{EPITECH}

GOMOKU

SMART BOTS FOR A SIMPLE GAME

GOMOKU

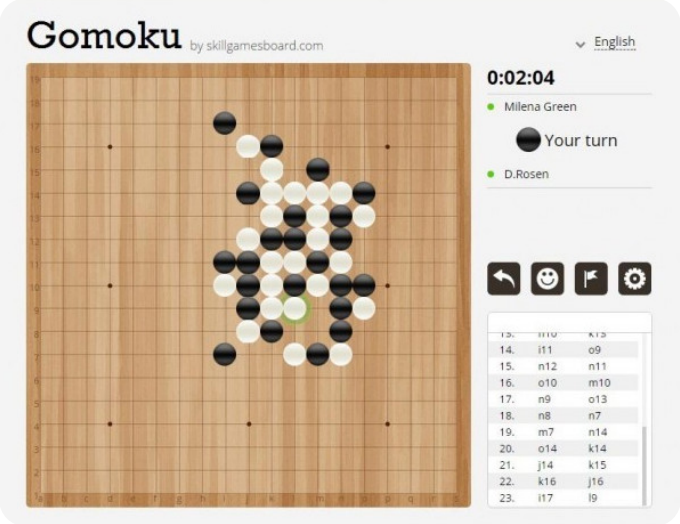
Binary name: pbrain-gomoku-ai
Language: Everything working on “the dump”
Compilation: When necessary, via Makefile, including re, clean and fclean rules

- ✓ The totality of your source files, except all useless files (binary, temp files, obj files,…), must be included in your delivery.
- ✓ All the bonus files should be in a directory named bonus.

The goal of this project is to implement a *Gomoku Narabe* game bot (also called *Wuzi Qi*, *Slope*, *Darpion* or *Five in a Row*), focusing on the performance of its artificial intelligence.

Your bot must be compliant with the communication protocol, at least the **Mandatory commands** part.

liskvork is freely available on the internet as binaries or directly as source code.
Challenge them to evaluate your bot’s strength!
You can also use piskvork (Windows only!) if you want to play yourself against your AI.



EPITECH

Feel free to implement an algorithm of your choice for your bot (Min-max, Monte-Carlo, Machine Learning or other). You will be evaluated on the efficiency of your bot, and on this criteria alone.

You may need to develop a rules management algorithm; do not hesitate to enrich your board representation and your data structures to optimize this algorithm!

There are some technical constraints you must comply with:

- ✓ whatever development language you choose, your program must compile **on Linux** using your makefile (if compilation is needed)
- ✓ **only standard libraries are allowed. No tensorflow or scikit-learn here**

Your program is to be compiled (if needed) and tested versus other programs automatically

Game Rules

This is a 2-player game that is played on a 20x20 game board (called **goban**). We use a very simplified version of the game. Every player plays a stone at their turn, and the game ends as soon as one has 5 stones in a row (vertically, horizontally or diagonaly) - and thus wins.

Evaluation

Your bot will be evaluated based on its results in actual game playing, via a 3-step process:

- ✓ **play to win**
  In order to validate the module, your program is intended to understand some winning situations and play the right move. We will send you pre-filled board and your program must make the right move, for example if there is a winning move your program must play it.
- ✓ **outsmart local bots**
  Programs are matched against bots of low to medium levels. The aim is to maximize your number of victories. Those who accumulate enough victories qualify for the tournament

Technical constraints

During the whole process, the rules are as follows:
- **5 seconds** maximum per move,
- **70 MB** of memory per bot,
- a forbidden move automatically leads to defeat,
- the use of a forbidden library leads to elimination.

Documenting how your AI works sounds like an excellent idea :)

v5