МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САМАРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ

Приложение 2
к рабочей программе дисциплины

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ДЛЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

Системы искусственного интеллекта

(наименование дисциплины(модуля)

Направление подготовки / специальность

09.03.03 Прикладная информатика
(код и наименование)

Направленность (профиль)/специализация

Прикладная информатика на железнодорожном транспорте
(наименование)

Содержание

1. Пояснительная записка

Цель промежуточной аттестации – оценивание промежуточных и окончательных результатов обучения по дисциплине, обеспечивающих достижение планируемых результатов освоения образовательной программы.

Перечень компетенций, формируемых в процессе освоения дисциплины

Код и наименование компетенции
ПК-3 - Способен разрабатывать графический дизайн интерфейса, проектировать пользовательские интерфейсы по готовому образцу или концепции интерфейса
ПК-3.3 - Использует методы искусственного интеллекта (машинного обучения) и анализа больших данных для решения прикладных задач

Результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения образовательной программы

Код и наименование индикатора достижения компетенции	Результаты обучения по дисциплине	Оценочные материалы
ПК-3 - Способен разрабатывать графический дизайн интерфейса, проектировать пользовательские интерфейсы по готовому образцу или концепции интерфейса	Обучающийся знает: методы анализа и работу исчислительных алгоритмов над логическими моделями	
	Обучающийся умеет:определять типы логических моделей, прототипы моделей и строить интеллектуальную систему	
	Обучающийся владеет:приемами построения логических систем вывода по экспертным правилам	
ПК-3.3 - Использует методы искусственного интеллекта (машинного обучения) и анализа больших данных для решения прикладных задач	Обучающийся знает: общую методологию в области искусственного интеллекта	
	Обучающийся умеет: выявлять эвристические методы, применяемые в системах.	
	Обучающийся владеет: принципами построения систем искусственного интеллекта.	

Промежуточная аттестация (экзамен) проводится в одной из следующих форм:
1) ответ на билет, состоящий из теоретических вопросов и практических заданий;
2) выполнение заданий в ЭИОС СамГУПС.

2. Типовые[1] контрольные задания или иные материалы для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих уровень сформированности компетенций

2.1 Типовые вопросы (тестовые задания) для оценки знаниевого образовательного результата

Проверяемый образовательный результат

Код и наименование индикатора достижения компетенции	Образовательный результат
ПК-3 - Способен разрабатывать графический дизайн интерфейса, проектировать пользовательские интерфейсы по готовому образцу или концепции интерфейса	Обучающийся знает: основные компоненты электронно - образовательной среды СамГУПС, доступные для обучающихся, основные системы видеоконференцсвязи ЭИОС, возможности ЭИОС для синхронного и асинхронного взаимодействия в рамках образовательного процесса, доступные в ЭИОС электронные библиотеки. Основные сервисы Microsoft Office 365, интегрированные в ЭИОС университета. Основные онлайн-сервисы и площадки, используемые в процессе самообразования
ПК-3 - Способен разрабатывать графический дизайн интерфейса, проектировать пользовательские интерфейсы по готовому образцу или концепции интерфейса	Обучающийся умеет: получать доступ к учебным планам, рабочим программам дисциплин (модулей), практик, к изданиям электронных библиотечных систем и электронным образовательным ресурсам, указанным в рабочих программах, использовать возможности систем видеоконференцсвязи для учебной (научной) работе и самообразования, с использованием средств ЭИОС, участвовать в проведении всех видов занятий, процедур оценки результатов обучения, реализация которых предусмотрена с применением электронного обучения, дистанционных образовательных технологий. Формировать свое электронное портфолио, в том числе сохранять свои работы, рецензий и оценки на них. Устанавливать на мобильные устройства сервисы ЭИОС университета, приложения Microsoft Office 365 и использовать их в учебной (научной) работе и самообразовании
ПК-3 - Способен разрабатывать графический дизайн интерфейса, проектировать пользовательские интерфейсы по готовому образцу или концепции интерфейса	Обучающийся владеет: навыками синхронного и (или) асинхронного взаимодействия посредством сети "Интернет" с использованием средств ЭИОС между участниками образовательного процесса. Навыками фиксации хода образовательного процесса, результатов промежуточной аттестации и результатов освоения программы бакалавриата в своем портфолио. Навыками использования сервисов ЭИОС университета, приложениями Microsoft Office 365 в процессе учебной (научной) работы и самообразовании

[1]Приводятся типовые вопросы и задания. Оценочные средства, предназначенные для проведения аттестационного мероприятия, хранятся на кафедре в достаточном для проведения оценочных процедур количестве вариантов. Оценочные средства подлежат актуализации с учетом развития науки, образования, культуры, экономики, техники, технологий и социальной сферы. Ответственность за нераспространение содержания оценочных средств среди обучающихся университета несут заведующий кафедрой и преподаватель – разработчик оценочных средств.

2.2. Примеры типовыхзаданий

Задание: Построение графа

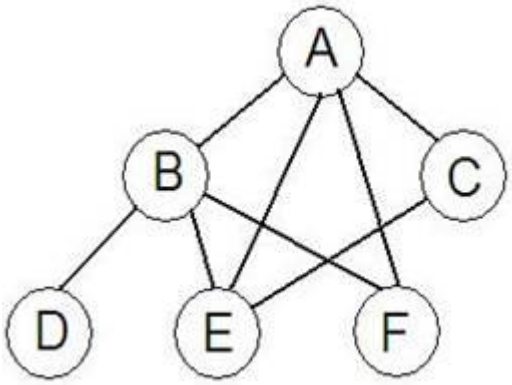
В рамках работы требуется написать программу, позволяющую отобразить на экране граф, информация о котором указана в определённом (на усмотрение студента) формате в обычном текстовом файле.

Для определённости и возможности отображения того, что будет указано в файле, ограничим число уровней графа четырьмя (4), а максимальную ширину одного уровня графа семью (7) вершинами. Граф таких размеров можно легко уместить на экране любого монитора. Граф будет иметь одну вершину, которую мы условно назовём корневой и договоримся, что при отображении корневой будет та вершина, которая указанна в файле первой.

В каком виде указывать вершины графа и какими дополнительными данными сопровождать каждую из вершин – дело ваше, главное, чтобы на основании указаний в файле программа смогла построить именно тот граф, который мы ожидаем.

Чтобы прояснить ситуацию, приведём пример.

Допустим, мы хотим, чтобы программа построила и отобразила следующий граф:



Для того чтобы задать этот (или любой подобный) граф, необходимо знать о нём следующее:

- какой узел является корневым;

- на скольких уровнях расположены вершины графа;

- какова ширина самого широкого уровня;

- какое максимальное число потомков могут иметь узлы графа;

- список потомков для каждого узла.

Владея подобной информацией для любого графа, можно написать программу, которая построит его и отобразит на экране.

Чтобы не указывать в текстовом файле все перечисленные сведений о выводимом графе, из списка нужно выбрать только те, которые действительно необходимы, а именно: список вершин графа и их потомков. Корневой узел, как было сказано, заносится в файл первым, поэтому специально обозначать его не следует. Имея список вершин графа и потомков каждой вершины, всегда можно посчитать число уровней графа, максимальную ширину уровня и

максимальное число потомков некоторой вершины. Таким образом, для отображения любого графа необходимо иметь лишь список вершин и потомков каждой вершины. Список потомков можно разделять, например, запятыми, а имя вершины от списка потомков можно отделять, например, двоеточием. Приведём пример такого текстового файла, в котором содержится информация о нарисованном графе:

A: B, E, F, C

B: D, E, F

C: E

D:

E:

F:

Имея файл с подобным содержимым, легко построить граф и вывести его узлы на экран. Каждый узел графа в программе следует представить либо структурой, либо объектом, либо каким-то другим способом, который вы придумаете, если придумаете. Обратите внимание, что на основании представленного файла легко узнать максимальное допустимое число потомков некоторого узла графа (В данном случае это будет 4 – число потомков узла A).

Узнав максимально возможное число потомков узла, в языке, например, СИ, можно объявить для его хранения примерно такую структуру:

struct node {

char: name;

struct node *children[4];

struct node *parent;

};

Поле name будет хранить букву, соответствующую вершине (в следующих работах вместо буквы потребуется хранение состояния игрового поля), в массиве children будут храниться указатели на потомков текущего узла, а поле parent будет содержать указатель на родительский узел (в данной работе он не нужен, а в следующий будет очень полезен).

Всё, структура есть, осталось только на основании данных в текстовом файле сформировать требуемое дерево и вывести его каким-либо образом на экран.

Вывод на экран может быть реализован аналогично тому, как это делалось в лабораторных работах на втором курсе (каждая вершина содержала координаты, в которые её необходимо вывести), либо более интеллектуальным способом. Каким – решать вам. Обратите внимание, что каждую из вершин следует отображать на уровень ниже последнего встречающегося в файле родителя вершины. Например, хотя узел E является потомком узла A, его следует отображать не на следующем за A уровнем, а через уровень, потому что помимо вершины A родителем узла E являются узлы B и C, которые как раз и располагаются на уровень ниже A.

Работа №2

Задание: решение головоломки «Восьмёрки» с использованием поиска в пространстве состояний

Выполнение работы можно условно поделить на 3 этапа:

1. Построение полного дерева пространства состояний для «восьмёрок».

2. Нахождение текущей комбинации игрового поля в пространстве состояний с помощью поиска «в ширину» или поиска «в глубину».

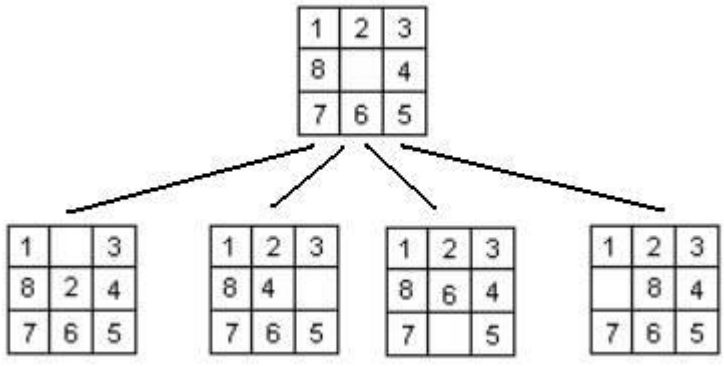
3. Получение на основании имеющегося дерева таких комбинаций игрового поля, последовательный переход по которым приведёт к решению поставленной задачи – сбору головоломки.

Рассмотрим каждый из этапов немного подробнее.

Построение дерева выполнялось вами в первой лабораторной работе и в случае, если всё было сделано правильно – весь наработанный код можно легко использовать при реализации данной работы. Дерево пространства состояний для восьмёрок должно строиться из целевой комбинации, которой является следующая:

1	2	3
8		4
7	6	5

В отличие от предыдущей работы, в данной конкретной задаче число потомков каждого узла будет максимум 4, как, например, при построении их из исходной комбинации. Каждый потомок некоторой вершины будет содержать возможные состояния игрового поля, в которые можно попасть, сделав ход из текущей вершины. Перед добавлением некоторого потомка в дерево пространства состояний необходимо убедиться, что там его ещё нет. В противном случае, дерево рискует получиться бесконечно большим. Из целевой комбинации можно сделать 4 различных хода: сдвинуть «8» вправо, «4» влево, «2» вниз или «6» вверх. Отобразив это в виде дерева, получим следующую картину:



Каждая из листовых вершин представленного дерева содержит состояние игрового поля, из которого можно сделать 3 различных хода, один из которых уже есть в представленном дереве: это корень дерева. По этой причине, каждый из представленных листовых узлов будет иметь не 3, а только 2 потомка. Продолжая построение дерева подобным образом, можно в итоге прийти к ситуации, когда ни из одной листовой вершины нет возможности сделать ход, который ещё не делали. При достижении подобной ситуации следует прийти к выводу, что дерево решений (пространство состояний игры «восьмёрки») построено, и его можно использовать.

При наличии построенного дерева, можно переходить к реализации интерфейса игры. Пользователь должен иметь возможность видеть игровое поле в каждый момент времени, должен иметь возможность перемещать фишки игрового поля допустимым образом, а также должен иметь возможность, нажав некоторую кнопку, запустить процесс автоматического сбора головоломки. В момент запуска автоматического сбора следует запустить нахождение текущей комбинации игрового поля в пространстве состояний с помощью поиска «в ширину» или поиска «в глубину». Какой из видов поиска выбирать – дело ваше. Главное, чтобы искомая комбинация присутствовала в построенном дереве решений. Независимо от выбранного вида поиска, пройденный по дереву путь следует сохранять.

Наличие сохранённых комбинаций, по которым выполнялся поиск, позволит получить путь от текущего состояния игрового поля к целевому. О том, как это сделать – было сказано на лекциях. После получения пути решения, его следует отобразить в виде последовательности деланных ходов для сбора головоломки, после чего задачу можно считать решённой.

В процессе выполнения работы возможно появление следующих проблем:

1. Построение дерева пространства состояний вызывает нехватку памяти. Если это произошло – введите в программу возможность строить не полное дерево, а дерево до какого-то конкретного уровня. Практика показывается, что, построив дерево хотя бы до уровня 8 или 9, вы легко сможете отладить весь остальной код.

2. Всё сделано правильно, но иногда программа зависает, уходя в бесконечный цикл. Обычно это происходит из-за того, что первоначальная комбинация игрового поля формируется случайно расстановкой фишек в случайные позиции. Подобный подход быстр, но в половине случаев приводит к зацикливанию алгоритма, поскольку половина комбинаций из всех возможных случайных расстановок фишек сбору по правилам поддаваться не будет. Избежать этого можно очевидным способом - в начале работы программы для получения комбинации, требующей сборки, необходимо размешивать собранную комбинацию.

3. Дерево построено, комбинация игрового поля найдена, а как получить путь к вершине дерева на основе имеющегося списка пройденных состояний – не совсем понятно. Решить эту проблему можно введением в каждый узел дерева указателя на родительский узел. Тогда, зная в каком из узлов вы в данный момент находитесь, можно всегда путём перехода по ссылками на родительские узлы подняться к вершине.

Задание: реализация интеллектуального противника в игре «крестики-нолики» с использованием поиска в пространстве состояний

Задание аналогично предыдущему, с разницей лишь в том, что программа предназначена не просто для сбора головоломки, а для игры с человеком, который по своей природе враждебен. Для того чтобы компьютер мог ходить и наносить урон противнику , его действия должны быть на чём-то основаны. В данной работе основанием предлагается сделать дерево решений игры «крестики-нолики». Для простоты можно условиться, что первым всегда ходит человек. Таким образом, дерево решений программы следует начинать строить с того состояния игрового поля, в котором оно окажется после первого хода человека.

После построения дерева, компьютеру следует найти в этом дереве ближайшую к корню собственную выигрышную комбинацию, а также ближайшую к корню выигрышную комбинацию противника. Если выяснится, что ближайшая собственная выигрышная комбинация ближе к корню, чем выигрышная комбинация противника (человека), то следует сделать ход по той ветке, где находится эта комбинация. Если оказывается, что комбинация

противника ближе – следует найти другую ветку пространства состояний, сделав ход по которой компьютер гарантированно не проиграет раньше, чем сможет выиграть или свести партию вничью.

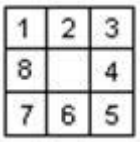
Как видно, задание немного проще предыдущего за счёт меньшего размера пространства состояний. Но и то и другое задание не должно вызвать сложностей труда, если грамотно сделана первая работа.

Работа №3

Задание: решение головоломки «Восьмёрки» или «Пятнашки» с помощью «жадного» алгоритма

Программа должна иметь какой-то интерфейс, элементы которого позволят пользователю перемешать собранную комбинацию фишек, начать собственноручно собирать перемешанную комбинацию и в любой момент нажать на кнопку, которая вызовет функцию автоматического решения головоломки и дособерёт то, что есть. Процесс решения при этом должен отображаться на экране шаг за шагом, с отображением отходов назад, если таковые будут иметь место в процессе нахождения решения. Как видно, по функционалу задача равносильна первой задача из предыдущей лабораторной работы

Использование жадного алгоритма для «пятнашек» и «восьмёрок» аналогично, поэтому для простоты восприятия приведём пример его использовании на примере «восьмёрок». В собранном виде подобная головоломка имеет следующий вид:

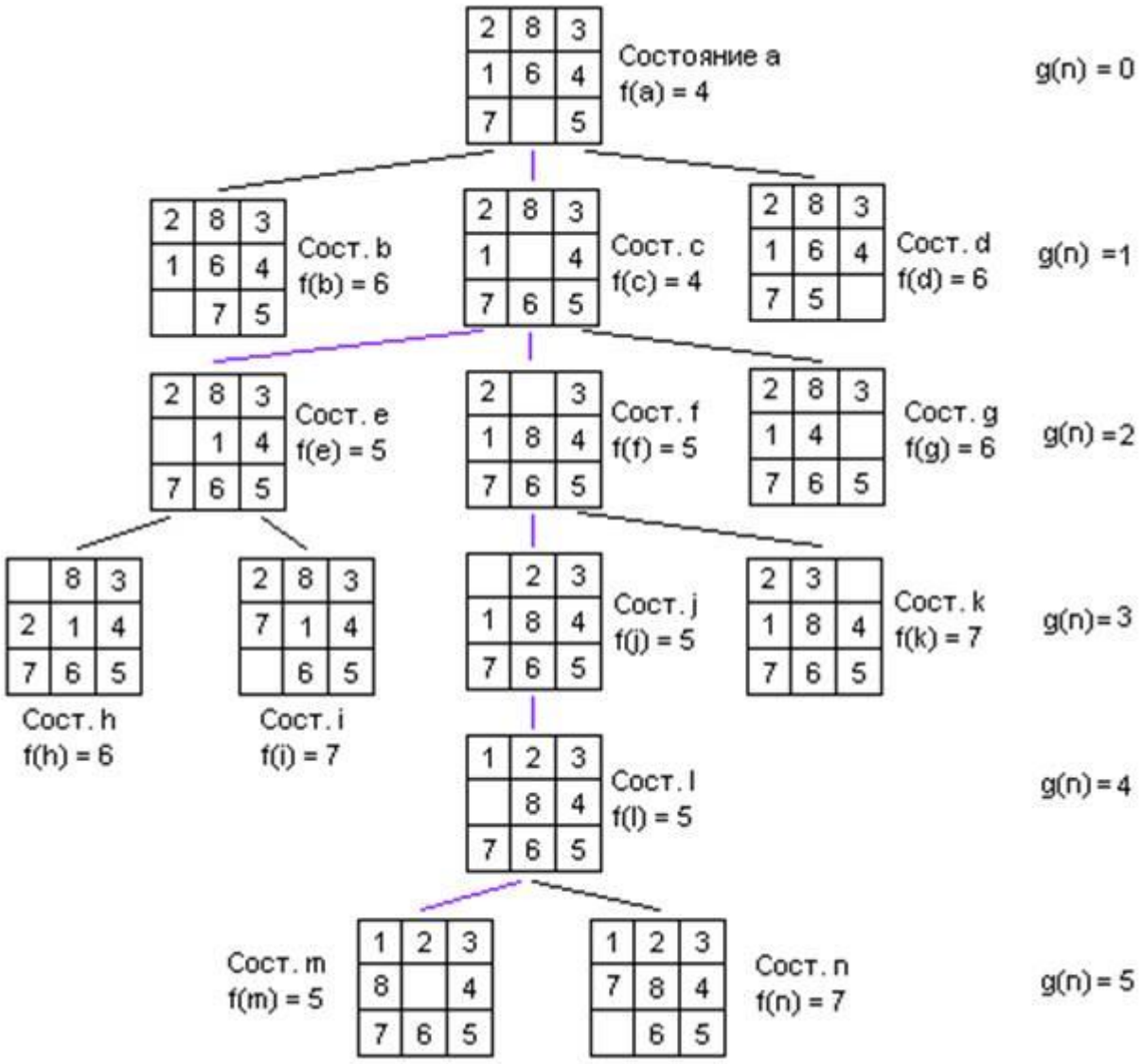


Задача состоит в том, чтобы собрать головоломку из некоторого произвольного состояния в целевое, указанное выше.

Смысл работы сводится к тому, чтобы из возможных вариантов каждого следующего хода выбрать наилучший. Сделать это нам (имеется в виду вам) поможет эвристическая функция следующего вида: f(n) = g(n) + h(n). В этой функции g(n) — фактическая длина пути от произвольного состояния n к начальному (то есть число ходов, которые необходимо сделать для того, чтобы прийти из начального состояния в состояние n), а h(n) — эвристическая оценка расстояния от состояния n к цели.

В самом примитивном случае значением h(n) можно считать число фишек, находящихся не на своих местах. В более продвинутом варианте этой функции учитывается также степень упорядоченности фишек друг относительно друга, что должно привести к более быстрому нахождению правильного пути. Мы рассмотрим менее продвинутый, оставив более продвинутый на самостоятельное изучение.

Приведём пример дерева, благодаря которому происходит нахождение решения из некоторого начального состояния, и поясним то, как с использованием этих данных получить решение задачи. Дерево представлено на рисунке:



Из рисунка видно, что состояние a является начальным, а состояние m — конечным (целевым). Для удобства, каждому состоянию соответствует буква, а рядом с каждым состоянием написано посчитанное значение функции f, соответствующее текущему состоянию. Напротив каждого уровня дерева написано значение функции g(n), где n — номер хода, а следовательно и номер уровня дерева. Чем больше уровень, тем больше значение функции g(n). Обратите внимание, что при подсчёте значения функции h(n) считается только количество фишек не на своих местах. Пустое поле фишкой не считается! Например, для состояния i не на своих местах находятся фишки 1, 2, 7 и 8, поэтому значение функции h(i) будет равно 4, значение функции g(i) равно 3, а значение функции f(i) равно h(i) + g(i) = 4 + 3 = 7.

Жадный алгоритм предоставляет возможность отказаться от использования так нелюбимых студентами графов и вместо них использовать 2 списка, которые при использовании жадного алгоритма по традиции называют open и closed. Список open содержит те состояния игрового поля, которые предстоит рассмотреть для нахождения целевого решения, а список closed содержит уже рассмотренные состояния.

Изначально список open содержит лишь состояние «a», а список closed пуст. Для удобства и лучшего понимания того, что происходит, рядом с именем состояния будем писать значение оценочной функции для этого состояния. Элементы списка open должны быть упорядочены по возрастанию в соответствии со значением оценочной функции.

Итак, изначально у нас следующая картина:

open = [a4]; closed = [ ]

Состояние a в списке open не является целевым, поэтому извлекаем его из списка, меняем на те состояния, которые получаются из «a», а само состояние «a» помещаем в список уже рассмотренных состояний closed. Из состояния «a» можно получить 3 новых состояния: «b», «c» и «d». После проделанных действий содержимое списков open и closed будет следующим:

open = [c4, b6, d6]; closed = [a4]

Состояние «c» имеет наименьшее значение оценочной функции, не является целевым, значит, выигрышное состояние надо искать от него. Перемещаем его из списка open в список closed, а в список open добавляем те состояния, которые получаются из состояния «c», получаем следующие значения в списках open и closed:

open = [e5, f5, b6, d6, g6]; closed = [a4, c4]

Действуя аналогичным образом дальше, последовательно получаем следующие состояния списков open и closed:

open = [e5, f5, b6, d6, g6]; closed = [a4, c4]

open = [f5, h6, b6, d6, g6, i7]; closed = [a4, c4, e5]

open = [j5, h6, b6, d6, g6, k7, i7]; closed = [a4, c4, e5, f5]

open = [l5, h6, b6, d6, g6, k7, i7]; closed = [a4, c4, e5, f5, j5]

open = [m5, h6, b6, d6, g6, n7, k7, i7]; closed = [a4, c4, e5, f5, j5, l5]

После достижения последнего из приведённых состояний видно, что состояние «m» — целевое, а значит, задача решена. Решением будет путь, находящийся в списке closed. Обратите внимание, что использовать его просто в том виде в каком он есть не представляется возможным, поскольку нет возможности за один ход попасть, например, из комбинации «e» в комбинацию «f». Однако в комбинацию «f» можно попасть из комбинации «c», вернуться в которую из «e» можно, сделав один ход назад по списку closed.

Ваша задача — распространить этот метод (при желании) на поле размером 4 x 4 и в процессе работы со списками отображать перемещения фишек на экране для максимальной наглядности.

Задание: реализация интеллектуального противника в любой игре для двух участников с использованием процедуры минимакса

О том, что такое минимакс – всем должно быть известно из курса лекций. В качестве альтернативы к предыдущему варианту третьей лабораторной работы предлагается использовать процедуру минимакса для реализации противника в любой интеллектуальной игре на двух или более человек (например, в игре «крестики-нолики»).

Работа №4:

Задание: распознавание формата файла по содержимому с помощью простейшей искусственной нейронной сети

Работа посвящена овладению минимальными основами построения и обучения искусственных нейронных сетей. Сетей как таковых не будет, да и основы не очень основные, но принципы обучения нейронов данная работа должна раскрыть.

В рамках работы предлагается построить нейронную сеть, состоящую из одного нейрона и обучить её решать какую-либо простейшую задачу. В качестве такой задачи предлагается определение типа файла по его содержимому. Таким образом, перед началом работы, вам необходимо выбрать для себя один тип файла, который вы хотите распознать, заготовить чем больше тем лучше различных файлов выбранного типа и файлов типа, отличного от выбранного (например, если вы хотите распознавать, является файл файлом в формате «pdf» или нет, вам потребуется около 40 файлов в формате «pdf» и около 40 файлов не в формате «pdf»).

После выбора типа файла для распознавания можно приступать к программированию.

В рамках работы требуется создать один нейрон с 256 входами и обучить его.

Перед началом работы необходимо подготовить данные для обучения нейрона. Эти данные можно получить, обрабатывая заготовленные файлы и формируя после обработки каждого файла входной вектор. Вектор, полученный после обработки некоторого файла, должен содержать набор вероятностей появления в файле того или иного символа. Для вычисления этого вектора необходимо:

1. Посчитать, сколько раз в файле встречается символ с некоторым ASCII кодом (коды символов будут соответствовать индексу элементов массива с входными данными, а значение, расположенное по соответствующему индексу, будет соответствовать числу символов с данным кодом в данном файле)

2. Поделить значение каждого элемент массива на общее число символов файла.

В результате проделанных действий будет получен набор векторов, состоящих из 256 вещественных чисел. Векторы, полученные после обработки файлов выбранного типа, следует отличать от векторов, полученных после обработки файлов других типов, поскольку в результате поступления на вход файлов «pdf» (который мы вроде как хотим распознать) нейрон должен выдавать 1, а при поступлении других файлов -1.

После того, как входные векторы готовы, необходимо приступить к обучению нейрона. Обучение нейрона сводится к циклическому выполнению следующих действий:

1. Найти сумму произведений элементов входных векторов на элементы весового вектора:

2. Сравнить полученное значение с 0. Если значение больше 0, то подать на выход нейрона 1, иначе – подать -1.

3. Если на выходе нейрона получилось значение, которое должно получиться, то ничего не делать и перейти к рассмотрению следующего вектора.

4. Если на выходе нейрона получилось значение 1, а должно было получиться -1, то элементы вектора весовых коэффициентов (w) следует уменьшить(wi = wi - 2cxi)

5. Если на выходе нейрона получилось значение -1, а должно было получиться 1, то элементы вектора весовых коэффициентов (w) следует увеличить(wi = wi + 2cxi)

Подобную последовательность действий необходимо выполнять с данными ото всех входных файлов до тех пор, пока примерно в 90-95% случаев получаемый выход нейрона будет совпадать с желаемым. Значение «c» в приведённой выше формуле – коэффициент скорости обучения (Обычно его принимают за 0.2 или около того).

После того как сеть обучена, полученные коэффициенты следует сохранить и попробовать с их помощью определить тип 20-30 файлов, которые не участвовали в процессе обучения. Если всё хорошо, то примерно в 90% случаев результат определения будет правильный. Вот такая наука

Задание: распознавание гласных и согласных букв (чётных и нечётных цифр или чего-то подобного) с помощью простейшей искусственной нейронной сети

Если по каким-то причинам идея распознавать тип файла с помощью искусственной нейронной сети вам не нравится, её можно изменить на любую другую аналогичную задачу. Чтобы задача в итоге не оказалась слишком сложной для реализации, распознавание чего-либо следует выполнять для определения принадлежности объектов к одной из двух групп (на выходе нейрона будет значение 1 или -1). В качестве подобных задач, в частности, могут быть задачи разделения букв на гласные и согласные, цифр на чётные и нечётные. Интересной также может показаться задача разделения какого-то набора букв, например, на буквы, встречающиеся в вашем имени и буквы, которые в вашем имени не встречаются.

В каждом из случаев, перед реализацией следует задать себе вопрос о том, что будут представлять собой входные данные для обучения и сколько будет входов у нейрона. Символы для разделения можно задавать, например, в виде двумерных массивов, размерности, к примеру 8x8. Например, числа 0 и 1 можно задать следующими массивами из нулей и единиц:

- это цифра ноль!

- это цифра один!

Любые символы можно задать аналогичным образом, сделав знакоместо меньшего или большего размера. Каждый символ в этом случае следует записать в отдельный файл в виде последовательности нулей и единиц. При задании символов, обратите внимание, что они не всегда могут быть написаны идеально правильно. Отсутствие какого-то «пикселя» из 64 или присутствие неожиданного пикселя не должно (в идеале) влиять на результат распознавания.

2.3. Перечень вопросов для подготовки обучающихся к промежуточной аттестации

1. Определение и область применения искусственного интеллекта.
2. Представление задач: предметная область, сущности, отношения, суждения, языки представления знаний.
3. Методы решения задач: планирование в пространстве состояний и планирование в пространстве задач.
4. Поиск в пространстве состояний: граф пространства состояний, проблемные ситуации, разрешенные ходы, представление решения в пространстве состояний.
5. Слепые методы поиска в пространстве состояний, понятие комбинаторной сложности.
6. Эвристический поиск в пространстве состояний.
7. Метод редукции задач, и/или-графы.
8. Игры с полной информацией: представление в виде и/или-графа, позиции игрока, позиции противника.
9. Минимаксный принцип поиска в игровых задачах: основной вариант, статические и рабочие оценки.
10. Составление расписаний с использованием поиска в пространстве состояний.
11. Экспертные системы, системы, основанные на знаниях.
12. Базовые функции экспертных систем.
13. Символические вычисления: символы, синтаксические правила, правила трансформации, списки, точные пары.
14. Продукционные системы (системы основанные на знаниях): грамматика и архитектура продукционных систем.
15. Продукционные системы: недетерминированный набор правил, разрешение конфликтов, конфликтующее множество, метаправила.
16. Представление неопределенностей знаний и данных: коэффициенты уверенности.
17. Ассоциативные сети и сети фреймов: понятие прототипа, фрейма, значения по умолчанию и демоны, скептические и доверчивые системы.
18. Нейронные сети: определение, основные компоненты, основные характеристики, область применения.
19. Сети с управляемым обучением: описание нейронной сети, правила вычисления входного сигнала, функции активности, понятия обобщающей способности сети, обучающего и тестового множеств, эпохи.
20. Дельта-правило обучения нейронной сети (правило Видроу-Хоффа).
21. Линейные и нелинейные задачи, понятие линейно отделимых множеств.
22. Моделирование логических отношений с помощью нейронных сетей.
23. Алгоритм обратного распространения ошибки.
24. Самоорганизующиеся карты признаков: понятие кластера, прототипа кластера, свойства идеального алгоритма кластеризации.

3. Методические материалы, определяющие процедуру и критерии оценивания сформированности компетенций при проведении промежуточной аттестации

Критерии формирования оценок по ответам на вопросы, выполнению тестовых заданий

- оценка **«отлично»** выставляется обучающемуся, если количество правильных ответов на вопросы составляет 100 – 90% от общего объёма заданных вопросов;
- оценка **«хорошо»** выставляется обучающемуся, если количество правильных ответов на вопросы – 89 – 76% от общего объёма заданных вопросов;
- оценка **«удовлетворительно»** выставляется обучающемуся, если количество правильных ответов на тестовые вопросы –75–60 % от общего объёма заданных вопросов;
- оценка **«неудовлетворительно»** выставляется обучающемуся, если количество правильных ответов – менее 60% от общего объёма заданных вопросов.

Критерии формирования оценок по результатам выполнения заданий

«Отлично/зачтено» – ставится за работу, выполненную полностью без ошибок и недочетов.

«Хорошо/зачтено» – ставится за работу, выполненную полностью, но при наличии в ней не более одной негрубой ошибки и одного недочета, не более трех недочетов.

«Удовлетворительно/зачтено» – ставится за работу, если обучающийся правильно выполнил не менее 2/3 всей работы или допустил не более одной грубой ошибки и двух недочетов, не более одной грубой и одной негрубой ошибки, не более трех негрубых ошибок, одной негрубой ошибки и двух недочетов.

«Неудовлетворительно/не зачтено» – ставится за работу, если число ошибок и недочетов превысило норму для оценки «удовлетворительно» или правильно выполнено менее 2/3 всей работы.

Виды ошибок:

- *грубые ошибки: незнание основных понятий, правил, норм; незнание приемов решения задач; ошибки, показывающие неправильное понимание условия предложенного задания.*
- *негрубые ошибки: неточности формулировок, определений; нерациональный выбор хода решения.*
- *недочеты: нерациональные приемы выполнения задания; отдельные погрешности в формулировке выводов; небрежное выполнение задания.*

Критерии формирования оценок по зачету с оценкой

«Отлично/зачтено» – студент приобрел необходимые умения и навыки, продемонстрировал навык практического применения полученных знаний, не допустил логических и фактических ошибок

«Хорошо/зачтено» – студент приобрел необходимые умения и навыки, продемонстрировал навык практического применения полученных знаний; допустил незначительные ошибки и неточности.

«Удовлетворительно/зачтено» – студент допустил существенные ошибки.

«Неудовлетворительно/не зачтено» – студент демонстрирует фрагментарные знания изучаемого курса; отсутствуют необходимые умения и навыки, допущены грубые ошибки.

Критерии формирования оценок по экзамену

«**Отлично**» (5 баллов) – обучающийся демонстрирует знание всех разделов изучаемой дисциплины: содержание базовых понятий и фундаментальных проблем; умение излагать программный материал с демонстрацией конкретных примеров. Свободное владение материалом должно характеризоваться логической ясностью и четким видением путей применения полученных знаний в практической деятельности, умением связать материал с другими отраслями знания.

«**Хорошо**» (4 балла) – обучающийся демонстрирует знания всех разделов изучаемой дисциплины: содержание базовых понятий и фундаментальных проблем; приобрел необходимые умения и навыки, освоил вопросы практического применения полученных знаний, не допустил фактических ошибок при ответе, достаточно последовательно и логично излагает теоретический материал, допуская лишь незначительные нарушения последовательности изложения и некоторые неточности. Таким образом данная оценка выставляется за правильный, но недостаточно полный ответ.

«**Удовлетворительно**» (3 балла) – обучающийся демонстрирует знание основных разделов программы изучаемого курса: его базовых понятий и фундаментальных проблем. Однако знание основных проблем курса не подкрепляются конкретными практическими примерами, не полностью раскрыта сущность вопросов, ответ недостаточно логичен и не всегда последователен, допущены ошибки и неточности.

«**Неудовлетворительно**» (0 баллов) – выставляется в том случае, когда обучающийся демонстрирует фрагментарные знания основных разделов программы изучаемого курса: его базовых понятий и фундаментальных проблем. У экзаменуемого слабо выражена способность к самостоятельному аналитическому мышлению, имеются затруднения в изложении материала, отсутствуют необходимые умения и навыки, допущены грубые ошибки и незнание терминологии, отказ отвечать на дополнительные вопросы, знание которых необходимо для получения положительной оценки.

Экспертный лист
оценочных материалов для проведения промежуточной аттестации по
дисциплине «Системы искусственного интеллекта»

по направлению подготовки/специальности

09.03.01 «Информатика и вычислительная техника»
(код и наименование)

Направленность (профиль)/специализация

(наименование)

Бакалавар
квалификация выпускника

1. Формальное оценивание			
Показатели		Присутствуют	Отсутствуют
Наличие обязательных структурных элементов:		+	
–титульный лист		+	
–пояснительная записка		+	
– типовые оценочные материалы		+	
–методические материалы, определяющие процедуру и критерии оценивания		+	
Содержательное оценивание			
Показатели	Соответствует	Соответствует частично	Не соответствует
Соответствие требованиям ФГОС ВО к результатам освоения программы	+		
Соответствие требованиям ОПОП ВО к результатам освоения программы	+		
Ориентация на требования к трудовым функциям ПС (при наличии утвержденного ПС)	+		
Соответствует формируемым компетенциям, индикаторам достижения компетенций	+		

Заключение: ФОС рекомендуется/ не рекомендуется к внедрению; обеспечивает/ не обеспечивает объективность и достоверность результатов при проведении оценивания результатов обучения; критерии и показатели оценивания компетенций, шкалы оценивания обеспечивают/ не обеспечивают проведение всесторонней оценки результатов обучения.