РЕШЕНИЕ ШАХМАТНЫХ ЗАДАЧ

Автор: Евтушенко Ян, студент I курса.
Руководитель: Кульман Татьяна Николаевна, доцент кафедры Информационных технологий, к.т.н.
Образовательное учреждение: ГОУ Московской области Международный университет природы, общества и человека «Дубна», филиал «Протвино», г. Протвино.

SOLVING CHESS COMPOSITIONS

Евтушенко Я.

Особенно интригующая область
программирования – задачи так называемого
искусственного интеллекта.
Никлаус Вирт

Направление искусственного интеллекта связано с решением задач тех видов человеческой деятельности, которые традиционно считаются интеллектуальными. К таким задачам относятся и шахматные задачи, решение которых требует большого творчества и поэтому, написание программ ведения шахматных партий, решения шахматных задач включают в себя изучение и применение многих математических и программистских методов.

Шахматными программами занимались и занимаются многие видные математики и программисты, такие как, К. Шеннон, М.М. Ботвинник, Г.М Адельсон-Вельский [1, 2, 3] и др. В шахматных программах достаточно точно сформулированы цели и формализованы ситуации по сравнению с другими задачами, имеющими отношение к искусственному интеллекту. Методы, создаваемые в шахматном программировании, имеют более широкие области применения, некоторые из них имеют прикладное значение [2, 3]. Кроме того, существуют объективные критерии оценки силы шахматных программ, выгодно отличающих их от других задач, моделирующих сложные процессы в системах управления.

В данной работе применяется метод *проб и ошибок и алгоритм перебора с возвратами*. Как правило, задача разделяется на отдельные подзадачи [4], решаемые с помощью *рекурсии*. Алгоритм поиска решения сводится к перебору вариантов и отбрасыванию ложных путей. Процесс решения рассматривается как процесс поиска, строящий дерево подзадач. Более подробное описание методов можно найти в [5].

Основные возможности программы:

- ввод любой допустимой позиции (вводить позицию с уже стоящим матом нельзя),
- нахождение мата в определённое количество ходов и, если такого не имеется, получать сообщение об этом.
- разыгрывание партий.

Программа написана на объектно-ориентированном языке программирования C# с использованием библиотеки .NET 3.5.

Для создания программы были последовательно решены 3 задачи:

- создание графического интерфейса доски,
- разработка компьютерного представления доски и правил игры,
- создание алгоритма, позволяющего искать мат на заданную глубину.

1. Создание графического интерфейса доски

Для решения этой задачи была создана форма (Form1), которая содержит в себе 64 кнопки, которые являются клетками доски. На каждую из них накладывается изображение фигуры, стоящей на ней, или пустой клетки. Также был создан класс Doska, содержащий массив структур, каждая из которых включает:

- bool AttackWhite – показывает атакована ли данная клетка белыми;
- bool AttackBlack – чёрными;
- структуру figur – указывает, какая фигура стоит на клетке;
- структуру color – указывает цвет фигуры.

В Form1 также добавлены события обработки клика для каждой из клеток и переменная bool click, при ложном значении которой фигура выделяется, а при истинном перемещается, если ход корректен, или убирается выделение, меняя значение click на обратное.

Главная форма также содержит кнопки:

- Start – выполняет расстановку стартовой позиции, а во время партии прерывает игру.
- buttonSvPos – позволяет ввести в Form1 собственную позицию.

2. Разработка компьютерного представления доски и правил игры

Для решения этой задачи для каждой фигуры был создан класс, проверяющий возможность её хода из начальных координат(xs,ys) в конечные (xe,ye). Координаты хранятся в классе Doska изменяются в Form1 в зависимости от нажатой клетки и состояния click.

Вот список этих классов:

- Bishop – содержит метод bishop(Doska d), принимающий в качестве аргумента доску, для которой нужно проверить возможный ход слоном.
- Rook – содержит метод rook(Doska d).Проверяет возможность хода ладьёй и, если ход возможен, изменяет соответствующий флажок в объекте d, показывающий какой ладьёй совершён ход, что в последствии будет использовано при проверке возможности рокировки.
- Peen – содержит метод peen(Doska d). Проверяет возможность хода пешкой.
- Knight – содержит метод knight(Doska d). Проверяет возможность хода конём.
- King – содержит метод king(Doska d). Проверяет возможность хода королём и запрещает делать им ход если клетка (xe,ye)находится под ударом фигуры противоположенного цвета.
- Queen – содержит метод queen(Doska d). Выполняет проверку на возможность хода ферзём. Медод queen(Doska d)-объединение методов bishop(Doska d) и rook(Doska d).

Также создан класс KlAttack, в котором, используя функциональность приведённого выше списка классов, метод proverka(Doska d) проверяет каждую клетку на возможность хода на неё какой-либо фигурой и соответственно меняет для неё поля AttackWhite и AttackBlack. И если на поле стоит король, и оно атакованно фигурой противоположенного цвета, то в классе CheckMate меняется значение поля CheckW или CheckB, указывающее на то, что белым или чёрным стоит шах.

Класс CheckMate содержит два метода: provMat(Doska d) и provPat(Doska d), и два поля: pat и mat, которые являются результатами проверки этих методов.

provMat выполняет проверку доски путём создания её копии (для этого в классе Doska есть конструктор не по умолчанию) и, перебирая все возможные ходы, проверяет эту копию на наличие шаха. Если во всех случаях шах ходящей стороне сохраняется (если изначально ни одной из сторон не стоит шах, то управление передается provPat), то поле mat становится равным true, и выводится сообщение о том, что стоит мат, затем

в Form1 блокируются все кнопки, кроме Start. Метод provPat выполняет проверку на наличие пата. Принцип проверки схож с provMat, но есть одно отличие: метод работает с доской только если CheckW и CheckB ложны.

В проекте созданы еще 2 формы: FormPeenB и FormPeenW для замены пешки, по её достижению последней горизонтали, на другую фигуру. В Form1 создается новый поток с одной из данных форм (в зависимости от цвета ходящей фигуры), после чего главный поток ждёт его завершения. В FormPeenB или в FormPeenW игрок должен выбрать фигуру, которая заменит пешку. Как только игрок сделает выбор, поток прекратит свою работу, и Form1 поменяет в объекте класса Doska пешку на выбранную фигуру.

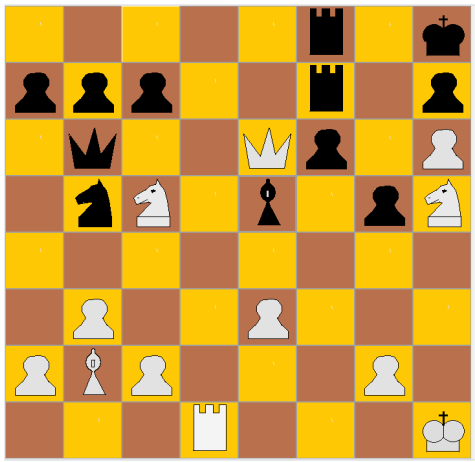
Также Form1 содержит коллекцию досок. После каждого хода в неё добавляется копия объекта доски и выполняется проверка, и если 3 доски в коллекции будут идентичны, то Form1 блокирует все кнопки, кроме Start, и выводит сообщение “Ничья из-за троекратного повторения ходов”. Также будет выведено сообщение о ничьей, если поле HodiBezVsatia будет равно 100. (Оно увеличивается на 1 после каждого хода, а после хода пешкой или взятия фигуры становится равным единице).

3. Создание алгоритма, позволяющего искать мат на заданную глубину

Для решения задач (например, поиска мата), создан класс YanDvij, который включает в себя методы MVOH(Doska d) и poiskMata(Doska d, byte glubina), поля score и tecGlub.

MVOH(Doska d) – ищет мат в 1 ход, переберая все возможные ходы за ходящую сторону, и выводит сообщение о том, найден мат или нет.

poiskMata(Doska d, byte glubina) – ищет мат на заданную глубину(в полуходах). Метод рекурсивен и использует поле tecGlub для отслеживания текущей глубины, увеличивая его на 1, если делается ход, и уменьшая его на 1, если делается возврат хода. Если глубина поиска равна 1, то просто выполняется проверка MVOH() и метод завершает свою работу. Если же глубина больше, то функция выполняет MVOH и, если мата нет, начинает перебирать все возможные ходы (после каждого хода вызывая саму себя). Эта программа позволяет решать задачи подобные этой:



Мат в 3 хода.
Решение: 1.Q:f7 R:f7 2.Rd8+ Rf8
3.R:f8#

Библиографический список

1. Ботвинник М.М. О кибернетической цели игры. – М.: Советское радио, 1975.
2. Ботвинник М.М. Шахматный метод решения переборных задач. – М.: Советский спорт, 1989.
3. Адельсон-Вельский Г.М., Арлазаров В.Л., Битман А.Р., Донской М.В. Машина играет в шахматы. – М.: Наука, 1983.

4. Вирт Н. Алгоритмы и структуры данных – СПб.: Невский диалект, 2001.
5. Корнилов Е.Н. Программирование шахмат и других логических игр. – СПб.: БХВ-Петербург, 2005.