МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. БАУМАНА

Факультет: Информатика и системы управления
Кафедра: Информационная безопасность (ИУ8)

ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Лабораторная работа №5 на тему:

«Исследование рекуррентной нейронной сети Хопфилда на примере задачи распознавания образов»

Вариант 4

Преподаватель:
Коннова Н.С.

Студент:
Куликова А.В.

Группа:
ИУ8-21М

Цель работы

Исследовать процедуры обучения и функционирования рекуррентной нейронной сети (РНС) Хопфилда в качестве устройства автоассоциированной памяти.

Постановка задачи

Номер варианта	Режим работы РНС Хопфилда	Запоминаемые образы
4	Асинхронный	3 4 5

Ход работы

Подготовленная картинка для генерации биполярного кода образа представлена на рисунках 1-3.

Рисунок 1 – Картинка Запоминаемые образы

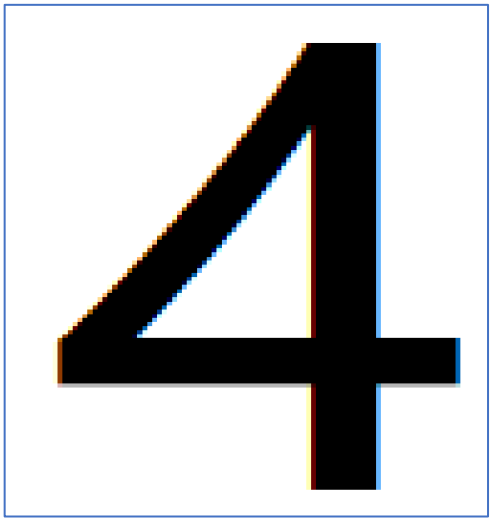


Рисунок 2 – Картинка Запоминаемые образы

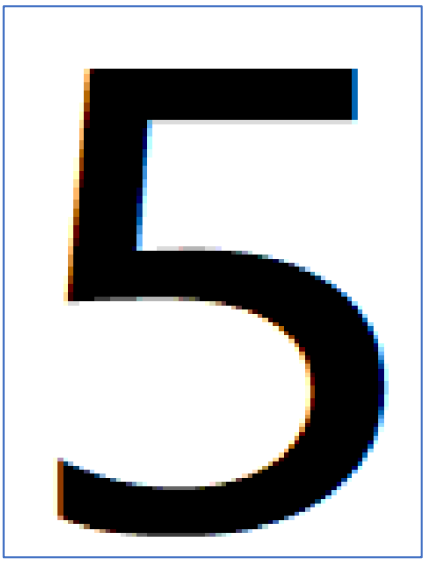


Рисунок 3 – Картинка Запоминаемые образы

Пусть требуется запомнить три образа, представляющих собой графическое изображение цифр «3», «4», «5», закодированных в биполярных матрицах-паттернов размерности 25×25:

Биполярный код образа "3" сгенерированного с рисунка 1:

[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1,
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, 1, 1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]

Биполярный код образа "4" сгенерированного с рисунка 2:

[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, 1, 1, 1, 1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, 1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
1, 1, 1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1,
-1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -
1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1,
1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, 1, 1,
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1,
1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1]

Биполярный код образа "5" сгенерированного с рисунка 3:

[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, -1, -1, -1, -

1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1,
-1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1]

Исходные образы согласно варианту задания приведены на рисунке 4

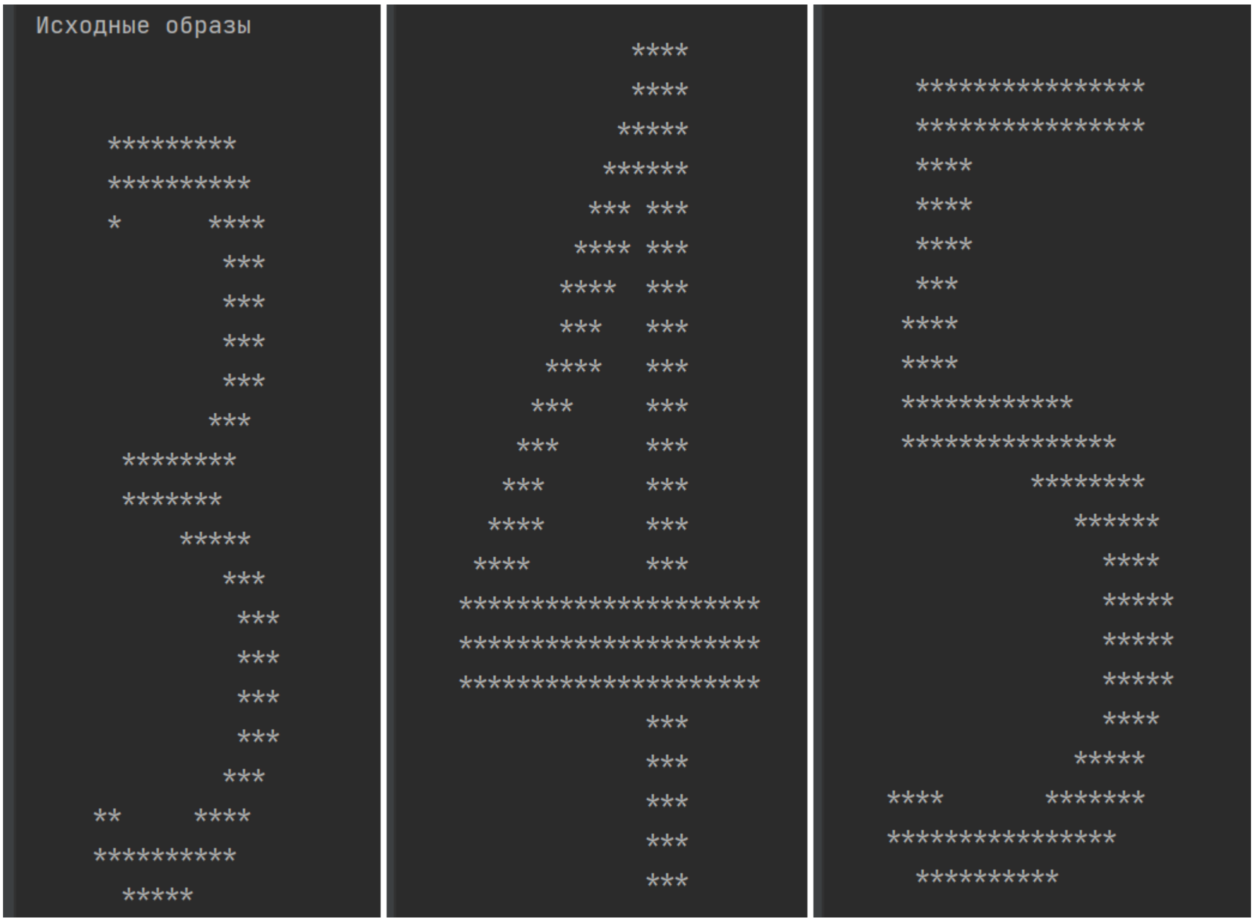


Рисунок 4 – Исходные образы

Фрагмент матрицы весов представлена на рисунке 5.

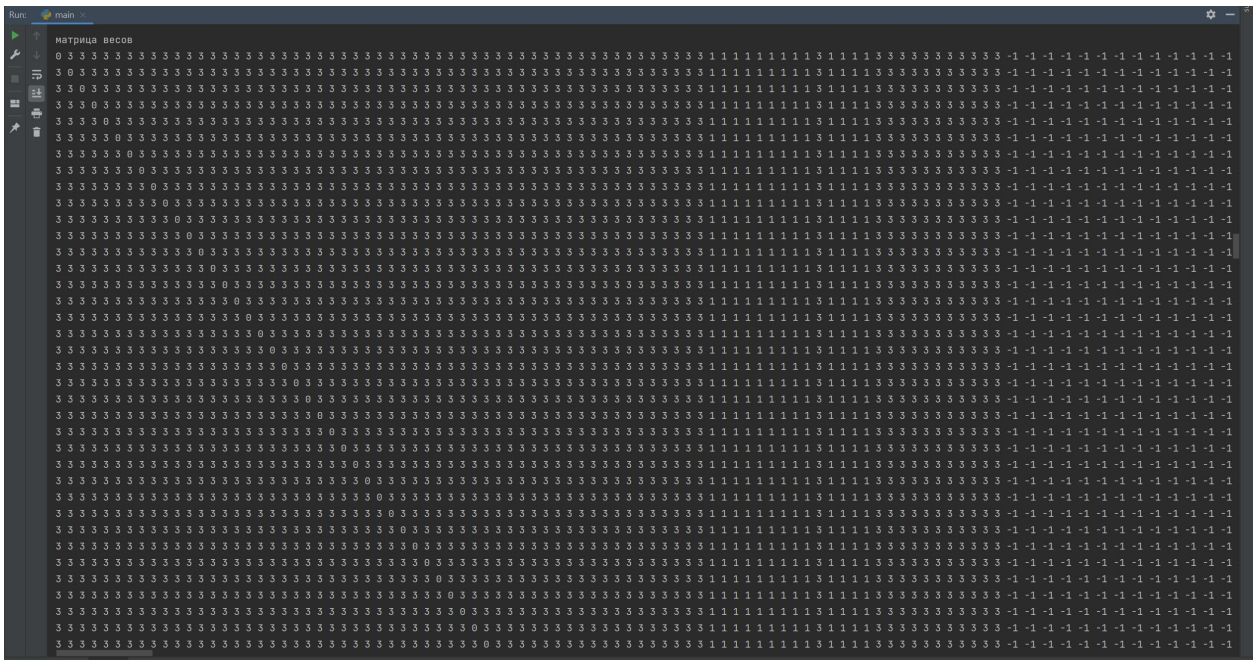


Рисунок 5 – Матрица весов (фрагмент)

Исказим все рабочие векторы и подадим их на вход РНС Хопфилда. В зависимости от поддаваемого изображения (толщина символов), РНС Хопфилда может не распознать (повышенный шум более 20-30%).

В данном случае символы на изображениях имеют тонкий шрифт из-за чего РНС Хопфилда успешно распознала в них эталонные объекты. Данный процесс распознавания представлен на рисунках 6-8.

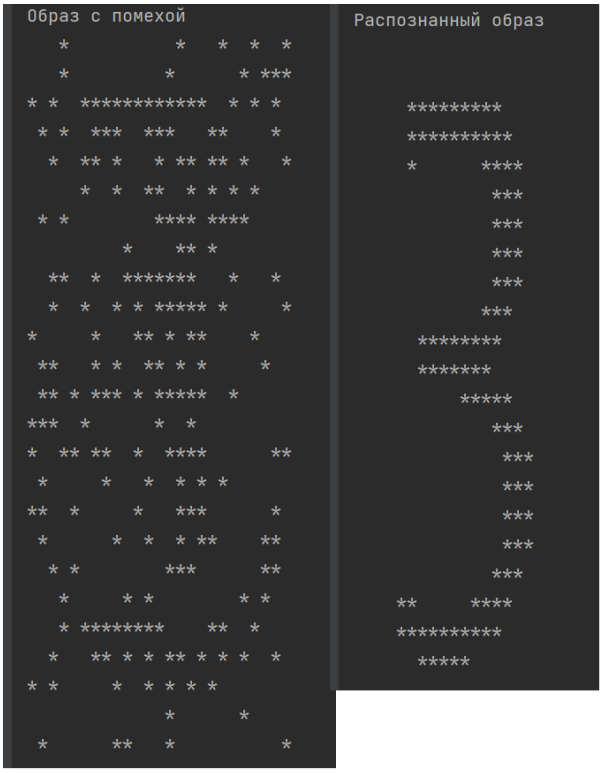


Рисунок 6 – Результат работы РНС Хопфилда. Образ «3» с помехой и распознанный

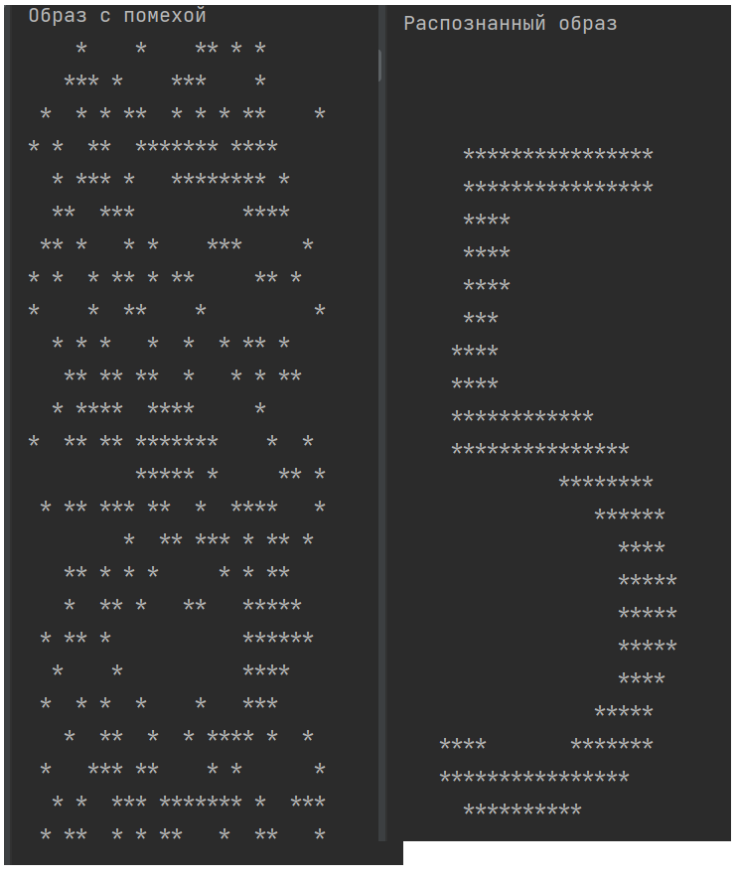


Рисунок 7 – Результат работы РНС Хопфилда. Образ «5» с помехой и распознанный

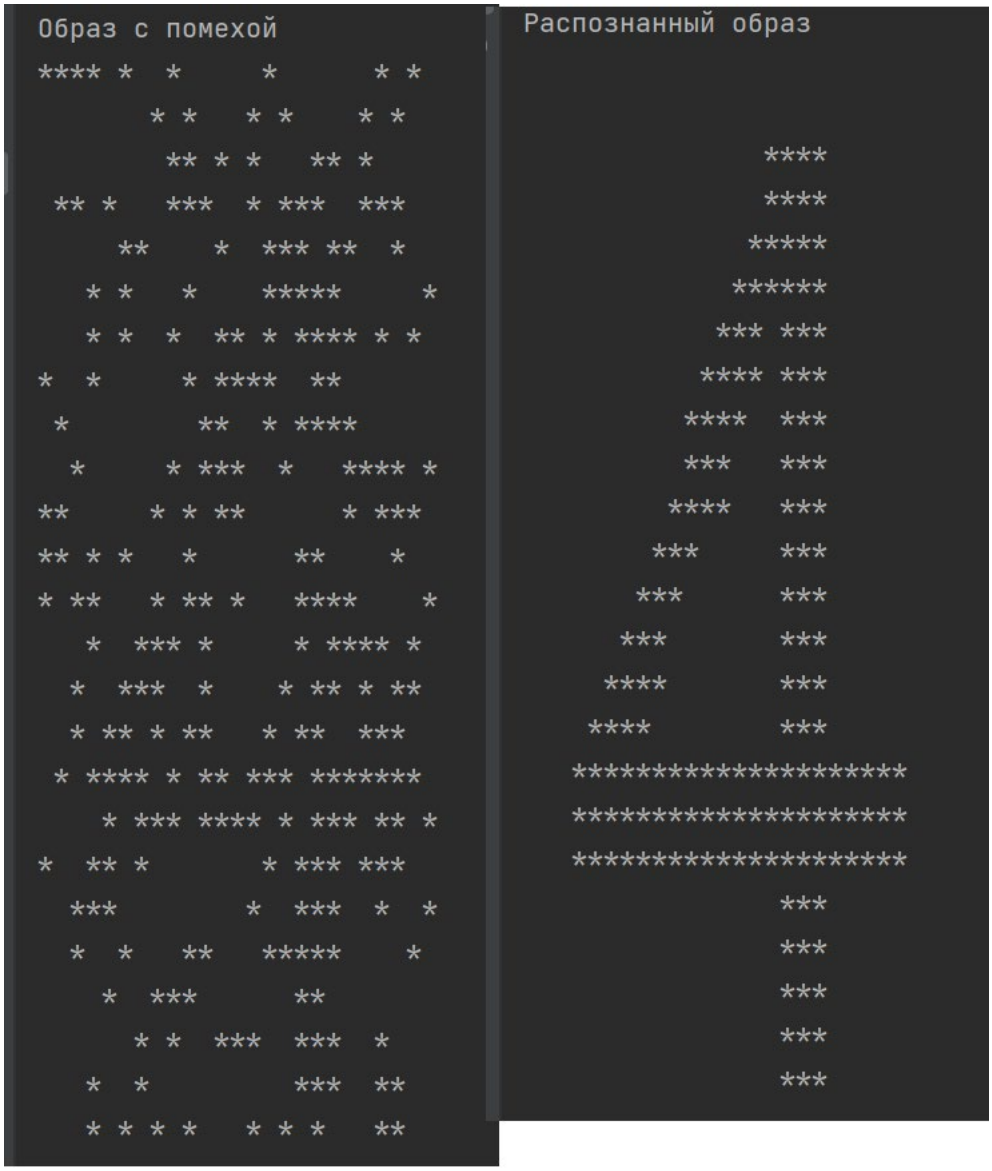


Рисунок 8 – Результат работы РНС Хопфилда. Образ «4» с помехой и распознанный

Затем подадим на вход РНС полностью зашумленный образец. В результате чего РНС Хопфилда распознает в нем образ «4». Данный процесс приведен на рисунке 9.

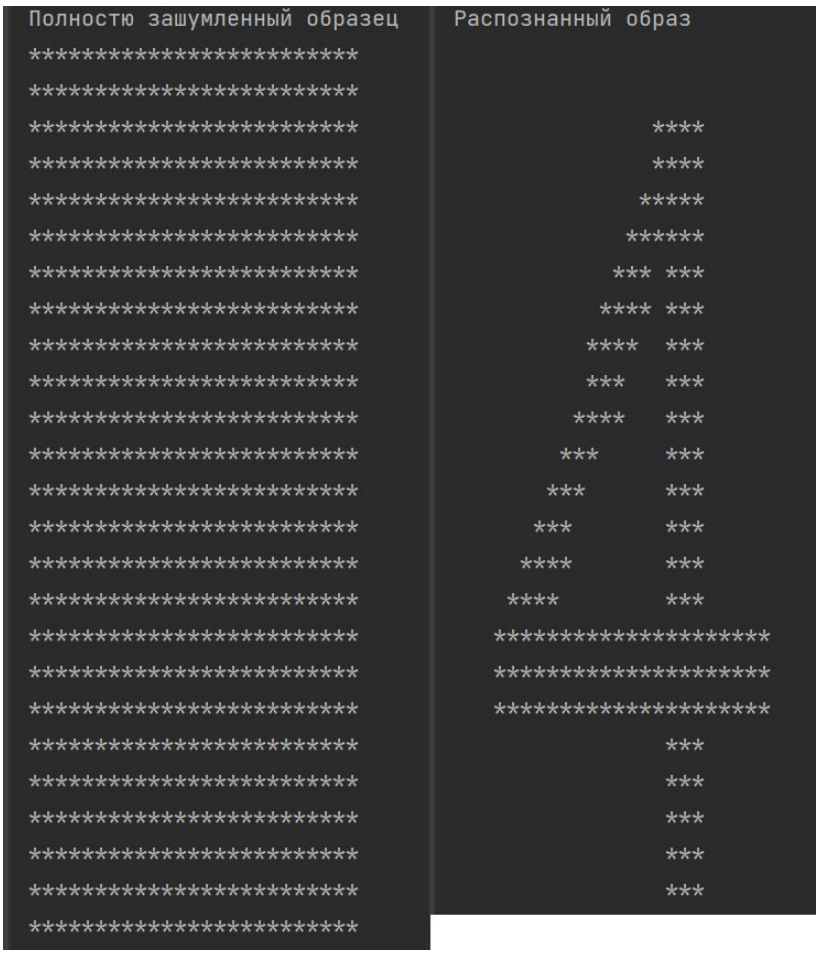


Рисунок 9 – Результат работы РНС Хопфилда. Полностью зашумленный образец и распознанный

Результат программы представлен в приложении А.

Исходный код программы представлен в приложении Б.

Выводы:

В процессе выполнения работы была изучена сеть Хопфилда, реализована в виде программы и протестирована на искаженных образах. По результатам которого протестирована возможность нейронной сети распознавать образы.

Было выяснено, что образа 25×25 достаточно чтобы исправить искаженный образ.

Так же была выявлена проблема с распознаванием мало различаемых образов, называемая химерой.

Также был простроен график зависимости ошибки от номера эпохи.

Контрольные вопросы

1. Как происходит обучение ассоциативного типа (без учителя)? Сформулируйте правило Хебба.

Обучение ассоциативного типа без учителя основано на правиле Хебба, которое формулируется следующим образом:

Правило Хебба гласит, что если два нейрона активируются одновременно, то сила связи между этими нейронами увеличивается. То есть, если нейрон A активируется и за ним следует активация нейрона B, то вес связи между нейронами A и B увеличивается. Это правило основано на идее, что нейроны, активирующиеся одновременно, имеют связь или ассоциацию.

2. Дайте определение ассоциативной памяти.

Ассоциативная память - это способность нейронных сетей или компьютерных систем запоминать и извлекать информацию, используя ассоциации и связи между элементами данных. В отличие от последовательной памяти, где данные хранятся и извлекаются по адресам, ассоциативная память позволяет осуществлять доступ к информации на основе сходства или связи между элементами.

В ассоциативной памяти информация хранится не в виде отдельных элементов, а в виде связей между этими элементами. При запросе на извлечение информации система ищет соответствия или ассоциации с входными данными, что позволяет эффективно находить и извлекать связанную информацию.

3. Расскажите о НС Хопфилда, алгоритмах ее обучения и функционировании.

Нейронная сеть Хопфилда - это один из видов рекуррентных нейронных сетей, предложенный американским физиком Джоном Хопфилдом в 1982 году. Основная идея НС Хопфилда заключается в использовании ассоциативной памяти для хранения и восстановления образов.

Функционирование нейронной сети Хопфилда основано на принципе ассоциативной памяти. Сеть состоит из узлов-нейронов, каждый из которых имеет два состояния: активное (1) и неактивное (0). Нейроны связаны между собой весами, которые определяют влияние одного нейрона на другой.

Обучение нейронной сети Хопфилда происходит путем предъявления ей образов, которые нужно запомнить. При этом веса связей между нейронами изменяются таким образом, чтобы сеть могла восстановить запомненные образы по частичным или зашумленным входным данным.

Алгоритм обучения нейронной сети Хопфилда основан на принципе хопфилдовского обучения, который заключается в коррекции весов связей между нейронами на основе предъявляемых образов. Этот процесс повторяется до тех пор, пока сеть не научится правильно восстанавливать запомненные образы.

После обучения нейронная сеть Хопфилда способна восстанавливать запомненные образы по частичным или зашумленным входным данным, используя ассоциативную память и веса связей между нейронами. НС Хопфилда может применяться для решения задач распознавания образов, хранения и восстановления информации, а также для моделирования различных процессов, связанных с ассоциативной памятью.

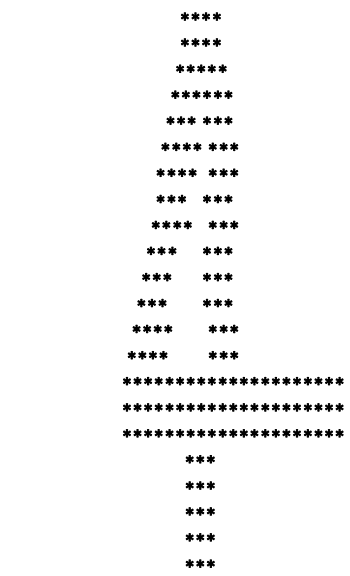
ПРИЛОЖЕНИЕ А

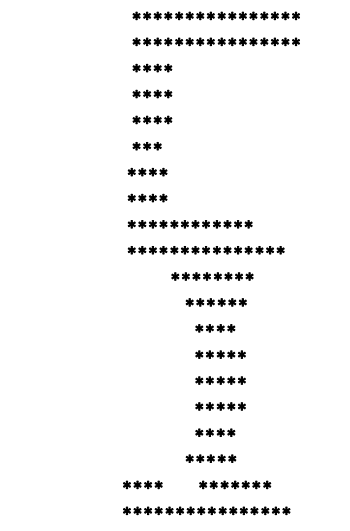
матрица "3"
[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1,
1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, 1, 1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]

матрица "4"
[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1,
1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1,
1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1,
1, 1, 1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
1, 1, 1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
1, 1, 1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
1, 1, 1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]

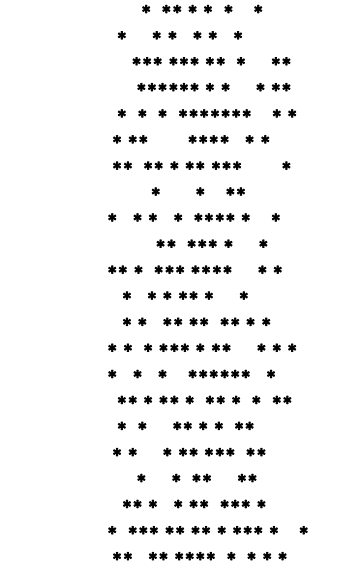
матрица "5"
[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1,
-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -
1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
-1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
Исходные образы

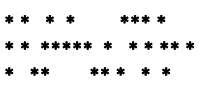
```
*********
**********
*    ****
    ***
    ***
    ***
    ***
   ***
 ********
 *******
   *****
    ***
     ***
     ***
     ***
     ***
    ***
**   ****
**********
 *****
```



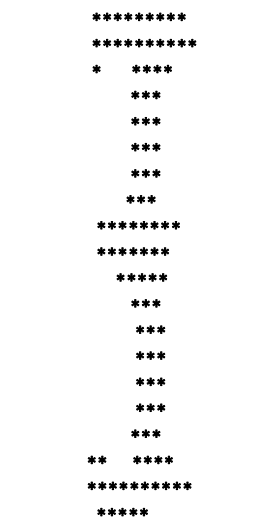


Образ с помехой

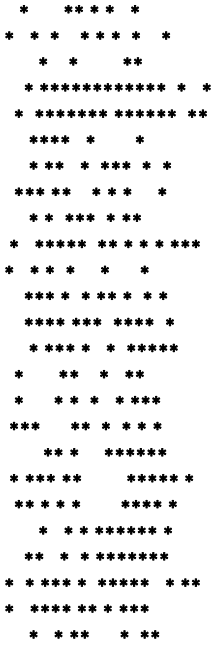




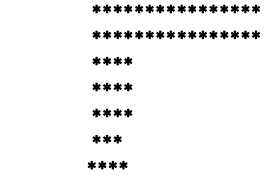
Распознанный образ

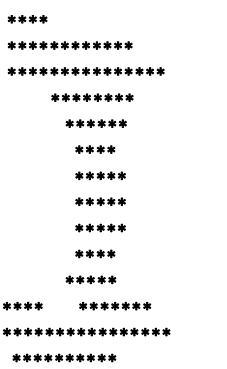


Образ с помехой

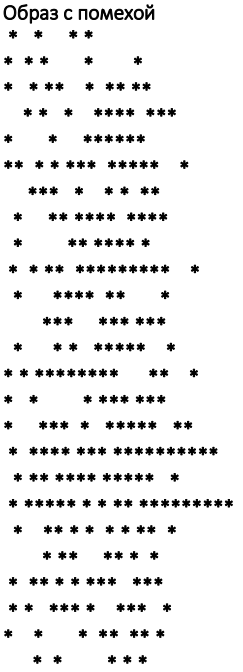


Распознанный образ

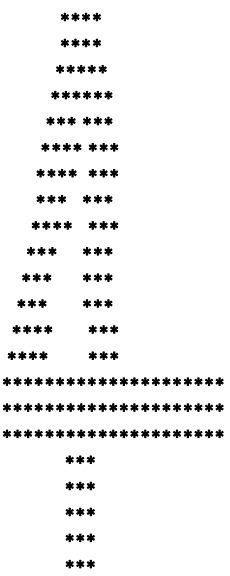




Образ с помехой



Распознанный образ



Полностью зашумленный образец

```
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
*************************
```

Распознанный образ

```
       ****
       ****
      *****
      ******
     *** ***
     **** ***
    ****  ***
    ***  ***
    ****  ***
   ***   ***
   ***    ***
  ***     ***
  ****    ***
 ****     ***
*********************
*********************
*********************
       ***
       ***
       ***
       ***
       ***
```

матрица весов
0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 3 1 1 1 1 3 3 3 3 3 3 3 3 3 3
3 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 -1 1 1 1 1 1 1 -1 -1 -3 -3 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 -1 -1 -1 1 1 1 1 3 3 3 3
3 3 3 3 3 3 3 1 1 1 1 3 3 3 1 -1 -1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 1 1 -1 -1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 3 3 1 1 1 -1 1 1 1 1 1 1 3 3 3 3 3 3
3 3 3 3 1 1 1 1 3 3 1 1 -1 1 1 1 3 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 -1 -1 1 -1 -1 -1 -1 1 3 3 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 -1 -1 -3 -3 -3 -1 -1 1 1 1 1 1 1 1 3 3 3 3 3
3 3 3 3 3 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 -1 -1 -1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 3 3 3 3 -1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 3 1 1 -3 -1 -1 1 1
1 1 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 3 1 1 -1 1 -1 1 1 1 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 -1 -1 -1 -1 -1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 -1
-1 -1 -1 -1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 1 -1 -1 -1 -1 -1 1 3 3 3 3 3 1 1 3 3 3 3 3 1 1 1 1 3 1 1 -1 1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 3 3 -1 -
1 -1 1 1 3 3 3 3 3 3 3 1 1 1 -1 1 1 1 1 3 3 3 1 1 -1 -1 -1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 3 1 1
1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 3 1 1 1 1 3 3 3 3 3 3 3 3 3 3
3 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 -1 1 1 1 1 1 1 -1 -1 -3 -3 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 -1 -1 -1 1 1 1 1 3 3 3 3
3 3 3 3 3 3 3 1 1 1 1 3 3 3 1 -1 -1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 1 1 -1 -1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 3 3 1 1 1 -1 1 1 1 1 1 1 3 3 3 3 3 3
3 3 3 3 1 1 1 1 3 3 1 1 -1 1 1 1 3 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 -1 -1 1 -1 -1 -1 -1 1 3 3 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 -1 -1 -3 -3 -3 -1 -1 1 1 1 1 1 1 1 3 3 3 3 3
3 3 3 3 3 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 -1 -1 -1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 3 3 3 3 -1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 3 1 1 -3 -1 -1 1 1
1 1 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 3 1 1 -1 1 -1 1 1 1 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 -1 -1 -1 -1 -1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 -1
-1 -1 -1 -1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 1 -1 -1 -1 -1 -1 1 3 3 3 3 3 1 1 3 3 3 3 3 1 1 1 1 3 1 1 -1 1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 3 3 -1 -
1 -1 1 1 3 3 3 3 3 3 3 1 1 1 -1 1 1 1 1 3 3 3 1 1 -1 -1 -1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 3 1 1
1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

3 3 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 3 1 1 1 1 3 3 3 3 3 3 3 3 3 3
3 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 -1 1 1 1 1 1 1 -1 -1 -3 -3 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 -1 -1 -1 1 1 1 1 3 3 3 3
3 3 3 3 3 3 3 1 1 1 1 3 3 3 1 -1 -1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 1 1 -1 -1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 3 3 1 1 1 -1 1 1 1 1 1 1 3 3 3 3 3 3
3 3 3 3 1 1 1 1 3 3 1 1 -1 1 1 1 3 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 -1 -1 1 -1 -1 -1 -1 1 1 3 3 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 -1 -1 -3 -3 -3 -1 -1 1 1 1 1 1 1 1 3 3 3 3 3
3 3 3 3 3 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 -1 -1 -1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 3 3 3 3 -1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 1 1 -3 -1 -1 1 1
1 1 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 3 1 1 -1 1 -1 1 1 1 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 -1 -1 -1 -1 -1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 1 -1
-1 -1 -1 -1 1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 1 1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 1 1 3 3 3 3 3 1 1 1 1 3 1 1 -1 1 1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 3 3 -1 -
1 -1 1 1 1 3 3 3 3 3 3 3 1 1 1 -1 1 1 1 1 1 3 3 3 1 1 -1 -1 -1 1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 3 1 1
1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 3 1 1 1 1 3 3 3 3 3 3 3 3 3 3
3 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 -1 1 1 1 1 1 1 -1 -1 -3 -3 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 -1 -1 -1 1 1 1 1 3 3 3 3
3 3 3 3 3 3 3 1 1 1 1 3 3 3 1 -1 -1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 1 1 -1 -1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 3 3 1 1 1 -1 1 1 1 1 1 1 3 3 3 3 3 3
3 3 3 3 1 1 1 1 3 3 1 1 -1 1 1 1 3 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 -1 -1 1 -1 -1 -1 -1 1 1 3 3 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 -1 -1 -3 -3 -3 -1 -1 1 1 1 1 1 1 1 3 3 3 3 3
3 3 3 3 3 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 -1 -1 -1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 3 3 3 3 -1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 1 1 -3 -1 -1 1 1
1 1 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 3 1 1 -1 1 -1 1 1 1 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 -1 -1 -1 -1 -1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 1 -1
-1 -1 -1 -1 1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 1 1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 1 1 3 3 3 3 3 1 1 1 1 3 1 1 -1 1 1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 3 3 -1 -
1 -1 1 1 1 3 3 3 3 3 3 3 1 1 1 -1 1 1 1 1 1 3 3 3 1 1 -1 -1 -1 1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 3 1 1
1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 3 1 1 1 1 3 3 3 3 3 3 3 3 3 3
3 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 -1 1 1 1 1 1 1 -1 -1 -3 -3 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 -1 -1 -1 1 1 1 1 3 3 3 3
3 3 3 3 3 3 3 1 1 1 1 3 3 3 1 -1 -1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 1 1 -1 -1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 3 3 1 1 1 -1 1 1 1 1 1 1 3 3 3 3 3 3
3 3 3 3 1 1 1 1 3 3 1 1 -1 1 1 1 3 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 -1 -1 1 -1 -1 -1 -1 1 1 3 3 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 -1 -1 -3 -3 -3 -1 -1 1 1 1 1 1 1 1 3 3 3 3 3
3 3 3 3 3 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 -1 -1 -1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 3 3 3 3 -1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 1 1 -3 -1 -1 1 1
1 1 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 3 1 1 -1 1 -1 1 1 1 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 -1 -1 -1 -1 -1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 1 -1
-1 -1 -1 -1 1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 1 1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 1 1 3 3 3 3 3 1 1 1 1 3 1 1 -1 1 1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 3 3 -1 -
1 -1 1 1 1 3 3 3 3 3 3 3 1 1 1 -1 1 1 1 1 1 3 3 3 1 1 -1 -1 -1 1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 3 1 1
1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 3 1 1 1 1 3 3 3 3 3 3 3 3 3 3
3 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 -1 1 1 1 1 1 1 -1 -1 -3 -3 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 -1 -1 -1 1 1 1 1 3 3 3 3
3 3 3 3 3 3 3 1 1 1 1 3 3 3 1 -1 -1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 1 1 -1 -1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 3 3 1 1 1 -1 1 1 1 1 1 1 3 3 3 3 3 3
3 3 3 3 1 1 1 1 3 3 1 1 -1 1 1 1 3 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 -1 -1 1 -1 -1 -1 -1 1 1 3 3 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 -1 -1 -3 -3 -3 -1 -1 1 1 1 1 1 1 1 3 3 3 3 3
3 3 3 3 3 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 -1 -1 -1 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 3 3 3 3 -1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 1 1 -3 -1 -1 1 1
1 1 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 3 3 1 1 -1 1 -1 1 1 1 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 -1 -1 -1 -1 -1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 1 -1
-1 -1 -1 -1 1 1 3 3 3 3 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 1 1 -1 -1 -1 -1 -1 1 1 3 3 3 3 3 1 1 3 3 3 3 3 1 1 1 1 3 1 1 -1 1 1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 3 3 -1 -
1 -1 1 1 1 3 3 3 3 3 3 3 1 1 1 -1 1 1 1 1 1 3 3 3 1 1 -1 -1 -1 1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 3 1 1
1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

… (ТУТ МНОГО СТРАНИЦ БЫЛО БЫ АЖ ДО 100+)

Process finished with exit code 0

ПРИЛОЖЕНИЕ Б

```
# БАЗА
# matrix = [
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
#     -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1]

from PIL import Image
def image_to_matrix(image_path):
     # Открываем изображение
    image = Image.open(image_path)
    # Преобразуем изображение в оттенки серого
    image = image.convert('L')
    # Изменяем размер изображения до 25x25 с интерполяцией ближайшего соседа
    image = image.resize((25, 25), Image.NEAREST)
    matrix = []
    for y in range(25):
        for x in range(25):
            # Получаем значение пикселя изображения
            pixel_value = image.getpixel((x, y))
            # Пороговое значение для определения закрытых символов
            if pixel_value < 128:
                matrix.append(1)
            else:
                matrix.append(-1)
    return matrix
```

```
matrix_3 = image_to_matrix("OBRAZ_3.png")
matrix_4 = image_to_matrix("OBRAZ_4.png")
matrix_5 = image_to_matrix("OBRAZ_5.png")

print("матрица \"3\"")
print(matrix_3)
print("\nматрица \"4\"")
print(matrix_4)
print("\nматрица \"5\"")
print(matrix_5)

import numpy as np
import random

def print_array(vector,vector_n=25,vector_m=25):
    for index_i in range(vector_n):
        for index_j in range(vector_m):
            if vector[index_i*array_m+index_j]==1:
                print("*", end = "")
            else:
                print(" ", end = "")
        print()

def interference_array(vector,vector_n=25,vector_m=25):
    for index_i in range(vector_n):
        for index_j in range(vector_m):
            if random.randint(0, 9)%4==1:
                if vector[index_i*array_m+index_j]==1:
                    vector[index_i*array_m+index_j]=-1
                else:
                    vector[index_i*array_m+index_j]=1

def function_activate(net, pred):
    # Если net больше нуля, возвращаем 1
    if (net > 0):
        return 1
    # Если net меньше нуля, возвращаем -1
    elif (net < 0):
        return -1
    # Если net равен нулю, возвращаем предыдущее значение pred
    else:
        return pred

def learning_verification(current_signal, previous_signal):
    # Проходим по всем элементам входных сигналов
    for index_i in range(len(current_signal)):
        # Если текущий элемент не равен соответствующему элементу в предыдущем
сигнале, возвращаем 0
        if current_signal[index_i] != previous_signal[index_i]:
            return 0
    # Если все элементы совпадают, возвращаем 1
```

```
    return 1

def calculation_net(matrix_weight, signal):
    # Получаем размер сигнала
    signal_size = len(matrix_weight[0])
    # Создаем текущий сигнал из входного сигнала
    current_signal = [signal[i] for i in range(signal_size)]
    # Создаем предыдущий сигнал, заполняя его -1
    previous_signal = [-1 for i in range(signal_size)]
    era = 0
    network_output = [0 for i in range(signal_size)]
    # Пока не завершено обучение
    while not learning_verification(current_signal, previous_signal):
        # Обнуляем сетевой выход и обновляем предыдущий сигнал
        for index_i in range(signal_size):
            network_output[index_i] = 0
            previous_signal[index_i] = current_signal[index_i]
        # Рассчитываем выходной сигнал для каждого элемента
        for index_k in range(signal_size):
            for index_j in range(index_k):
                network_output[index_k] += matrix_weight[index_k][index_j] *
current_signal[index_j]
            for index_j in range(index_k + 1, signal_size):
                network_output[index_k] += matrix_weight[index_k][index_j] *
previous_signal[index_j]
            current_signal[index_k] = function_activate(network_output[index_k],
previous_signal[index_i])
        era += 1
    # Формируем результат
    result = [0 for _ in range(signal_size)]
    for i in range(signal_size):
        result[i] = current_signal[i]
    return result

array_n=25
array_m=25

vector_3 = matrix_3
vector_4 = matrix_4
vector_5 = matrix_5

print("Исходные образы")
print_array(vector_3)
print_array(vector_4)
print_array(vector_5)

# Создаем матрицу matrix_weight размером len(vector_3) x len(vector_3) и
заполняем нулями
matrix_weight = np.array([[0 for index_i in range(len(vector_3))] for index_j in
range(len(vector_3))], int)
# Заполняем матрицу matrix_weight значениями
```

```
for index_i in range(len(vector_3)):
    for index_j in range(len(vector_3)):
        # Рассчитываем значение для каждой ячейки матрицы
        matrix_weight[index_i][index_j] = vector_5[index_i] * vector_5[index_j] +
vector_3[index_i] * vector_3[index_j] + vector_4[index_i] * vector_4[index_j]
        # Если индексы совпадают, устанавливаем значение ячейки в 0
        if index_i == index_j:
            matrix_weight[index_i][index_j] = 0

interference_array(vector_3)
print()
print()
print("Образ с помехой")
print_array(vector_3)
print()
print()
print()
print()
interference_array(vector_5)
interference_array(vector_4)
result=[0 for i in range(len (matrix_weight))]
result=calculation_net(matrix_weight,vector_3)
print("Распознанный образ")
print_array(result)
print()
print()

print("Образ с помехой")
print_array(vector_5)
result=calculation_net(matrix_weight,vector_5)
print()
print()
print("Распознанный образ")
print_array(result)
print()
print()

print("Образ с помехой")
print_array(vector_4)
result=calculation_net(matrix_weight,vector_4)
print()
print()
print("Распознанный образ")
print_array(result)

vector_poln=[]
for index_i in range(len(vector_3)):
    vector_poln.append(1)

print()
print()
```

```
print("Полностю зашумленный образец")
print_array(vector_poln)
print()
print()
print("Распознанный образ")
print_array(result)
print()

print('матрица весов')
for index_i in range(len(vector_3)):
        for index_j in range(len(vector_3)):
            print(matrix_weight[index_i][index_j],end=' ')
        print()
```