Data Structures

|  |  |
| --- | --- |
| **Name/ Structure named defined using “using”** | **Description** |
| BoardSequence | Used to store full or partial parts of the board; evaluates to std::vector<char> |
| MoveHistory | Used to save the history of moves of a player; evaluates to std::vector<std::pair<Player\*, Position>> |
| analysis | A tuple to store the analysis of a move -- evaluates to  std::tuple<int, int, int, int, int, int, Score, Score, int, std::string>  Contains information about whether it is a winning move or not, whether the opponent can win, the differential in score after the move, the differential in score if opponent makes the move, the capture differential, opponent capture differential, and whether opponent can capture two plies ahead, the psedo score, the negative distance the center and the position  Initial plan was to create a class, but I ran out of time; A vector of this tuple is then sorted to find the most optimal move |
| priority queues | Usesd to store the move and their score for different types of moves; this was later discarded for one single tuple instead of having multiple queues and if blocks |
| StrategicMove | evaluates to a std::pair<Position, std::string>; the second string is the rationale for the move |
| map | Used to player to stones, stones to number of captures, and player to score |
| vector | Used for storing various other data structures -- no arrays used |
| set | Used to return list of moves -- for example, available moves, winning moves, capturing moves, etc |
| Table | Used to represent board, scores, and history for display -- from tabulate |
| graph | Not explicitly used in the code, but MoveAnalysis and Board use graph algorithms for various methods |