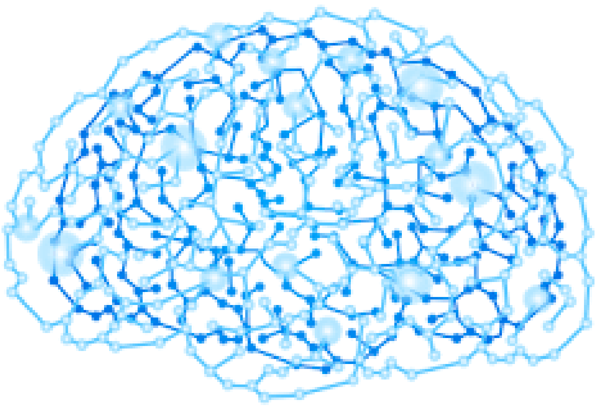
B2 - Gomoku

B-AIA-500

Gomoku

Bootstrap



2.0

Gomoku

language: C, C++, Python3.7 (only standard distribution)

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

At the end of this Bootstrap, you should have a sufficient global knowledge to start working on your algorithms.
3 themes must be introduced before you are able to do this:

1. data model and complexity
2. the game interface and rules
3. basic AI algorithms

Data Model

AI algorithms are by definition highly memory-consuming. Indeed, it would generally be possible to find optimal strategy with infinite time and space… which is not available!

For that reason, the key to a good algorithm is to control use of space and time.

Before starting to program the very behavior of your AI, the first problem you need to deal with is:
"how can I represent my data in memory as efficiently as possible?"

Your program must indeed compute all the information about the running game to play the best possible move.
The way this data is represented is crucial for optimization purposes…
Will you choose to stock as much data as possible (consuming RAM) or to make live computations (consuming CPU)?

Answer these questions and find some suitable data models before going further.

Game Interface

At the end of the project, your program should be able to play against AIs of various difficulty levels. The best ones may even be selected for a competitive tournament. For that reason, it is necessary your program complies with the same rules as everyone else:

1. Game rules. We use a simplified version here. Every player plays a stone at his/her turn, and the game ends as soon as one has 5 stones in a row (vertical, horizontal or diagonal)
2. Game protocol. We decided to play with the Piskvork protocol, available here http://petr.lastovicka.sweb.cz/protocl2en.htm

Before starting to develop your AI, **implement Piskvork [protocol]**. We actually recommend you quickly write and test a dumb AI (say, random) to make sure you comply with the interface.

Only the mandatory commands are required.

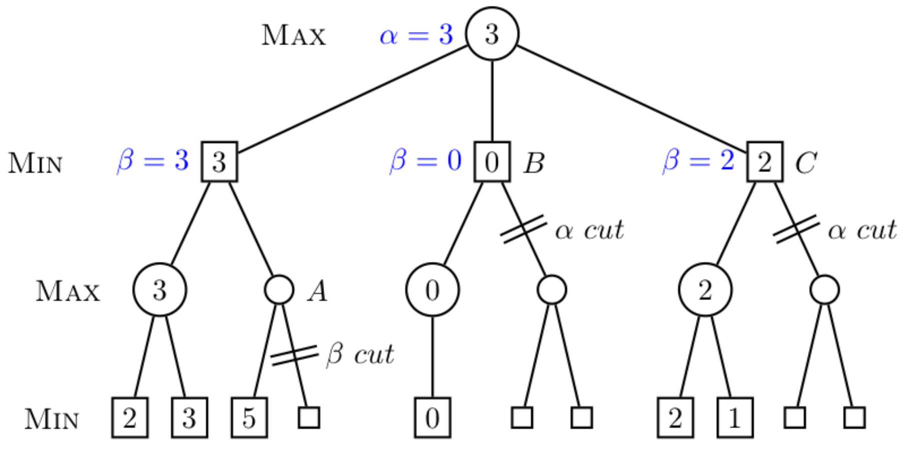
Algorithms

Once you brainstormed about data model and you have implemented the Piskvork protocol, you need to think about the proper AI algorithm for your bot.
Several choices seem relevant, and you have to study, understand and try.

The most obvious choice would be the **MinMax** algorithm (along with alpha-beta pruning and a couple of optimizations). Here you mostly have to design smart rules and control depth.

You could also try a statistical algorithm, based on **Monte-Carlo** method for instance. These techniques mix up pretty well with the former ones.

Finally, you could go for something completely different but very powerful, and dig for **Machine Learning** based algorithms. Be careful to use only methods you are able to implement yourself, since scientific libraries are not allowed for that project (no tensorflow, scikit-learn or scipy for instance). If you have mastered the functional programming projects, you already know some of them.



Google all the terms you don't perfectly master, write and execute bits of code, read some papers, discuss with your mates, search for optimizations,. . .

By the way, any idea on how to optimize your **evaluation function**?