THE TECHNICAL UNIVERSITY OF ŁÓDŹ

Faculty of Technical Physics, Information Technology and Applied Mathematics,

Information Technology

|  |
| --- |
| *Bachelor of Engineering Thesis*An analysis of min-max algorithm implementations in logic gamesPaweł CiążyńskiStudent’s number: 171120 |

###### Supervisor:

###### PhD Włodzimierz Mosorow

Łódź, 2016

POLITECHNIKA ŁÓDZKA   
Wydziałem Fizyki Technicznej, Informatyki i Matematyki Stosowanej

#### Paweł Ciążyński

#### PRACA DYPLOMOWA INŻYNIERSKA

Analiza implementacji algorytmu min-max  
w grach logicznych

Łódź, 2016 r.

Opiekun: Dr Włodzimierz Mosorow

**STRESZCZENIE**

Człowiek ma styczność z grami cały czas. Niekiedy nie zdaje sobie nawet z tego sprawy. Czy można nazwać grą moment, w którym człowiek śpieszy się do pracy i próbuje dojechać tam w krótszym czasie niż jego współpracownicy? To także może być grą, a jak każdą grę – można ją rozwiązać. W swojej pracy pragnę przedstawić czytelnikowi w jaki sposób można tego dokonać i co dokładnie znaczy „rozwiązać grę”. Praca jest podzielona na cztery części. W pierwszej części przybliżam czytelnikowi teoretyczne aspekty sztucznej inteligencji w grach logicznych. Wprowadzam różne sposoby rozwiązywania gier. W drugiej części przedstawiam na czym polega algorytm min-max, jego złożoność oraz różne odmiany. Trzecia część jest częścią praktyczną, gdzie przedstawiam implementację oraz działanie algorytmów dla dwóch gier logicznych. Ostatnia, czwarta część podsumowuje i analizuje wyniki działania różnych odmian algorytmu dla poszczególnych gier oraz prezentuje, która odmiana algorytmu min-max działa najlepiej.

TECHNICAL UNIVERSITY OF LODZ   
FACULTY OF *Technical Physics, Information Technology and Applied Mathematics*

Information Technology

#### Paweł Ciążyński

#### BSc THESIS

#### An analysis and comparison of differentmin-max algorithm implementations in logic games

Lodz, 2016

Supervisor: PhD Włodzimierz Mosorow

**ABSTRACT**

*[streszczenie po angielsku]*

**Content**

[1. Introduction 6](#_Toc446432394)

[2. Definitions 7](#_Toc446432395)

[2.1. Definition of a game 7](#_Toc446432396)

[2.2. Game tree 8](#_Toc446432397)

[2.3. Definition of Artificial Intelligence 9](#_Toc446432398)

[2.4. Solving games 10](#_Toc446432399)

[3. Game solving Methods 10](#_Toc446432400)

[3.1. How to solve the game? 10](#_Toc446432401)

[3.1.1. Depth-first search (DFS) 10](#_Toc446432402)

[3.1.2. Breadth-first search (BFS) 11](#_Toc446432403)

[3.2. Comparison of DFS and BFS 12](#_Toc446432404)

[3.3. Min-max 13](#_Toc446432405)

[3.3.1. Basic min-max 13](#_Toc446432406)

[3.3.2. Alpha cuts 13](#_Toc446432407)

[3.3.3. Beta cuts 13](#_Toc446432408)

[3.3.4. Alpha/Beta cuts 14](#_Toc446432409)

[4. Project Implementation 14](#_Toc446432410)

[4.1. Common code 14](#_Toc446432411)

[4.2. Connect-four 14](#_Toc446432412)

[4.2.1. Heuristics 14](#_Toc446432413)

[4.2.2. Implementation 14](#_Toc446432414)

[4.2.3. Results 14](#_Toc446432415)

[4.3. Tic-tac-toe 14](#_Toc446432416)

[4.3.1. Heuristics 14](#_Toc446432417)

[4.3.2. Implementation 14](#_Toc446432418)

[4.3.3. Results 14](#_Toc446432419)

[5. Analysis of results 14](#_Toc446432420)

[5.1. Connect-four 15](#_Toc446432421)

[5.1.1. Computational complexity 15](#_Toc446432422)

[5.1.2. Time complexity 15](#_Toc446432423)

[5.2. Tic-tac-toe 15](#_Toc446432424)

[5.2.1. Computational complexity 15](#_Toc446432425)

[5.2.2. Time complexity 15](#_Toc446432426)

[6. Conclusions 15](#_Toc446432427)

[7. References 16](#_Toc446432428)

# Introduction

In these days, computers are very fast machines and can do tons of calculations in a split of second. However, sometimes these computers are not fast enough. The speed is sufficient for solving many small logic games like tic-tac-toe, but people also would like to solve very complicated games like, for example chess.

In my engineering thesis I want to present different approaches on solving logic games. Later on I will concentrate mostly on Min-max algorithm and its various implementations. I will write two games in java programming language (Connect Four and Tic-tac-toe). Then the goal of my thesis is to write and present how to solve the game and solve it using different variations of my Min-max algorithm. I do not aim to write perfect artificial intelligence program for these two games, but to present how these algorithms work in theory and in practice.

# Definitions

……………………………………….

## Definition of a game

A game is a physical or mental activity with set rules, in which participates one or more players. Games are undertaken for enjoyment and sometimes even used as educational tool.

There are different types of games. We can divide games into few categories while taking into account different aspects of them:

* Number of players:
  + No players (e.g. game of life)
  + One player (e.g. solitaire)
  + Two players (e.g. tennis, tic-tac-toe, chess)
  + More players (e.g. soccer)
* Amount of sum:
  + Zero-sum game (e.g. tic-tac-toe, chess) – it’s a game in which the gain (loss) of one player is a loss (gain) for another player.
  + Non-zero-sum game – a game in which the gain for one player does not necessary means loss for the opponent. Example of this game can be Prisoner’s dilemma or Battle of the sexes.
* Information to which player has access:
  + Complete information – situation in which the player has full knowledge about the board situation and strategies available to the other player.
  + Non-perfect information – player knows only some part of the board situation or moves available to his opponent.
* Determinism:
  + Deterministic (e.g. connect-four) – a game in which there is no random factor.
  + Non-deterministic (e.g. poker, dices) – random factor is a common thing in this game.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Number of players** | **Amount of sum** | **Information** | **Determinism** |
| **Chess** | 2 | Zero-sum | Complete | Deterministic |
| **Solitaire** | 1 | Zero-sum | Non-perfect | Non-deterministic |
| **Game of life** | 0 | Non-zero-sum | Complete | Deterministic |
| **Soccer** | 22 | Zero-sum | Non-perfect | Non-deterministic |
| **Poker** | 2 – 8 | Non-zero-sum | Non-perfect | Non-deterministic |
| **Connect Four** | 2 | Zero-sum | Complete | Deterministic |
| **Prisoners’ dilemma** | 2 | Non-zero-sum | Complete | Deterministic |

Logic game is a type of game in which only matters intelligence of a player, ability to find best solution to the given problem. Unlike in physical games, logic games don’t need a player to be played (e.g. game of life).

## Game tree

A game tree is a graph which nodes represent states in the game and connections between them are different possible moves. These graphs are used to present and check all possible moves in subsequent turns.

Different games have got different graph complexity. It means that amount of moves players can perform in each turn is different for each game. For example game chess has got a lot more complex graph than the game Connect-four. Graph complexity also depends on the round of the game. For example in game tic-tac-toe, player who moves first can place his pawn on one out of nine fields, but next player can place his pawn on only one out of eight fields, etc.

Easy and simple way to estimate the games upper graph complexity is to count all possible states on the board. For example in the game tic-tac-toe, all possible states are all variations with repetitions with set of 3 elements placed on 9 fields. It gives us

V=n^k=3^9=19,183

different states.

However this is only the upper estimation of graph complexity. Many games have got symmetric boards. It means that for each axis of symmetry we can divide the amount of states by two, because there exists exactly the same game state, but only rotated or reflected. For example board of game tic-tac-toe has got four axes of symmetry (two diagonal, vertical and horizontal) and one axis of rotation.

## Definition of Artificial Intelligence

Artificial Intelligence (AI) is the ability of a computer program to perform humanlike actions and decisions. Artificial Intelligence can also act as an expert system, or a program for the perception and recognition of shapes.

In 1950 Alan Mathison Turing proposed that ability of a computer system to act as a human being in a conversation with other people can be a test for system’s intelligence. (Turing test).

John McCarthy for the first time coined term ‘Artificial Intelligence’ in 1955. He defined it as ‘the science and engineering of making intelligent machines'. In modern world AI is used to solve many different problems, such as acting as an expert system, data mining, logistics, voice recognition and many others.

## Solving games

In 1913 Ernst Zermelo published an article in which he analyzed the game chess and if for each position on the board there is a mathematical way to determine result of the game (win or loss) and to determine next best move for a player. However Zermelo didn’t answered if the starting position does provide a win for any player, he also stated that if it was a true statement, playing chess would be pointless, because each time we would know the winner even before the start of the game.

What does it mean to solve a game? Solving a game means that we found set of moves that will lead the player to the end of the game with this players win, despite of his opponents’ moves.

# Game solving Methods

…………………………………………………..

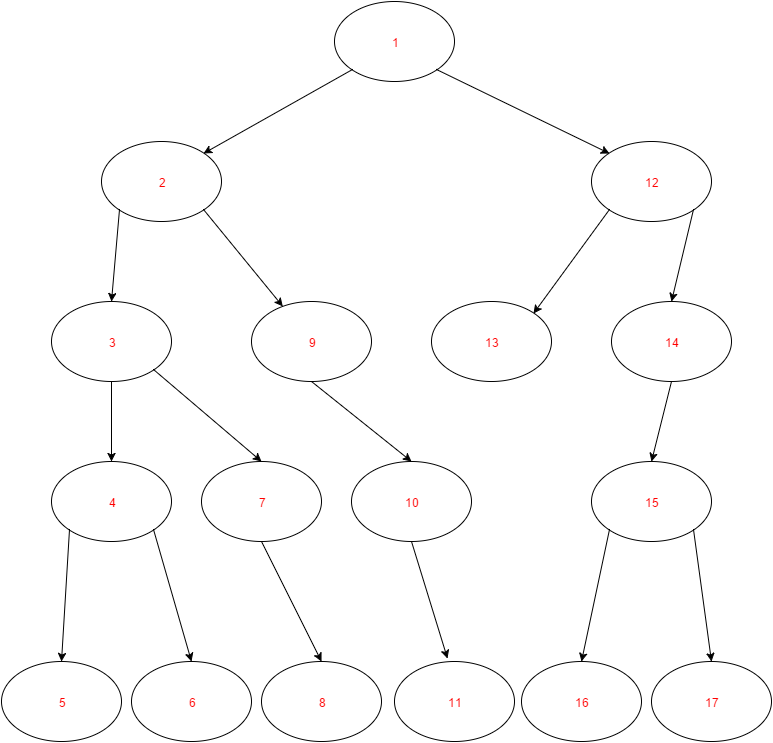
## How to solve the game?

There are many different ways to solve the game. In my thesis I will only concentrate on searching graphs of games. Computer algorithm can create all possible moves, that players can take and then search through the graph. There are also many different ways to find the solution of the graph. We can divide them into two main groups of algorithms: Depth-first search (DFS) and Breadth-first search (BFS).

### Depth-first search (DFS)

This algorithm moves from the top of the graph to the bottom in vertical direction. Algorithm begins from root node (current state of the game) and checks its first child (first possible move from this state of the game). Then it checks the first child of the next node, and so on. When algorithm reaches the state in which there are no more children of currently visiting node, it goes back to the point where it could choose different path. This algorithm usually finds the solution pretty quickly, however it most likely will not be the most optimal solution.

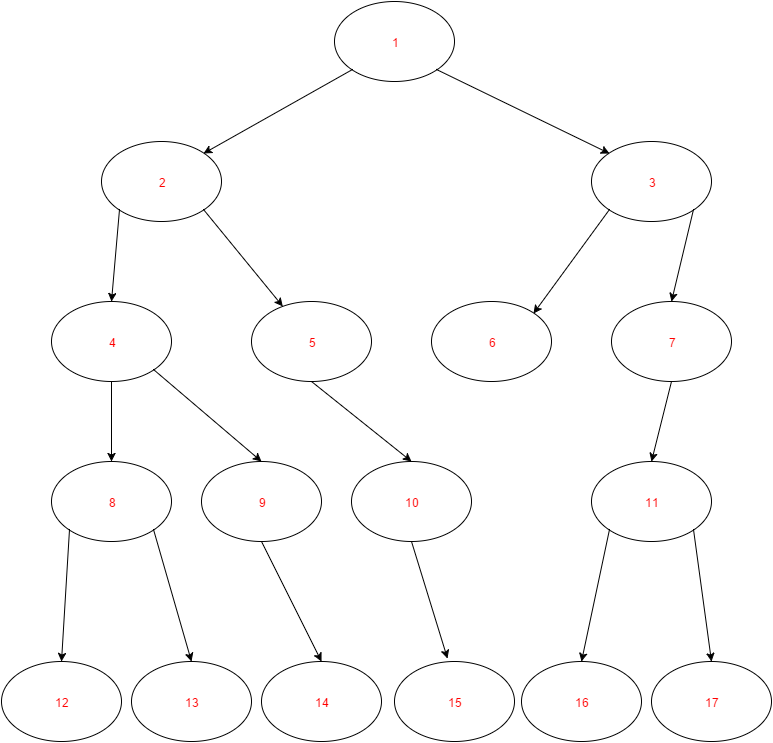
[graph, description]



### Breadth-first search (BFS)

In this method computer moves from the top of the graph (current state of the game) to the bottom in horizontal direction. It means that algorithm visits neighbor nodes before it progress to visit child nodes in the graph. Using this algorithm we are certain to find the shortest possible solution.

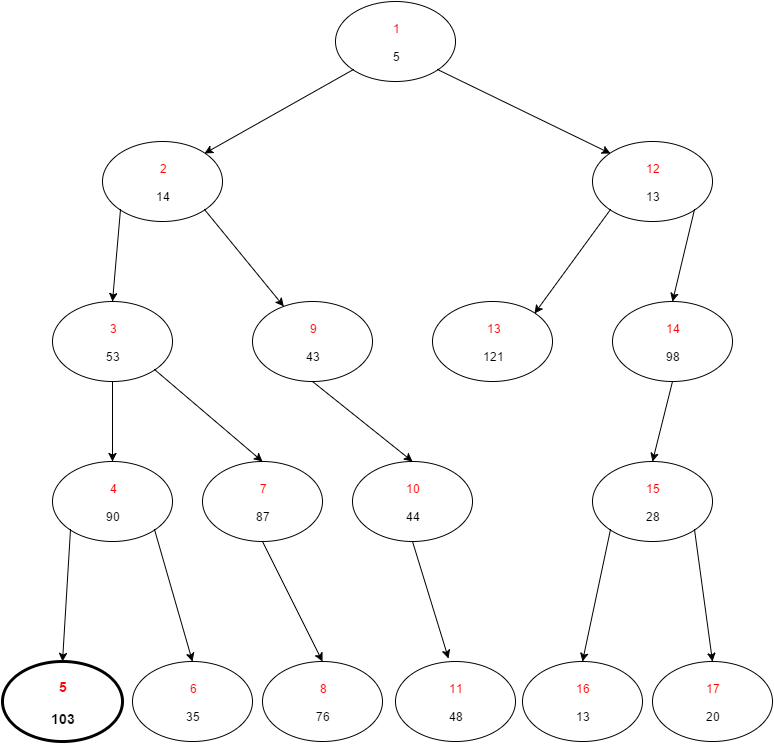
[graph, description]



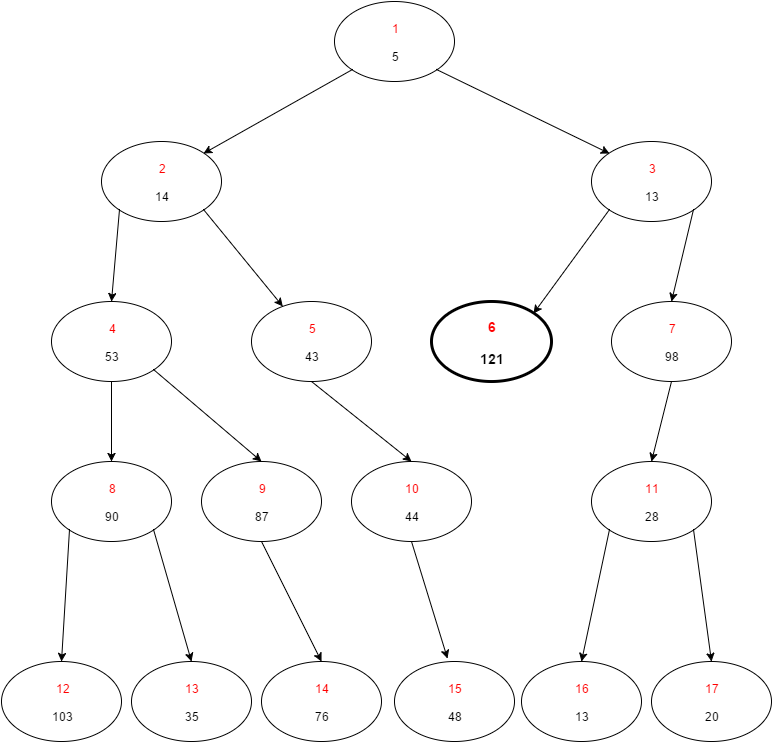
## Comparison of DFS and BFS

Depth-first search and Breadth-first search are two different approaches for graph searching. Time of finding expected node in the tree depends on the given graph. Let us consider the graph below. This is purely random generated graph in which we want to find the node with value greater than 100.

At first let us analyze with DFS algorithm. As we can see, it finds the node with value greater than 100 after checking 5 nodes.



Now let us consider solving the same problem with BFS algorithm. Using BFS approach we found desired node in after checking 6 nodes. Also, this time we found completely different node with value greater than 100. It is due to the fact, that we first check all the nodes which are children of given node.



## Min-max

Up to this point we discussed solving graphs without the division for two players, we only wanted to find one node with given properties. This can be useful in many situations, but now let us consider a game for two players, where

### Basic min-max

### Alpha cuts

### Beta cuts

### Alpha/Beta cuts

# Project Implementation

…………………………….



## Common code

## Connect-four

### Heuristics

### Implementation

### Results

## Tic-tac-toe

### Heuristics

### Implementation

### Results

# Analysis of results

## Connect-four

### Computational complexity

### Time complexity

## Tic-tac-toe

### Computational complexity

### Time complexity

# Conclusions

…………………………….

# References

[1] …………………………………...

[2] …………………………………...

[3] …………………………………...

[4] …………………………………...

[5] …………………………………...

[6] …………………………………...

[7] …………………………………...

[8] …………………………………...

[9] …………………………………...

[10] …………………………………...

[11] …………………………………...

[12]