions of the general algorithm.

Example.

The following example describes how **current_field_score** is calculated:

Depth 1. 16 8 4 2
16 0 4 4
0 0 0 0
0 0 0 0
Left = 1684 is more preferred move; Right = 160; up = 552;

Depth 2. Left: 16 8 4 2
16 8 0 0
0 0 0 0
0 0 0 0

merge score = field[1][1] * weight[1][1] * 10 / distance[1][1] + field[2][1] * weight[2][1] * 10 / distance[2][1] = 16 * 7 * 10 / 1 + 8 * 6 * 10 / 2 = 1120 + 240 = 1360

board score = field[1][1] * weight[1][1] + field[2][1] * weight[2][1] + field[3][1] * weight[3][1] + field[4][1] * weight[4][1] + field[1][2] * weight[1][2] + field[2][2] * weight[2][2] = 16 * 7 + 8 * 6 + 4 * 5 + 2 * 4 + 16 * 6 + 8 * 5 = 324

current_field_score = merge score + board score =1360 + 324 = 1684

Depth 2. Right: 16 8 4 2
0 0 16 8
0 0 0 0
0 0 0 0

board score = field[1][1] * weight[1][1] + field[2][1] * weight[2][1] + field[3][1] * weight[3][1] + field[4][1] * weight[4][1] + field[3][2] * weight[3][2] + field[4][2] * weight[4][2] = 16 * 7 + 8 * 6 + 4 * 5 + 2 * 4 + 16 * 4 + 8 * 3 = 276

monotonic score = field[3][1] * weight[3][1] + field[3][2] * weight[3][2] + field[4][1] * weight[4][1] + field[4][2] * weight[4][2] = 4 * 5 + 16 * 4 + 2 * 4 + 8 * 3 = 116

current_field_score = board score – monotonic score = 276 – 116 = 160

Depth 2. Up: 32 8 8 2
0 0 0 4
0 0 0 0
0 0 0 0

Monotonic score = field[4][1] * weight[4][1] + field[4][2] * weight[4][2] = 2 * 4 + 4 * 3 = 20

Merge score = field[2][1] * weight[2][1] * 10 / distance[2][1] = 8 * 6 * 10 / 2 = 240

Board score = board score = field[1][1] * weight[1][1] + field[2][1] * weight[2][1] + field[3][1] * weight[3][1] + field[4][1] * weight[4][1] + field[4][2] * weight[4][2] = 32 * 7 + 8 * 6 + 8 * 5 + 2 * 4 + 4 * 3 = 332

current_field_score = merge score + board score – monotonic score = 240 + 332 – 20 = 552

Another example how **current_field_score** is calculated, is located in appendix 6.1

2.2.2 Method 1.1

This method uses both exceptions of the general algorithm.

Uses same matrix as method 1.For each possible merge, **merge score** equation is defined:

$$field[x][y] * weight\ [x][y] * free\ space * \frac{1}{distance[x][y]}$$

In this method to the **current_field_score** I am adding value from equation named **board score** as following equation:

$$\sum_{x=1}^{4}\sum_{y=1}^{4} field[x][y] * weight[x][y]$$

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = merge score + board score

2.2.3 Method 2.

No exceptions are used. Weight matrix looks like this:

$$\begin{bmatrix} 0.135 & 0.121 & 0.102 & 0.0999 \\ 0.0997 & 0.088 & 0.076 & 0.0724 \\ 0.0606 & 0.0562 & 0.0371 & 0.0161 \\ 0.0125 & 0.0099 & 0.0057 & 0.0033 \end{bmatrix}$$

For each possible merge, **merge score** equation is defined:

Field[x][y] * weight[x][y] * free space

For each row if it is monotonic, **monotonic score** equation is used:

$$\sum_{x=1}^{4} field[x][y] * weight[x][y]$$

Y – Row number.

For each column if it is monotonic, **monotonic score** equation is used:

$$\sum_{y=1}^{4} field[x][y] * weight[x][y]$$

X – Column number.

In this method to the **current_field_score** I am adding value from equation named **board score** as following equation:

$$\sum_{x=1}^{4}\sum_{y=1}^{4} field[x][y] * weight[x][y]$$

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = merge score + monotonic score + board score

Examples how **current_field_score** is calculated are located in appendix 6.2

2.2.4 Method 3.

This method uses exception 2.

$$\begin{bmatrix} 0.135 & 0.121 & 0.102 & 0.0999 \\ 0.0997 & 0.088 & 0.076 & 0.0724 \\ 0.0606 & 0.0562 & 0.0371 & 0.0161 \\ 0.0125 & 0.0099 & 0.0057 & 0.0033 \end{bmatrix}$$

Weight matrix

For each possible merge, **merge score** equation is defined:

Field[x][y] * weight[x][y] * free space

For each row if it is monotonic, **monotonic score** equation is used:

$$\sum_{x=1}^{4} field[x][y] * weight[x][y] * free\ space$$

Y – Row number.

For each column if it is monotonic, **monotonic score** equation is used:

$$\sum_{y=1}^{4} field[x][y] * weight[x][y] * free\ space$$

X – Column number.

In this method to the **current_field_score** I am adding value from equation named **board score** as following equation:

$$\sum_{x=1}^{4}\sum_{y=1}^{4} field[x][y] * weight[x][y]$$

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = merge score + monotonic score + board score

Examples how **current_field_score** is calculated are located in appendix 6.3

2.2.5 Method 3.8

$$\begin{bmatrix} 0.135 & 0.121 & 0.102 & 0.0999 \\ 0.0997 & 0.088 & 0.076 & 0.0724 \\ 0.0606 & 0.0562 & 0.0371 & 0.0161 \\ 0.0125 & 0.0099 & 0.0057 & 0.0033 \end{bmatrix}$$

Weight matrix

In this method to the **current_field_score** I am adding value from equation named **board score** as following equation:

$$\sum_{x=1}^{4}\sum_{y=1}^{4} field[x][y] * weight[x][y]$$

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = board score

2.2.6 Method 4.

This method uses exception 2.

$$\begin{bmatrix} 0.135 & 0.121 & 0.102 & 0.0999 \\ 0.0997 & 0.088 & 0.076 & 0.0724 \\ 0.0606 & 0.0562 & 0.0371 & 0.0161 \\ 0.0125 & 0.0099 & 0.0057 & 0.0033 \end{bmatrix}$$

Weight matrix

For each possible merge, **merge score** equation is defined:

Field[x][y] * weight[x][y]

For each row if it is monotonic, **monotonic score** equation is used:

$$\sum_{x=1}^{4} field[x][y] * weight[x][y]$$

Y – Row number.

For each column if it is monotonic, **monotonic score** equation is used:

$$\sum_{y=1}^{4} field[x][y] * weight[x][y]$$

X – Column number.

In this method to the **current_field_score** I am adding value from equation named **board score** as following equation:

$$\sum_{x=1}^{4}\sum_{y=1}^{4} field[x][y] * weight[x][y]$$

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = merge score + monotonic score + board score

Examples how **current_field_score** is calculated are located in appendix 6.4

2.2.7 Method 5.

This method uses exception 2.Weight matrix is exactly the same as in previous method.

$$\begin{bmatrix} 0.135 & 0.121 & 0.102 & 0.0999 \\ 0.0997 & 0.088 & 0.076 & 0.0724 \\ 0.0606 & 0.0562 & 0.0371 & 0.0161 \\ 0.0125 & 0.0099 & 0.0057 & 0.0033 \end{bmatrix}$$

Weight matrix

For each possible merge, **merge score** equation is defined:

Weight[x][y] * free space.

For each row if it is monotonic, **monotonic score** equation is used:

$$\sum_{x=1}^{4} weight[x][y] * free\ space$$

Y – Row number.

For each column if it is monotonic, **monotonic score** equation is used:

$$\sum_{y=1}^{4} weight[x][y] * free\ space$$

X – Column number.

In this method to the **current_field_score** I am adding value from equation named **board score** as following equation:

$$\sum_{x=1}^{4}\sum_{y=1}^{4} field[x][y] * weight[x][y]$$

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = merge score + monotonic score + board score

Examples how **current_field_score** is calculated are located in appendix 6.5

2.2.8 Method 6.

This method uses exception 2.Weight matrix is exactly the same as in previous method.

$$\begin{bmatrix} 0.135 & 0.121 & 0.102 & 0.0999 \\ 0.0997 & 0.088 & 0.076 & 0.0724 \\ 0.0606 & 0.0562 & 0.0371 & 0.0161 \\ 0.0125 & 0.0099 & 0.0057 & 0.0033 \end{bmatrix}$$

Weight matrix

Here, for each possible merge, **merge score** value simply the biggest weight of merged locations:

Weight[x][y]

For each row if it is monotonic, **monotonic score** equation is used:

$$\sum_{x=1}^{4} weight[x][y]$$

Y – Row number.

For each column if it is monotonic, **monotonic score** equation is used:

$$\sum_{y=1}^{4} weight[x][y]$$

X – Column number.

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = merge score + monotonic score

Examples how **current_field_score** is calculated are located in appendix 6.6

2.2.9 Method 7.

This method uses both exceptions.

$$\begin{bmatrix} 0.135 & 0.121 & 0.102 & 0.0999 \\ 0.0997 & 0.088 & 0.076 & 0.0724 \\ 0.0606 & 0.0562 & 0.0371 & 0.0161 \\ 0.0125 & 0.0099 & 0.0057 & 0.0033 \end{bmatrix}$$

Weight matrix

For each possible merge, **merge score** equation is defined:

Field[x][y] * weight[x][y] * free space

For each row if it is monotonic, **monotonic score** equation is used:

$$\sum_{x=1}^{4} field[x][y] * weight[x][y] * free\ space$$

Y – Row number.

For each column if it is monotonic, **monotonic score** equation is used:

$$\sum_{y=1}^{4} field[x][y] * weight[x][y] * free\ space$$

X – Column number.

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = merge score + monotonic score

Examples how **current_field_score** is calculated are located in appendix 6.7.

2.2.10 Method 7.5

This method does not use exceptions.

$$\begin{bmatrix} 0.135 & 0.121 & 0.102 & 0.0999 \\ 0.0997 & 0.088 & 0.076 & 0.0724 \\ 0.0606 & 0.0562 & 0.0371 & 0.0161 \\ 0.0125 & 0.0099 & 0.0057 & 0.0033 \end{bmatrix}$$

Weight matrix

For each possible merge, **merge score** equation is defined:

Field[x][y] * free space

For each row if it is monotonic, **monotonic score** equation is used:

$$\sum_{x=1}^{4} field[x][y] * weight[x][y] * free\ space$$

Y – Row number.

For each column if it is monotonic, **monotonic score** equation is used:

$$\sum_{y=1}^{4} field[x][y] \ * \ weight[x][y] * free\ space$$

X – Column number.

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = merge score + monotonic score

2.2.11 Method 7.7

This method does not use exceptions.

$$\begin{bmatrix} 0.135 & 0.121 & 0.102 & 0.0999 \\ 0.0997 & 0.088 & 0.076 & 0.0724 \\ 0.0606 & 0.0562 & 0.0371 & 0.0161 \\ 0.0125 & 0.0099 & 0.0057 & 0.0033 \end{bmatrix}$$

Weight matrix

For each possible merge, **merge score** equation is defined:

Field[x][y] * free space

For each row if it is monotonic, **monotonic score** equation is used:

$$\sum_{x=1}^{4} field[x][y] * weight[x][y] * free\ space$$

Y – Row number.

For each column if it is monotonic, **monotonic score** equation is used:

$$\sum_{y=1}^{4} field[x][y] \ * \ weight[x][y] * free\ space$$

X – Column number.

In this method to the **current_field_score** I am adding value from equation named **board score** as following equation:

$$\sum_{x=1}^{4}\sum_{y=1}^{4} field[x][y]$$

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = merge score + monotonic score + board score

2.2.12 Method 8.

This method uses both exceptions.

$$\begin{bmatrix} 0.135 & 0.121 & 0.102 & 0.0999 \\ 0.0997 & 0.088 & 0.076 & 0.0724 \\ 0.0606 & 0.0562 & 0.0371 & 0.0161 \\ 0.0125 & 0.0099 & 0.0057 & 0.0033 \end{bmatrix}$$

Weight matrix

For each possible merge, **merge score** equation is defined:

Field[x][y] * weight[x][y]

For each row if it is monotonic, value of **monotonic score** equation equals to:

$$\sum_{x=1}^{4} field[x][y]$$

Y – Row number.

For each column if it is monotonic, value of **monotonic score** equation equals to:

$$\sum_{y=1}^{4} field[x][y]$$

X – Column number.

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = merge score + monotonic score

Examples how **current_field_score** is calculated are located in appendix 6.8

2.3 All directions methods.

Move total scores are calculated for all four directions. Pseudo code for all directions is the exactly the same as for three. When checking a row or column for monotonicity, it is done in both directions.

2.3.1 Method 9.

Weight matrix compared to previous strategy is completely different, because now all four directions are allowed to make a move into, each corner and side of the field has higher priority than centre locations:

$$\begin{bmatrix} 3 & 2 & 2 & 3 \\ 2 & 1 & 1 & 2 \\ 2 & 1 & 1 & 2 \\ 3 & 2 & 2 & 3 \end{bmatrix}$$

Weight matrix

For each possible merge, **merge score** equation is defined:

$$\text{Field[x][y]} * \text{free space}$$

For each row if it is monotonic, value of **monotonic score** equation equals to:

$$\sum_{x=1}^{4} field[x][y]$$

Y – Row number.

For each column if it is monotonic, value of **monotonic score** equation equals to:

$$\sum_{y=1}^{4} field[x][y]$$

X – Column number.

In this method to the **current_field_score** I am adding value from equation named **board score** as following equation:

$$\sum_{x=1}^{4}\sum_{y=1}^{4} field[x][y] * weight[x][y]$$

The way score is calculated for each field layout at target depth is as shown in equation:

$$\text{current\_field\_score} = \text{merge score} + \text{monotonic score} + \text{board score}$$

Examples how **current_field_score** is calculated are located in appendix 6.9

2.3.2 Method 10.

Weight matrix is exactly the same as in previous method.

$$\begin{bmatrix} 3 & 2 & 2 & 3 \\ 2 & 1 & 1 & 2 \\ 2 & 1 & 1 & 2 \\ 3 & 2 & 2 & 3 \end{bmatrix}$$

Weight matrix

For each possible merge, **merge score** equation is defined:

$$\text{(Field[x1][y1] * weight[x1][y1] + field[x2][y2] * weight[x2][y2]) * free space}$$

For each row if it is monotonic, **monotonic score** equation is used:

$$\left(\sum_{x=1}^{4} field[x][y] * weight[x][y]\right) * free\ space$$

Y – Row number.

For each column if it is monotonic, **monotonic score** equation is used:

$$\left(\sum_{y=1}^{4} field[x][y] * weight[x][y]\right) * free\ space$$

X – Column number.

In this method to the **current_field_score** I am adding value from equation named **board score** as following equation:

$$\sum_{x=1}^{4}\sum_{y=1}^{4} field[x][y] * weight[x][y]$$

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = merge score + monotonic score + board score

Examples how **current_field_score** is calculated are located in appendix 6.10

2.3.3 Method 11.

Weight matrix is exactly the same as in previous method.

$$\begin{bmatrix} 3 & 2 & 2 & 3 \\ 2 & 1 & 1 & 2 \\ 2 & 1 & 1 & 2 \\ 3 & 2 & 2 & 3 \end{bmatrix}$$

Weight matrix

For each possible merge, **merge score** equation is defined:

(Weight[x1][y1] + weight[x2][y2]) * free space

For each row if it is monotonic, **monotonic score** equation is used:

$$\left(\sum_{x=1}^{4} weight[x][y]\right) * free\ space$$

Y – Row number.

For each column if it is monotonic, **monotonic score** equation is used:

$$\left(\sum_{y=1}^{4} weight[x][y] \right) * free\ space$$

X – Column number.

In this method to the **current_field_score** I am adding value from equation named **board score** as following equation:

$$\sum_{x=1}^{4} \sum_{y=1}^{4} field[x][y] * weight[x][y]$$

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = merge score + monotonic score + board score

Examples how **current_field_score** is calculated are located in appendix 6.11

2.3.4 Method 12.

Weight matrix is slightly different from previous method, values were changed so that only corners would be highly valued than all other cells:

$$\begin{bmatrix} 2 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 2 \end{bmatrix}$$

Weight matrix

For each possible merge, **merge score** equation is defined:

(Field[x1][y1] * weight[x1][y1] + field[x2][y2] * weight[x2][y2]) * free space

For each row if it is monotonic, **monotonic score** equation is used:

$$\left(\sum_{x=1}^{4} field[x][y] * weight[x][y] \right) * free\ space$$

Y – Row number.

For each column if it is monotonic, **monotonic score** equation is used:

$$\left(\sum_{y=1}^{4} field[x][y] * weight[x][y] \right) * free\ space$$

X – Column number.

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = merge score + monotonic score

Examples how **current_field_score** is calculated are located in appendix 6.12

2.3.5 Method 13.

$$\begin{bmatrix} 3 & 2 & 2 & 3 \\ 2 & 1 & 1 & 2 \\ 2 & 1 & 1 & 2 \\ 3 & 2 & 2 & 3 \end{bmatrix}$$

Weight matrix

In this method to the **current_field_score** I am adding value from equation named **board score** as following equation:

$$\sum_{x=1}^{4}\sum_{y=1}^{4} field[x][y] * weight[x][y] * free\ space$$

The way score is calculated for each field layout at target depth is as shown in equation:

current_field_score = board score

2.4 Implementation of the 2048 solver.

Swing framework is used to draw the game field, paint and drawTile functions are responsible for drawing.

It is possible to play the game manually by using the arrow keys; to let AI play the game, Enter key must be pressed. The main class file is in game package, which chooses the method to run from fourdirection or threedirection packages. In any methods class file inside function named AI local variable depth_limit is for choosing the depth limit for optimal move depth search. Find_best_move function is responsible for depth search, inside it is declared if board score equation is used. Get_field_situation function sets if merge and monotonic equations are used. SameRowNumbers function is for all merge-able cells equation calculation, row_monotonic and column_monotonic are for monotonicity equation calculation. Source code for developed algorithm is uploaded together with thesis in a zip file.

3 Analysis of the solutions.

Each method have been tested 10 times. Best move search average value of highest cell values achieved in each game, minimum and maximum cell values from highest cell values achieved in all attempts, frequency at which 2048 and 1024 values appear for each method will be written in table 1. Table 2 gives short overview about each methods equations, how many directions are used and if exceptions are in use. Table 3 contains each methods time parameters, as average time per game and turn, minimum and maximum times.

Table 1 Test results

Method #	**Depth limit**	**Average value**	**Min value**	**Max value**	**1024 frequency (%)**	**2048 frequency (%)**
1	3	121	64	256	0	0
1.1	3	524	128	1024	20	0
2	3	486	256	1024	20	0
3	3	742	256	2048	20	10
3.8	3	768	512	2048	20	10
4	3	486	256	1024	10	0
5	3	742	256	1024	50	0
6	3	262	64	512	0	0
7	3	640	256	1024	30	0
7.5	3	755	128	1024	60	0
7.7	3	716	256	2048	30	10
8	3	588	256	1024	30	0
9	3	370	128	512	0	0
10	3	806	256	2048	40	10
11	3	307	128	512	0	0
12	3	652	128	2048	30	10
13	3	640	256	1024	30	0

Table 2 Method equations

Method #	**Logarithm name**	**Merge equation**	**Board score equation**	**Monotonic equation**	**Exceptions included**
Method 1	Three directions	Field * weight * free space / distance	Field * weight	Field * weight	Both exceptions
Method 1.1	Three directions	Field * weight * free space / distance	Field * weight	None	Both exceptions
Method 2	Three directions	Field * weight * free space	Field * weight	Field * weight	None
Method 3	Three directions	Field * weight * free space	Field * weight	Field * weight * free space	Exception 2
Method 3.8	Three directions	Not used	Field * weight * free space	Not used	None
Method 4	Three directions	Field * weight	Field * weight	Field * weight	Exception 2
Method 5	Three directions	Weight * free space	Field * weight	Weight * free space	None
Method 6	Three directions	Weight	Not used	Weight	Exception 2
Method 7	Three directions	Field * weight * free space	Not used	Field * weight * free space	Both exceptions
Method 7.5	Three directions	Field * free space	Not used	Field * weight * free space	None
Method 7.7	Three directions	Field * free space	Field	Field * weight * free space	None
Method 8	Three directions	Field * weight	Not used	Field	Both exceptions
Method 9 matrix 1	Four directions	Field * free space	Field * weight	Field	None
Method 10 matrix 1	Four directions	Field 1 * weight 1 * free space + field 2 * weight 2 *	Field * weight	Field * weight * free space	None

		free space			
Method #	**Logarithm name**	**Merge equation**	**Board score equation**	**Monotonic equation**	**Exceptions included**
Method 11 matrix 1	Four directions	Weight 1 * free space + weight 2 * free space	Field * weight	Weight * free space	None
Method 12 matrix 2	Four directions	Field 1 * weight 1 * free space + field 2 * weight 1 * free space	None	Field * weight * free space	None
Method 13 matrix 1	Four directions	None	Field * weight * free space	None	None

Table 3 Method time table

Method #	**Average time per game(seconds)**	**Average time per turn(seconds)**	**Minimum time for a game(seconds)**	**Maximum time for a game(seconds)**
Method 1	23	0.16	7	37
Method 1.2	54	0.13	30	75
Method 3	98	0.17	62	140
Method 4	71	0.16	55	94
Method 5	110	0.18	53	238
Method 6	40	0.13	20	82
Method 7	92	0.16	63	117
Method 8	84	0.13	46	152
Method 9	195	0.6	126	304
Method 10	290	0.5	155	434
Method 11	163	0.5	114	214
Method 12	224	0.45	125	437
Method 13	224	0.4	138	396

Methods 3, 7.7, 10 and 12 could win the game at 10% win-rate, two methods using four direction movements. Method 10 using four direction score calculation had the highest

average from highest cell numbers created, taking into account cell values and weight, amount of free space in all equations except board score equation. Methods moving in all directions were almost three times slower than those calculating scores in only three directions. Method 1 was unsuccessful because for each non-monotonic row or column it subtracted a penalty from the score, as is proven by method 1.1 since it is similar to method 1 in everything except not using non-monotonic penalty, not enough for winning however. Method 3 using a different weight matrix than method 1, turned out to be successful by a small margin. Method 7 appeared to be similar in average highest value achieved across 10 tries although without winning, it did not take into account cell values in merge and monotonic equations compared to third method. Method 3 had a highest average value increase compared to method 2 when amount of empty space was added to its monotonic equation, also method 5 had an equal average value increase when weight was changed to amount of free space in merge and monotonic equations.

Chapter 1.8 algorithm using a higher depth search is much more effective which achieves 100% chance of creating a 2048 cell, which none of algorithms used are able to recreate. Win rate from chapter 1.7 was almost similar with some methods displayed in solution, both performing at an equal depth level.

4 Summary

The objective of this thesis was to create multiple working algorithms for a game named 2048. These strategies had to fulfil the objective of the game, which is to create 2048 cell on the game field. Strategy principals were explored that were suggested by other people and used as a basis in creating algorithms to beat the game.

Created strategies were compared to each other and to developed algorithms done by individuals separately from my own.

Monotonicity, whole game field layout, weight of each cell, cell value in calculations and best move depth search turned out to be required aspects for a winning algorithm.

The win rate of algorithms is not very high, it is very much possible to make it better considering the results described in chapter 1.There are a lot of calculation possibilities to test which vary from each other by a single difference.

5 References

1. (*Logic – what is the optimal algorithm for the game 2048.* March 2014. stackoverflow.com/questions/22342854/what-is-the-optimal-algorithm-for-the-game-2048.)
2. (Using artificial intelligence to solve the 2048 game. Vasilis Vryniotis. April 2014. blog.datumbox.com/using-artificial-intelligence-to-solve-the-2048-game-java-code.)
3. (Yiyuan Lee. 2048 AI – The intelligent bot. May 2014. codemyroad.wordpress.com/2014/05/14/2048-ai-the-intelligent-bot.)
4. (Nick Statt. 2048 starts easy; gets hard. Here's how to make it easy again. March 2014. cnet.com/news/2048-starts-easy-gets-hard-heres-how-to-make-it-easy-again.)
5. (Strategy and algorithm to win 2048. 2048strategy.com/2048-strategy/.)
6. (Mark Seifter. Mega-R3. Games, Minimax, Alpha-Beta. Youtube. youtube.com/watch?v=hM2EAvMkhtk.).
7. Tic tac toe AI. *Nanyang Technological University*, *Singapore.* n.p. May 2012. Web 26 Dec 2016 <http://www.ntu.edu.sg/home/ehchua/programming/java/JavaGame_TicTacToe_AI.html.>
8. Alpha-beta n.p. n.d. Web 26 Dec 2016 <http://orion.lcg.ufrj.br/Dr.Dobbs/books/book9/mf1210.html.>
9. 2048 AI. Stack Overflow. n.p. 23 Jan 2016 Web 20 Oct 2016 <http://stackoverflow.com/questions/22342854/what-is-the-optimal-algorithm-for-the-game-2048.>
10. (*2048 (video game).*Dec 2016. https://en.wikipedia.org/wiki/2048_(video_game))

6 Appendix

6.1 Example how score is calculated.

Cell values of the game board will be as in this matrix.

$$\begin{bmatrix} 256 & 128 & 32 & 16 \\ 128 & 64 & 8 & 4 \\ 4 & 0 & 0 & 0 \\ 2 & 0 & 0 & 4 \end{bmatrix}$$

Examples are without random element spawning. At depth 3, each leaf node representing cell positions on the board has a score, creating average score on each board state in previous depth levels from which moves in three directions were made.

Move left score = 3950.5; Move right score = 4573; Move up score = 8850.5

Ai would make move up since it is with the most favourable score.

Depth = 1.

1) Move left:

256	128	32	16
128	64	8	4
4	0	0	0
2	4	0	0

Total_score = (Total_score (from 1.2 depth 2) + Total_score (from 1.3 depth 2)) / 2 = (3967 + 3934) / 2 = 3950.5

Depth = 2.

1.1) move left cannot be made.

1.2) move right:

256	128	32	16
128	64	8	4
0	0	0	4
0	0	2	4

Total_score = (current_field_score (from 1.2.1 depth 3) + current_field_score (from 1.2.3 depth 3)) / 2 = (3950 + 3984) / 2 = 3967

Depth = 3.

1.2.1) move left:

256	128	32	16
128	64	8	4
4	0	0	0
2	4	0	0

Board score = 256 * 7 + 128 * 6 + 32 * 5 + 16 * 4 + 128 * 6 + 64 * 5 + 8 * 4 + 4 * 3 + 4 * 5 + 2 * 4 + 4 * 3 = 1792 + 768 + 160 + 64 + 768 + 320 + 32 + 12 + 20 + 8 + 12 = 3956.

Monotonic score = 2 + 4 = 6

current_field_score = board score – monotonic score = 3950.

1.2.2) move right cannot be made.

1.2.3) move up:

256	128	32	16
128	64	8	8
0	0	2	4
0	0	0	0

Merge score = 8 * 4 * 6 / 4 = 48.

Board score = 256 * 7 + 128 * 6 + 32 * 5 + 16 * 4 + 128 * 6 + 64 * 5 + 8 * 4 + 8 * 3 + 2 * 3 + 4 * 2 = 1792 + 768 + 160 + 64 + 768 + 320 + 32 + 24 + 6 + 8 = 3942.

Monotonic score = 6

current_field_score = merge score + board score – monotonic score

= 48 + 3942 – 6 = 3984

Depth = 2. 1.3) move up: 256 128 32 16 128 64 8 4 4 4 0 0 2 0 0 0 Total_score = ((current_field_score (from 1.3.1 depth 3) + (current_field_score (from 1.3.2 depth 3)) / 2 = (3904 + 3964) / 2 = 3934 Depth = 3. 1.3.2) move right: 256 128 32 16 128 64 8 4 0 0 0 8 0 0 0 2 Board score = 256 * 7 + 128 * 6 + 32 * 5 + 16 * 4 + 128 * 6 + 64 * 5 + 8 * 4 + 4 * 3 + 8 * 2 + 2 * 1 = 1792 + 768 + 160 + 64 + 768 + 320 + 32 + 12 + 16 + 2 = 3934. Monotonic score = 16 + 4 + 8 + 2 = 30. **current_field_score =** board score – monotonic score = 3934 – 30 = 3904. 1.3.3) move up cannot be made. Depth = 1. 2) Right: 256 128 32 16 128 64 8 4 0 0 0 4 0 0 2 4 Total_score = ((Total_score (from 2.1 depth 2) + (Total_score (from 2.2 depth 2)) / 2 = (5197 + 3949) / 2 = 4573 Depth = 2. 2.1) move up: 256 128 32 16 128 64 8 8 0 0 2 4 0 0 0 0 Average score = (current_field_score (from 2.1.2 depth 3) + current_field_score (from	Depth = 3. 2.1.1) move left: 256 128 32 16 128 64 16 0 2 4 0 0 0 0 0 0 Board score = 256 * 7 + 128 * 6 + 32 * 5 + 16 * 4 + 128 * 6 + 64 * 5 + 16 * 4 + 2 * 5 + 4 * 4 = 1792 + 768 + 160 + 64 + 768 + 320 + 64 + 10 + 16 = 3962. Monotonic score = 6 **current_field_score** = board score – monotonic score = 3962 – 6 = 3956. Depth = 3. 2.1.2) move right: 256 128 32 16 0 128 64 16 0 0 2 4 0 0 0 0 Merge score = 128 * 6 * 7 / 2 + 16 * 4 * 7 / 4 = 2688 + 112 = 2800 Board score += 256 * 7 + 128 * 6 + 32 * 5 + 16 * 4 + 128 * 5 + 64 * 4 + 16 * 3 + 2 * 3 + 4 * 2 = 1792 + 768 + 160 + 64 + 640 + 256 + 48 + 6 + 8 = 3742 Monotonic score = 32 + 64 + 2 + 2 + 4 = 104 **current_field_score =** merge score + board score – monotonic score = 2800 + 3742 – 104 = 6438 2.1.3) cannot move up. Depth = 2. 2.2) move left: 256 128 32 16 128 64 8 4 4 0 0 0 2 4 0 0 Total_score = (current_field_score (from 2.2.1 depth 3) + current_field_score (from 2.2.2 depth 3)) / 2 = (3938 + 3960) / 2 = 3949

2.2.1 depth 3)) / 2 = (6438 + 3956) / 2 = 5197	

Depth = 3. 2.2.1) move up: 256 128 32 16 128 64 8 4 4 4 0 0 2 0 0 0 Board score += 256 * 7 + 128 * 6 + 32 * 5 + 16 * 4 + 128 * 6 + 64 * 5 + 8 * 4 + 4 * 3 + 4 * 5 + 4 * 4 + 2 * 4 = 1792 + 768 + 160 + 64 + 768 + 320 + 32 + 12 + 20 + 16 + 8 = 3960. **current_field_score =** board score = 3960 2.2.2) move right: 256 128 32 16 128 64 8 4 0 0 0 4 0 0 2 4 Merge score = 4 * 3 * 5 / 5 = 12 Board score = 256 * 7 + 128 * 6 + 32 * 5 + 16 * 4 + 128 * 6 + 64 * 5 + 8 * 4 + 4 * 3 + 4 * 2 + 2 * 2 + 4 * 1 = 1792 + 768 + 160 + 64 + 768 + 320 + 32 + 12 + 8 + 4 + 4 = 3932 Monotonic score = 6 **current_field_score =** merge score + board score – monotonic score = 12 + 3932 – 6 = 3938 2.2.3) move left cannot be made. Depth = 2. 2.3) move right cannot be made. Depth = 1. 3) move up: 256 128 32 16 128 64 8 8 4 0 0 0 2 0 0 0 Total_score = (Total_score (from 3.1 depth 2) + Total_score (from 3.2 depth 2)) / 2 = (6442 + 11259) / 2 = 8850.5	Depth = 2. 3.1) move left: 256 128 32 16 128 64 16 0 4 0 0 0 2 0 0 0 Total_score = current_field_score (from 3.1.3 depth 3) / 1 = 6442 3.1.1) move left cannot be made. 3.1.2) move up cannot be made. 3.1.3) move right: 256 128 32 16 0 128 64 16 0 0 0 4 0 0 0 2 Merge score = 128 * 6 * 7 / 2 + 16 * 4 * 7 / 4 = 2688 + 112 = 2800 Board score = 256 * 7 + 128 * 6 + 32 * 5 + 16 * 4 + 128 * 5 + 64 * 4 + 16 * 3 + 4 * 2 + 2 * 1 = 1792 + 768 + 160 + 64 + 640 + 256 + 48 + 8 + 2 = 3738 Monotonic score = 32 + 64 = 96 **current_field_score =** merge score + board score – monotonic score = 2800 + 3738 – 96 = 6442 Depth = 3. Depth = 2. 3.2) move right: 256 128 32 16 0 128 64 16 0 0 0 4 0 0 0 2 Total_score = (current_field_score (from 3.2.2 depth 3) + current_field_score (from 3.2.3 depth 3)) / 2 = (18554 + 3964) / 2 = 11259

Depth = 3. 3.2.1) move right cannot be made. 3.2.2) move up: 256 256 32 32 0 0 64 4 0 0 0 2 0 0 0 0 Merge score = 256 * 7 * 8 / 1 + 32 * 5 * 8 / 3 = 14336 + 426 = 14762 Board score = 256 * 7 + 256 * 6 + 32 * 5 + 32 * 4 + 64 * 4 + 4 * 3 + 2 * 2 = 1792 + 1536 + 160 + 128 + 256 + 12 + 4 = 3888 Monotonic score = 32 + 64 = 96 **current_field_score =** merge score + board score – monotonic score = 14762 +3888 – 96 = 18554	3.2.3) move left: 256 128 32 16 128 64 16 0 4 0 0 0 2 0 0 0 Board score += 256 * 7 + 128 * 6 + 32 * 5 + 16 * 4 + 128 * 6 + 64 * 5 + 16 * 4 + 4 * 5 + 2 * 4 = 1792 + 768 + 160 + 64 + 768 + 320 + 64 + 20 + 8 = 3964 **current_field_score =** board score = 3964 3.3) move up cannot be made.

6.2. Example how score is calculated.

1.Original state: 512 64 2 2 32 2 2 32 0 0 0 0 0 0 0 0 Left = 240.7; Right =214.8; Up = 240.9 is highest value 1.left: 512 64 4 0 32 4 32 0 0 0 0 0 0 0 0 0 Monotonic score =512 * 0.135 + 64 * 0.121 + 4 * 0.102 + 512 * 0.135 + 32 * 0.0997 + 64 * 0.121 + 4 * 0.088 = 69.12 + 7.744 + 0.408 + 69.12 + 3.19 + 7.744 + 0.35 = 157.676 Board score = 512 * 0.135 + 64 * 0.121 + 4 * 0.102 + 32 * 0.097 + 4 * 0.088 + 32 * 0.076 = 83 **current_field_score =** monotonic score + board score = 157.676 + 83 = 240.7	2.right: 0 512 64 4 0 32 4 32 0 0 0 0 0 0 0 0 monotonic score = 512 * 0.121 + 64 * 0.102 + 4 * 0.0999 + 512 * 0.121 + 32 * 0.088 + 64 * 0.102 + 4 * 0.076 = 61.952 + 6.528 + 0.3996 + 61.952 + 2.816 + 6.528 + 0.304 = 140.48 Board score = 512 * 0.121 + 64 * 0.102 + 4 * 0.099 + 32 * 0.088 + 4 * 0.076 + 32 * 0.0724 = 74.3 **current_field_score =** monotonic score + board score = 140.48 + 74.3 = 214.8

3.up:

512	64	4	2
32	2	0	32
0	0	0	0
0	0	0	0

Monotonic : 512 * 0.135 + 64 * 0.121 + 4 * 0.102 + 2 * 0.0999 + 512 * 0.135 + 32 * 0.0997 + 64 * 0.121 + 2 * 0.088 + 4 * 0.102 = 69.12 + 7.744 + 0.408 + 0.1998 + 69.12 + 3.19 + 7.744 + 0.176 + 0.408 = 158.11

Board score = 512 * 0.135 + 64 * 0.121 + 4 * 0.102 + 2 * 0.0999 + 32 * 0.088 + 2 * 0.088 + 32 * 0.0724 = 82.8

current_field_score = monotonic score + board score = 158.11 + 82.8 = 240.9

2.

256	256	128	128
0	0	128	128
0	0	0	0
0	0	0	0

Left = 323; Up = 1080 is highest value; Right = 550.6

1 left:

512	256	0	0
256	0	0	0
0	0	0	0
0	0	0	0

Monotonic: 512 * 0.135 + 256 * 0.0997 + 512 * 0.135 + 256 * 0.121 = 69.12 + 25.52 + 69.12 + 30.98 = 197.74

Board score = 512 * 0.135 + 256 * 0.121 + 256 * 0.0997 = 125.6

current_field_score = monotonic score + board score = 197.74 + 125.6 = 323

2. up:

256	256	256	256
0	0	0	0
0	0	0	0
0	0	0	0

Merge score = (256 * 0.135 + 256 * 0.102) * 12 = (34.56 + 26.11) * 12 = 728.04

Monotonic score = (256 * 0.135 + 256 * 0.121 + 256 * 0.102 + 256 * 0.0999) * 2 = (34.56 + 30.976 + 26.11 + 25.57) * 2 = 235.4

Board score = 256 * 0.135 + 256 * 0.121 + 256 * 0.102 + 256 * 0.0999 = 117.2

current_field_score = merge score + monotonic score + board score = 728.04 + 235.4 + 117.2 = 1080

3. right:

0	0	512	256
0	0	0	256
0	0	0	0
0	0	0	0

Merge score = 256 * 0.0999 * 13 = 332.47

Monotonic: 512 * 0.102 + 256 * 0.0999 * 2 + 256 * 0.0724 = 52.224 + 51.15 + 18.53 = 121.9

Board score = 512 * 0.102 + 256 * 0.0999 + 256 * 0.0724 = 96.3

current_field_score = merge score + monotonic score + board score = 332.47 + 121.9 + 96.3 = 550.6

6.3 Example how score is calculated.

1.Original state:

512	64	2	2
32	2	2	32
0	0	0	0
0	0	0	0

Left = 1650 is highest value; Right = 1478.3; Up = 1496.1

1.left:

512	64	4	0
32	4	32	0
0	0	0	0
0	0	0	0

Monotonic score = (512 * 0.135 + 64 * 0.121 + 4 * 0.102 + 512 * 0.135 + 32 * 0.0997 + 64 * 0.121 + 4 * 0.088) * 10 = (69.12 + 7.744 + 0.408 + 69.12 + 2.32 + 7.744 + 0.304) * 10 = 1567.6

Board score = 512 * 0.135 + 64 * 0.121 + 4 * 0.102 + 32 * 0.0997 + 4 * 0.088 + 32 * 0.076 = 83.2

current_field_score = monotonic score + board score = 1567.6 + 83.2 = 1650

2.right:

0	512	64	4
0	32	4	32
0	0	0	0
0	0	0	0

Monotonic score = (512 * 0.121 + 64 * 0.102 + 4 * 0.0999 + 512 * 0.121 + 32 * 0.088 + 64 * 0.102 + 4 * 0.076) * 10 = 1404

Board score = 512 * 0.121 + 64 * 0.102 + 4 * 0.0999 + 32 * 0.088 + 4 * 0.076 + 32 * 0.0724 = 74.3

current_field_score = monotonic score + board score = 1404 + 74.3 = 1478.3

3.up:

512	64	4	2
32	2	0	32
0	0	0	0
0	0	0	0

Monotonic score = (512 * 0.135 + 64 * 0.121 + 4 * 0.102 + 2 * 0.0999 + 512 * 0.135 + 32 * 0.0997 + 64 * 0.121 + 2 * 0.088 + 4 * 0.102) * 9 = (69,12 + 7,744 + 0.408 + 0.2 + 69.12 + 2.3 + 7.744 + 0.15 + 0.408) * 9 = 1413

Board score = 512 * 0.135 + 64 * 0.121 + 4 * 0.102 + 2 * 0.0999 + 32 * 0.0997 + 2 * 0.088 + 32 * 0.0724 = 83.1

current_field_score = monotonic score + board score = 1413 + 83.1 = 1496.1

2.

256	256	128	128
0	0	128	128
0	0	0	0
0	0	0	0

Left = 3140.4; Up = 3658 is highest value; Right = 2959

1 left:

512	256	0	0
256	0	0	0
0	0	0	0
0	0	0	0

Monotonic score: (512 * 0.135 * 2 + 256 * 0.121 * 2 + 256 * 0.0997 * 2) * 12 = 3014.8

Board score = 512 * 0.135 + 256 * 0.121 + 256 * 0.0997 = 125.6

current_field_score = monotonic score + board score = 3014.8 + 125.6 = 3140.4

2. up: 256 256 256 256 0 0 0 0 0 0 0 0 0 0 0 0 Merge score: (256 * 0.135 +256 * 0.102) * 12 = (34.56 + 26.1) * 12 = 728 Monotonic score: (256 * 0.135 * 2 + 256 * 0.121 * 2 + 256 * 0.102 * 2 + 256 * 0.0999 * 2) * 12 = (69.12 + 62+ 52.2 + 51.15) * 12 = 2813.6 Board score = 256 * 0.135 + 256 * 0.121 + 256 * 0.102 + 256 * 0.0999 = 117.2 **current_field_score =** merge score + monotonic score + board score = 728 + 2813.6 + 117.2 = 3658	3. right: 0 0 512 256 0 0 0 256 0 0 0 0 0 0 0 0 Merge score =256 * 0.0999 * 13 = 332.5 Monotonic score = (512 * 0.102 * 2 + 256 * 0.0999 * 2 + 256 * 0.0724* 2) * 13 = 2504.6 Board score = 512 * 0.102 + 512 * 0.0999 + 256 * 0.0724 = 121.9 **current_field_score =** merge score + monotonic score + board score = 332.5 + 2504.6 + 121.9 = 2959

6.4 Example how score is calculated.

1.Original state: 512 64 2 2 32 2 2 32 0 0 0 0 0 0 0 0 Left = 237.5; Right = 221.9; Up = 241.2 is highest value. 1.left: 512 64 4 0 32 4 32 0 0 0 0 0 0 0 0 0 Monotonic score: 512 * 0.135 + 64 * 0.121 + 4 * 0.102 + 512* 0.135 + 32 * 0.0997 + 64 * 0.121 + 4 * 0.088 = 69.12 + 6.53 + 0.41 + 69.12 + 2.32 + 6.53 + 0.304 = 154.3 Board score = 512 * 0.135 + 64 * 0.121 + 4 * 0.102 +32 * 0.0997 + 4 * 0.088 + 32 * 0.076 = 83.2 **current_field_score =** monotonic score + board score = 154.3 + 83.2 = 237.5	2.right: 0 512 64 4 0 32 4 32 0 0 0 0 0 0 0 0 Monotonic score: 512 * 0.121 + 64 * 0.102 + 4 * 0.0999 + 512 * 0.121 + 32 * 0.088 + 64 * 0.102 + 4 * 0.076 = 140.5 Board score = 512 * 0.135 + 64 * 0.102 + 4 * 0.0999 + 32 * 0.088 + 4 * 0.076 + 32 * 0.0724 = 81.4 **current_field_score =** monotonic score + board score = 140.5 + 81.4 = 221.9 3.Up: 512 64 4 2 32 2 0 32 0 0 0 0 0 0 0 0 Monotonic score: 512 * 0.135 + 64 * 0.121 + 4 * 0.102 + 2 * 0.0999 + 512 * 0.135 + 32 * 0.0997 + 64 * 0.121 + 2 * 0.088 + 4 * 0.102 = 158.1

Board score = 512 * 0.135 + 64 * 0.121 + 4 * 0.102 + 2 * 0.0999 + 32 * 0.0997 + 2 * 0.088 + 32 * 0.0724 = 83.1

current_field_score = monotonic score + board score = 158.1 + 83.1 = 241.2

2.

256	256	128	128
0	0	128	128
0	0	0	0
0	0	0	0

Left = 376.83; Up = 412.3 is highest value; Right = 340

1 left:

512	256	0	0
256	0	0	0
0	0	0	0
0	0	0	0

Monotonic score: 512 * 0.135 * 2 + 256 * 0.121 * 2 + 256 * 0.0997 * 2 = 251.23

Board score = 512 * 0.135 + 256 * 0.121 + 256 * 0.0997 = 125.6

current_field_score = monotonic score + board score = 251.23 + 125.6 = 376.83

2. Up:

256	256	256	256
0	0	0	0
0	0	0	0
0	0	0	0

Merge score: 256 * 0.135 + 256 * 0.102 = 60.67

Monotonic score: (256 * 0.135 + 256 * 0.121 + 256 * 0.102 + 256 * 0.0999) * 2 = 234.45

Board score = 256 * 0.135 + 256 * 0.121 + 256 * 0.102 + 256 * 0.0999 = 117.2

current_field_score = merge score + monotonic score + board score = 60.67 + 234.45 + 117.2 = 412.3

3. Right:

0	0	512	256
0	0	0	256
0	0	0	0
0	0	0	0

Merge score: 256 * 0.0999 = 25.57

Monotonic score: 512 * 0.102 * 2 + 256 * 0.0999 * 2 + 256 * 0.0724 * 2 = 192.6

Board score = 512 * 0.102 + 512 * 0.0999 + 256 * 0.0724 = 121.9

current_field_score = merge score + monotonic score + board score = 25.57 + 192.6 + 121.9 = 340

6.5 Example how score is calculated.

1.Original state:

512	64	2	2
32	2	2	32
0	0	0	0
0	0	0	0

Left = 91.2; Right = 81.4; Up = 92.1 is highest value

1.left:

512	64	4	0
32	4	32	0
0	0	0	0
0	0	0	0

Monotonic score: (0.135 + 0.121 + 0.102 + 0.135 + 0.0997 + 0.121 + 0.088) * 10 = 8

Board score = 512 * 0.135 + 64 * 0.121 + 4 * 0.102 + 32 * 0.0997 + 4 * 0.088 + 32 * 0.076 = 83.2

current_field_score = monotonic score + board score = 8 + 83.2 = 91.2

2.Right:

0	512	64	4
0	32	4	32
0	0	0	0
0	0	0	0

Monotonic score: (0.121 + 0.102 + 0.0999 + 0.121 + 0.088 + 0.102 + 0.076) * 10 = 7.1

Board score = 512 * 0.121 + 64 * 0.102 + 4 * 0.0999 + 32 * 0.088 + 4 * 0.076 + 32 * 0.0724 = 74.3

current_field_score = monotonic score + board score = 74.3 + 7.1 = 81.4

3.Up:

512	64	4	2
32	2	0	32
0	0	0	0
0	0	0	0

Monotonic score: (0.135 + 0.121 + 0.102 + 0.099 + 0.135 + 0.0997 + 0.121 + 0.088 + 0.102) * 9 = 9.02

Board score = 512 * 0.135 + 64 * 0.121 + 4 * 0.102 + 2 * 0.0999 + 32 * 0.0997 + 2 * 0.088 + 32 * 0.0724 = 83.1

current_field_score = monotonic score = 9 + 83.1 = 92.1

2.

256	256	128	128
0	0	128	128
0	0	0	0
0	0	0	0

Left = 134.8 is highest value; Up = 131; Right = 104.7

1 left:

512	256	0	0
256	0	0	0
0	0	0	0
0	0	0	0

Monotonic score: (0.135 * 2 + 0.121 * 2 + 0.0997 * 2) * 13 = 9.2

Board score = 512 * 0.135 + 256 * 0.121 + 256 * 0.0997 = 125.6

current_field_score = monotonic score + board score = 9.2 + 125.6 = 134.8

2. up:

256	256	256	256
0	0	0	0
0	0	0	0
0	0	0	0

Merge score: (0.135 + 0.102) * 12 = 2.8

Monotonic score: (0.135 + 0.121 + 0.102 + 0.0999) * 2 * 12 = 0.46 * 2 * 12 = 11

Board score = 256 * 0.135 + 256 * 0.121 + 256 * 0.102 + 256 * 0.0999 = 117.2

current_field_score = merge score + monotonic score + board score = 2.8 + 11 + 117.2 = 131

3. right:

0	0	512	256
0	0	0	256
0	0	0	0
0	0	0	0

Merge score: 0.0999 * 13 = 1.3

Monotonic score: (0.102 + 0.0999 + 0.0724) * 2 * 13 = 7.1

Board score = 512 * 0.102 + 256 * 0.0999 + 256 * 0.0724 = 96.3

current_field_score = merge score + monotonic score + board score = 1.3 + 7.1 + 96.3 = 104.7

6.6 Example how score is calculated.

1.Original state:

512	64	2	2
32	2	2	32
0	0	0	0
0	0	0	0

Left = 0.8; Right = 0.71; Up = 1 is highest

1.left:

512	64	4	0
32	4	32	0
0	0	0	0
0	0	0	0

Monotonic score: 0.135 + 0.121 + 0.102 + 0.135 + 0.0997 + 0.121 + 0.088 = 0,8

current_field_score = monotonic score = 0,8

2.right:

0	512	64	4
0	32	4	32
0	0	0	0
0	0	0	0

Monotonic score: 0.121 + 0.102 + 0.0999 + 0.121 + 0.088 + 0.102+ 0.076 = 0.71

current_field_score = monotonic score = 0.71

3.up:

512	64	4	2
32	2	0	32
0	0	0	0
0	0	0	0

Monotonic score: 0.135 + 0.121 + 0.102+ 0.0999 +0.135 + 0.0997 + 0.121 + 0.088 + 0.102 = 1

current_field_score = monotonic score = 1

2.

256	256	128	128
0	0	128	128
0	0	0	0
0	0	0	0

Left = 0.7; Up = 1.16 is highest value; Right = 0.5

1 left:

512	256	0	0
256	0	0	0
0	0	0	0
0	0	0	0

Monotonic score: (0.135 + 0.121 + 0.0997) * 2 = 0,7

current_field_score = monotonic score = 0,7

2. up:

256	256	256	256
0	0	0	0
0	0	0	0
0	0	0	0

Merge score: 0.135 + 0.102 = 0.24

Monotonic score: (0.135 + 0.121 + 0.102 + 0.0999) * 2 = 0.92

current_field_score = merge score + monotonic score = 0.24 + 0.92 = 1.16

3. right:

0	0	512	256
0	0	0	256
0	0	0	0
0	0	0	0

Merge score: 0.0999

Monotonic score: (0.102 + 0.0999 + 0.0724) * 2 = 0,5

current_field_score = merge score + monotonic score = 0,5

6.7 Example how score is calculated.

1.Original state:

512	64	2	2
32	2	2	32
0	0	0	0
0	0	0	0

Left = 1577 is highest value; Right = 1406; Up = 1423.

1.left:

512	64	4	0
32	4	32	0
0	0	0	0
0	0	0	0

Monotonic score: (512 * 0.135 + 64 * 0.121 + 4 * 0.102 + 512 * 0.135 + 32 * 0.0997 + 64 * 0.121 + 4 * 0.088) * 10 = (69.12 + 7.74 + 0.41 + 69.12 + 3.2 + 7.74 + 0.35) * 10 = 157.7 * 10 = 1577

current_field_score = monotonic score = 1577

2.right:

0	512	64	4
0	32	4	32
0	0	0	0
0	0	0	0

Monotonic score: (512 * 0.121 + 64 *0.102 + 4 * 0.0999 + 512 * 0.121 + 32 * 0.088 + 64 * 0.102 + 4 * 0.076) * 10 = (62 + 6.53 + 0.4 + 62 + 2.82 + 6.53 + 0.3) * 10 = 140.58 * 10 = 1406

current_field_score = monotonic score = 1406

3.up:

512	64	4	2
32	2	0	32
0	0	0	0
0	0	0	0

Monotonic score: (512 * 0.135 + 64 * 0.121 + 4 * 0.102 + 2 * 0.0999 + 512 * 0.135 + 32 * 0.0997 + 64 * 0.121 + 2 * 0.088 + 4 * 0.102) * 9 = 158.1 * 9 = 1423

current_field_score = monotonic score = 1423

2.

256	256	128	128
0	0	128	128
0	0	0	0
0	0	0	0

Left = 3266.1; Up = 3514.2 is highest value; Right = 2836.3

1 left:

512	256	0	0
256	0	0	0
0	0	0	0
0	0	0	0

Monotonic score: (512 * 0.135 + 256 * 0.121 + 256 * 0.0997) * 2 * 13 = 125.62 * 2 * 13 = 3266.1

current_field_score = monotonic score = 3266.1

2. up:

256	256	256	256
0	0	0	0
0	0	0	0
0	0	0	0

Merge score: (256 * 0.135 + 256 * 0.102) * 12 = 60.7 * 12 = 728.4

Monotonic score: (256 * 0.135 + 256 * 0.121 + 256 * 0.102 + 256 * 0.0999) * 2 * 12 = 117.2 * 2 * 12 = 2812.8

current_field_score = merge score + monotonic score = 728.4 + 2812.8 = 3514.2

3. right:

0	0	512	256
0	0	0	256
0	0	0	0
0	0	0	0

Merge score: (256 * 0.0999) * 13 = 332.5

Monotonic score: (512 * 0.102 + 256 * 0.0999 + 256 * 0.0724) * 2 * 13 = 96.3 * 2 * 13 = 2503.8

Total score = 332.5 + 2503.8 = 2836.3

current_field_score = merge score + monotonic score = 332.5 + 2503.8 = 2836.3

6.8 Example how score is calculated.

1.Original state:

512	64	2	2
32	2	2	32
0	0	0	0
0	0	0	0

Left = 1192; Right = 1192; Up = 1196 is highest value

1.left:

512	64	4	0
32	4	32	0
0	0	0	0
0	0	0	0

Monotonic score: 512 + 64 + 4 + 512 + 32 + 64 + 4 = 1192

current_field_score = monotonic score = 1192

2.right:

0	512	64	4
0	32	4	32
0	0	0	0
0	0	0	0

Monotonic score: 512 + 64 + 4 + 512 + 32 + 64 + 4 = 1192

current_field_score = monotonic score = 1192

3.up:

512	64	4	2
32	2	0	32
0	0	0	0
0	0	0	0

Monotonic score: 512 + 64 + 4 + 2 + 512 + 32 + 64 + 2 + 4 = 1196

current_field_score = monotonic score = 1196

2.

256	256	128	128
0	0	128	128
0	0	0	0
0	0	0	0

Left = 2048; Up = 2108.67 is highest score; Right = 2073.57

1 left:

512	256	0	0
256	0	0	0
0	0	0	0
0	0	0	0

Monotonic score: (512 + 256 + 256) * 2 = 2048

current_field_score = monotonic score = 2048

2. up:

256	256	256	256
0	0	0	0
0	0	0	0
0	0	0	0

Merge score: (256 * 0.135 + 256 * 0.102) = 60.67

Monotonic score: 256 * 8 = 2048

current_field_score = merge score + monotonic score = 2048 + 60.67 = 2108.67

3. right:

0	0	512	256
0	0	0	256
0	0	0	0
0	0	0	0

Merge score: 256 * 0.0999 = 25.57

Monotonic score: (512 + 256 + 256) * 2 = 2048

current_field_score = merge score + monotonic score = 2048 + 25.57 = 2073.57

6.9 Example how score is calculated.

1.

32	8	0	0
256	64	8	0
256	256	16	0
0	2	0	0

Left = 2658; Up = 3264 is highest value;
Down = 2692;

Right = 2934.

Left:

32	8	0	0
256	64	8	0
512	16	0	0
2	0	0	0

Monotonic score: 32 + 8 + 256 + 64 + 8 + 512 + 16 + 2 + 2 + 8 + 8 = 916

Board score = 32 * 3 + 8 * 2 + 256 * 2 + 64 + 8 + 512 * 2 + 16 + 2 * 3 = 1742

current_field_score = monotonic score + board score = 916 + 1742 = 2658

Right:

0	0	32	8
0	256	64	8
0	0	512	16
0	0	0	2

Merge score: 16 * 8 = 128

Monotonic score: 32 + 8 + 256 + 4 + 8 + 512 + 16 + 2 + 2 + 256 + 256 + 512 + 64 + 32 = 1960

Board score = 32 * 2 + 8 * 3 + 256 + 64 + 8 * 2 + 512 + 16 * 2 + 2 * 3 = 974

current_field_score = merge score + monotonic score + board score = 1960 + 974 = 2934

Up:

32	8	8	0
512	64	16	0
0	256	0	0
0	2	0	0

Merge score: 8 * 8 = 64

Monotonic score: 512 + 32 + 16 + 8 + 32 + 8 + 8 + 512 + 64 + 16 + 256 * 2 + 4 = 1724

Board score = 32 * 3 + 8 + 8 + 512 * 2 + 64 + 16 + 256 + 2 * 2 = 1476

current_field_score = merge score + monotonic score + board score = 64 + 1724 + 1476 = 3264

Down:

0	8	0	0
0	64	0	0
32	256	8	0
512	2	16	0

Monotonic score: 512 + 32 + 16 + 8 + 64 + 64 + 8 + 8 = 712

Board score = 8 * 2 + 64 + 32 * 2 + 256 + 8 + 512 * 3 + 2 * 2 + 16 * 2 = 1980

current_field_score = monotonic score + board score = 712 + 1980 = 2692

2.

32	16	0	16
0	4	4	16
0	0	8	0
0	32	32	0

Left = 1520; Right = 1384; Up = 1088; Down = 1836 is highest value.

Left:

32	32	0	0
8	16	0	0
8	0	0	0
64	0	0	0

Merge score: 16 * 10 + 64 * 10 = 160 + 640 = 800

Monotonic score: 32 + 16 + 64 + 64 + 16 + 64+ 64 = 320

Board score = 32 * 3 + 32 * 2 + 8 * 2 + 16 + 8*2 + 64 * 3 = 400

current_field_score = merge score + monotonic score + board score = 800 + 320 + 400 = 1520

Right:

0	0	32	32
0	0	8	16
0	0	0	8
0	0	0	64

Merge score: 64 * 10 = 640

Monotonic score: 32 * 4 + 16 + 8 + 8 + 8 + 64 + 64 + 32 + 8 = 336

Board score = 32 * 2 + 32 * 3 + 8 + 16 * 2 + 8 * 2 + 64 * 3 = 408

current_field_score = merge score + monotonic score + board score = 640 + 336 + 408 = 1384

Up:

32	16	4	32
0	4	8	0
0	32	32	0
0	0	0	0

Merge score: 64 * 8 = 512

Monotonic score: 32 * 4 + 8 + 4 + 32 * 4 = 268

Board score = 32 * 3 + 16 * 2 + 4 * 2 + 32 * 3 + 4 + 8 + 32 + 32 = 308

current_field_score = merge score + monotonic score + board score = 512 + 268 + 308 = 1088

Down:

0	0	0	0
0	16	4	0
0	4	8	0
32	32	32	32

Merge score: 128 * 8 = 1024

Board score = 16 + 4 + 4 + 8 + 32 * 3 + 32 * 2 + 32 * 2 + 32 * 3 = 352

Monotonic score: 32 * 8 + 8 + 4 + 16 + 4 + 32 + 8 + 4 + 32 * 4 = 460

current_field_score = merge score + monotonic score + board score = 1024 + 352 + 460 = 1836

6.10 Example how score is calculated.

1.

32	8	0	0
256	64	8	0
256	256	16	0
0	2	0	0

Left = 15854; Right = 18030; Up = 25188 is highest value; Down = 16380

Left:

32	8	0	0
256	64	8	0
512	16	0	0
2	0	0	0

Monotonic score: (16 * 1 + 512 * 2 + 16 * 1 + 256 * 2 + 64 + 8 + 32 * 3 + 8 * 2 + 4 * 3) * 8 = 14112

Board score = 32 * 3 + 8 * 2 + 256 * 2 + 64 + 8 + 512 * 2 + 16 + 2 * 3 = 1742

current_field_score = monotonic score + board score = 14112 + 1742 = 15854

Right:

0	0	32	8
0	256	64	8
0	0	512	16
0	0	0	2

Merge score: (8 * 3 + 8 * 2) * 8 = 320

Monotonic score: (512 + 512 + 64 + 32 * 2 + 512 + 16 * 2 + 256 + 64 + 8 * 2 + 32 + 8 * 2 + 4 * 3) * 8 = 16736

Board score = 32 * 2 + 8 * 3 + 256 + 64 + 8 * 2 + 512 + 16 * 2 + 2 * 3 = 974

current_field_score = merge score + monotonic score + board score = 16736 + 320 + 974 = 18030

Up:

32	8	8	0
512	64	16	0
0	256	0	0
0	2	0	0

Merge score = 16 * 2 * 2 * 8 = 512

Monotonic score = (512 * 2 + 32 * 3 + 16 + 8 * 2 + 32 * 3 + 8 * 2 + 8 * 2 + 512 * 2 + 64 + 16 + 256 + 256 + 4) * 8 = 23200

Board score = 32 * 3 + 8 + 8 + 512 * 2 + 64 + 16 + 256 + 2 * 2 = 1476

current_field_score = merge score + monotonic score + board score = 512 + 23200 + 1476 = 25188

Down:

0	8	0	0
0	64	0	0
32	256	8	0
512	2	16	0

Monotonic score: (512 * 3 + 32 * 2 + 16 * 2 + 8 + 64 + 64 + 16 * 2) * 8 = 14400

Board score = 8 * 2 + 64 + 32 * 2 + 256 + 8 + 512 * 3 + 2 * 2 + 16 * 2 = 1980

current_field_score = monotonic score + board score = 14400 + 1980 = 16380

2.

32	16	0	16
0	4	4	16
0	0	8	0
0	32	32	0

Left = 11120; Right = 10488; Up = 5396; Down = 11968is highest value

Left:

32	32	0	0
8	16	0	0
8	0	0	0
64	0	0	0

Merge score: (16 * 2 * 2 +32 * 3 + 32 * 2) * 10 = 2240

Monotonic score: (64 * 3 + 64 * 2 + 16 + 8 * 2 + 16 * 2 + 64 * 3 + 64 * 3 + 32 * 2 + 16) * 10 = 8480

Board score = 32 * 3 + 32 * 2 + 8 * 2 + 16 + 8*2 + 64 * 3 = 400

current_field_score = merge score + monotonic score + board score = 2240 + 8480 + 400 = 11120

Right:

0	0	32	32
0	0	8	16
0	0	0	8
0	0	0	64

Merge score: (32 * 3 + 32 * 2) * 10 = 1600

Monotonic score: (64 * 3 + 64 * 2 + 32 * 2 + 8 + 64 * 3 + 64 * 3 +16 * 2 + 16 * 2 + 8) * 10 = 8480

Board score = 32 * 2 + 32 * 3 + 8 + 16 * 2 + 8 * 2 + 64 * 3 = 408

current_field_score = merge score + monotonic score + board score = 1600 + 8480 + 408 = 10488

Up: 32 16 4 32 0 4 8 0 0 32 32 0 0 0 0 0 Merge score: (32 + 32) * 8 = 512 Monotonic score: (64 * 3 + 32 + 8 + 4 * 2 + 64 * 3 + 32 * 4 + 8 + 4) * 8 = 4576 Board score = 32 * 3 + 16 * 2 + 4 * 2 + 32 * 3 + 4 + 8 + 32 + 32 = 308 **current_field_score =** merge score + monotonic score + board score = 512 + 4576 + 308 = 5396	Down: 0 0 0 0 0 16 4 0 0 4 8 0 32 32 32 32 Merge score: (32 * 3 + 32 * 2 + 32 * 2 + 32 * 3) * 8 = 2560 Monotonic score: (32 * 3 + 32 * 2 + 32 * 2 + 32 * 3 + 32 * 3 + 32 * 2 + 32 * 2 + 32 * 3 + 32 * 2 + 8 + 4 + 8 + 4 + 16 + 4 + 32 * 4 * 3) * 8 = 9056 Board score = 16 + 4 + 4 + 8 + 32 * 3 + 32 * 2 + 32 * 2 + 32 * 3 = 352 **current_field_score =** merge score + monotonic score + board score = 2560 + 9056 + 352 = 11968

6.11 Example how score is calculated.

1. 32 8 0 0 256 64 8 0 256 256 16 0 0 2 0 0 Left = 1902; Up = 1708; Down = 2092 is highest value; Right = 1214. Left: 32 8 0 0 256 64 8 0 512 16 0 0 2 0 0 0 Monotonic score: (3 + 3 + 2 + 1 + 2 + 1 + 1 + 3 + 2 + 1 + 1) * 8 = 160 Board score = 32 * 3 + 8 * 2 + 256 * 2 + 64 + 8 + 512 * 2 + 16 + 2 * 3 = 1742 **current_field_score =** monotonic score + board score = 160 + 1742 = 1902	Right: 0 0 32 8 0 256 64 8 0 0 512 16 0 0 0 2 Merge score: (3 + 2) * 8 = 48 Monotonic score: (1 + 1 + 2 + 1 + 2 + 3 + 3 + 2 + 3 + 1 + 1 + 1 + 1 + 2) * 8 = 192 Board score = 32 * 2 + 8 * 3 + 256 + 64 + 8 * 2 + 512 + 16 * 2 + 2 * 3 = 974 **current_field_score =** merge score + monotonic score = 48 + 192 + 974 = 1214

Up:

32	8	8	0
512	64	16	0
0	256	0	0
0	2	0	0

Merge score: (2 + 2) * 8 = 32

Monotonic score: (2 + 3 + 1 + 2 + 2 + 2 + 1 + 1 + 2 + 1 + 1 + 3 + 2 + 2) * 8 = 200

Board score = 32 * 3 + 8 + 8 + 512 * 2 + 64 + 16 + 256 + 2 * 2 = 1476

current_field_score = merge score + monotonic score + board score = 32 + 200 + 1476 = 1708

Down:

0	8	0	0
0	64	0	0
32	256	8	0
512	2	16	0

Monotonic score: (3 + 2 + 2 + 1 + 1 + 1 + 2 + 2) * 8 = 112

Board score = 8 * 2 + 64 + 32 * 2 + 256 + 8 + 512 * 3 + 2 * 2 + 16 * 2 = 1980

current_field_score = monotonic score + board score = 112 + 1980 = 2092

2.

32	16	0	16
0	4	4	16
0	0	8	0
0	32	32	0

Left = 740 is highest value; Right = 698; Up = 468; Down = 624.

Left:

32	32	0	0
8	16	0	0
8	0	0	0
64	0	0	0

Merge score: (2 + 2 + 3 + 2) * 10 = 110

Monotonic score: (3 + 3 + 2 + 2 + 1 + 2 + 3 + 2 + 3 + 2) * 10 = 230

Board score = 32 * 3 + 32 * 2 + 8 * 2 + 16 + 8*2 + 64 * 3 = 400

current_field_score = merge score + monotonic score + board score = 740

Right:

0	0	32	32
0	0	8	16
0	0	0	8
0	0	0	64

Merge score: (3 + 2) * 10 = 60

Monotonic score: (3 + 3 + 2 + 2 + 2 + 1 + 3 + 2 + 2 + 3) * 10 = 230

Board score = 32 * 2 + 32 * 3 + 8 + 16 * 2 + 8 * 2 + 64 * 3 = 408

current_field_score = merge score + monotonic score + board score = 60 + 230 + 408 = 698

Up:

32	16	4	32
0	4	8	0
0	32	32	0
0	0	0	0

Merge score: (1 + 1) * 8 = 16

Monotonic score: (1*4 + 1 + 1 + 3 * 4) * 8 = 144

Board score = 32 * 3 + 16 * 2 + 4 * 2 + 32 * 3 + 4 + 8 + 32 + 32 = 308

current_field_score = merge score + monotonic score + board score = 16 + 144 + 308 = 468

Down:

0	0	0	0
0	16	4	0
0	4	8	0
32	32	32	32

Merge score: (3 + 2 + 2 + 3) * 8 = 80

Monotonic score: (3*2 + 2*2 + 2 *2 + 3*2 + 1 + 1 + 1 + 1) * 8 = 192

Board score = 16 + 4 + 4 + 8 + 32 * 3 + 32 * 2 + 32 * 2 + 32 * 3 = 352

current_field_score = merge score + monotonic score + board score = 80 + 192 + 352 = 624

6.12 Example how score is calculated

1.

32	8	0	0
256	64	8	0
256	256	16	0
0	2	0	0

Left = 8548; Right = 17356 is highest value. Not efficient move. Up = 15362; Down = 10242

Left:

32	8	0	0
256	64	8	0
512	16	0	0
2	0	0	0

Monotonic score: (2 * 2 * 2 + 512 + 16 + 256 + 64 + 8 + 32 * 2 + 8 + 8 + 8) * 8 = 7616

Board score = 32 * 2 + 8 + 256 + 64 + 8 + 512 + 16 + 2*2 = 932

current_field_score = monotonic score + board score = 7616 + 932 = 8548

Right:

0	0	32	8
0	256	64	8
0	0	512	16
0	0	0	2

Merge score: (8 * 2 + 8) * 8 = 192

Monotonic score: (2 * 2 * 2 + 512 + 16 + 256 + 64 + 8 + 32 + 8 * 2 + 256 * 2 + 512 + 64 + 32) * 8 = 16256

Board score = 256 + 32 + 8*2 + 64 + 8 + 512 + 16 + 2 *2 = 908

current_field_score = merge score + monotonic score + board score = 192 + 16256 + 908 = 17356

Up:

32	8	8	0
512	64	16	0
0	256	0	0
0	2	0	0

Merge score: (8 + 8) * 8 = 128

Monotonic score: (32 * 2 + 8 + 8 + 512 + 64 + 16 + 256 * 2 + 2 * 2 + 16 + 8 + 512 + 32 * 2) * 8 = 14304

Board score = 32 * 2 + 8 + 8 +512 + 64 + 16 + 256 + 2 = 930

current_field_score = merge score + board score + monotonic score = 128 + 14304 + 930 = 15362

Down:

0	8	0	0
0	64	0	0
32	256	8	0
512	2	16	0

Monotonic score: (512 * 2 + 32 + 16 + 8 + 4 * 2 + 8 * 2) * 8 = 8832

Board score = 512 * 2 + 2 + 16 + 32 + 256 + 8 + 64 + 8 = 1410

current_field_score = monotonic score + board score = 8832 + 1410 = 10242

2.

32	16	0	16
0	4	4	16
0	0	8	0
0	32	32	0

Left = 6736 is highest value; Right = 6096; Up = 3840; Down = 5539

Left:

32	32	0	0
8	16	0	0
8	0	0	0
64	0	0	0

Merge score: (32 * 2 + 32 + 8 +8) * 10 = 1120

Monotonic score: (64 * 2 * 2 + 32 + 16 + 32 * 2 * 2 + 32 * 2 + 8 * 2 + 16 + 8) * 10 = 5360

Board score = 32 * 2 + 32 + 8 + 16 + 8 + 64 * 2 = 256

current_field_score = merge score + monotonic score + board score = 1120 + 5360 + 256 = 6736

Right: 0 0 32 32 0 0 8 16 0 0 0 8 0 0 0 64 Merge score: (32 + 32 * 2) * 10 = 960 Monotonic score: (64 * 2 * 2 + 8 + 8 + 16 + 8 + 32 * 2 * 2 + 32 * 2) * 10 = 4880 Board score = 32 + 32 * 2 + 8 + 16 + 8 + 64 * 2 = 256 **current_field_score =** merge score + monotonic score + board score = 960 + 4880 + 256 = 6096 Up: 32 16 4 32 0 4 8 0 0 32 32 0 0 0 0 0 Merge score: (32 + 32) * 8 = 96 Monotonic score: (32 * 4 + 8 + 4 + 32 * 4 * 2 + 32 + 8 + 4) * 8 = 3520 Board score = 32 * 2 + 16 + 4 + 32 * 2 + 4 + 8 + 32 + 32 = 224 **current_field_score =** merge score + monotonic score + board score = 96 + 3520 + 224 = 3840	Down: 0 0 0 0 0 16 4 0 0 4 8 0 32 32 32 32 Merge score: (32 * 2 * 2 + 32 * 2) * 8 = 1536 Monotonic score: (32 * 2 * 4 + 32 * 4 + 32 + 8 + 4 + 8 + 4 + 16 + 4) * 8 = 3680 Board score = 32 * 2 + 32 + 32 + 32 * 2 + 4 + 8 + 16 + 4 = 224 **current_field_score =** merge score + monotonic score + board score = 1635 + 3680 + 224 = 5539