**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по учебной практике**

**Тема: Генетические алгоритмы**

|  |  |  |
| --- | --- | --- |
| Студент гр. 2304 |  | Деменев К.О. |
| Студентка гр. 2304 |  | Иванова М.А. |
| Студент гр. 2304 |  | Шумилов А.В. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2024

## Цель работы.

Изучить генетические алгоритмы, научиться применять их на практике. Разработать генетический алгоритм, метрику его качества, прототип GUI, а также способ представления данных. Создать прототип программы, решающей задачу о неограниченном рюкзаке с помощью генетического алгоритма.

## Задание. Вариант 1.

Задача о рюкзаке (1 рюкзак)

Дано N вещей, каждая i-я имеет вес Wi и стоимость Ci. Необходимо заполнить рюкзак с максимальной вместимостью по весу Wmax вещами так, чтобы суммарная стоимость вещей в рюкзаке была максимальной. Можно класть несколько копий одной вещи в рюкзак.

**Распределение ролей в команде.**

* Деменев К.О. – разработка и реализация GUI;
* Иванова М.А. – написание отчета, частичная реализация алгоритма;
* Шумилов А.В. – организация работы в команде, разработка структуры проекта, частичная реализация алгоритма.

Генетический алгоритм был разработан совместно.

## Выполнение работы.

Генетический алгоритм

В ходе решения задачи о неограниченном рюкзаке с помощью генетического алгоритма был разработан следующий способ представления данных.

*Геном* - упорядоченный набор чисел, больше или равных нуля, - представляется в виде целочисленного массива. Длина генома соответствует количеству введённых пользователем вещей для заполнения рюкзака. Порядок ввода предметов пользователем запоминается и фиксируется: значение по индексу в массиве генома определяет количество экземпляров определенной вещи.

*Функция приспособленности (метрика качества)* текущего потомка равняется либо суммарной стоимостей вещей, лежащих в рюкзаке, если их суммарный вес не превышает максимально допустимого; либо нулю в противном случае.

Сам *генетический алгоритм* выглядит следующим образом:

1. Случайная генерация начальной популяции из N хромосом;
2. Вычисление функции приспособленности каждой особи текущего поколения;
3. С помощью турнирного отбора, численностью 2, выбирается N родителей для следующего поколения;
4. Внутри отобранного промежуточного поколения родителей произвольным образом выбирается пара особей для скрещивания;
5. Производится равномерное скрещивание родителей с вероятностью Pc, получается сразу 2 потомка;
6. Шаги 4-5 повторяются до тех пор, пока не получится поколение детей размера N;
7. Производится мутация потомков с вероятностью Pm: каждый ген внутри генома мутирует с вероятностью Pg по следующей формуле: новая переменная = старая переменная , знак + или – выбирается с равной вероятностью;
8. Производится элитарный отбор в следующее поколение: берется 10% лучших представителей предыдущего поколения, а остальные 90% выбираются случайно;
9. Шаги 2-8 повторяются до тех пор, пока на свет не будет произведено заданное число поколений M.

Количество особей в поколении (N), количество поколений (M), вероятности скрещивания (Pc), мутации всего генома (Pm), отдельного гена (Pg), а также параметр – гиперпараметры, подбираемые вручную для лучшей сходимости алгоритма.

Организация кода

В ходе работы была разработана следующая структура проекта:

* Класс Item отвечает за предмет, помещаемым в рюкзак. Поля класса - цена и вес предмета.
* Класс Backpack представляет собой особь поколения, т.е. одно из возможных решений задачи. Поля класса - геном, представляющий решение задачи; вес и стоимость рюкзака.
* Класс Generation представляет собой поколение. Поле класса - массив особей вида Backpack.
* Класс GeneticAlgorithm является реализацией генетического алгоритма. Поля класса – массив вещей, введенных пользователем; длина генома и максимальный допустимый вес рюкзака. Методы – отбор родителей и особей в следующее поколение, а также скрещивание и мутация.
* Класс Application является реализацией всего приложения: имеет интерфейсы ввода и вывода данных, реализует связь между графическим интерфейсом и реализацией алгоритма GeneticAlgorithm.

Реализация графического интерфейса

Для написания GUI была использована библиотека PyQt6.

Каждое окно организовано как отдельный класс: InputWindow – окно ввода вручную, IterationWindow – окно работы алгоритма, AlgParamsWindow – окно параметров алгоритма.

Нажатие каждой из кнопок воспринимается как отдельное событие, которое обрабатывается в соответствии с назначением кнопки.

При запуске программы отображается начальное окно (рис. 1). При выборе параметров работы алгоритма (рис. 2) или одного из способов ввода данных, указанных в начальном окне: «загрузить данные из файла» и «ввод данных вручную» (рис. 3, 4) - открываются соответствующие окна.

При работе алгоритма отображается окно, содержащее информацию о текущем шаге алгоритма, текущем поколении и график развития поколений (рис. 5).

Итоговое окно работы алгоритма имеет формат, сходный с окном отображения текущего шага и содержит кнопки перезапуска алгоритма с новыми параметрами и закрытия программы. При закрытии программы отображается окно подтверждения выхода (рис. 6).

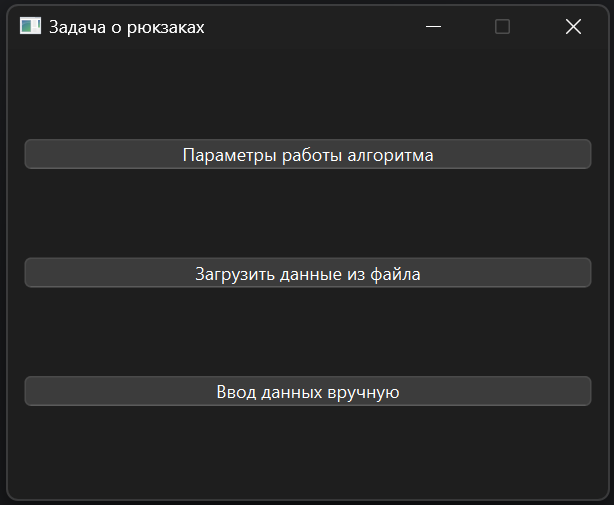


Рисунок 1 - Окно при запуске программы

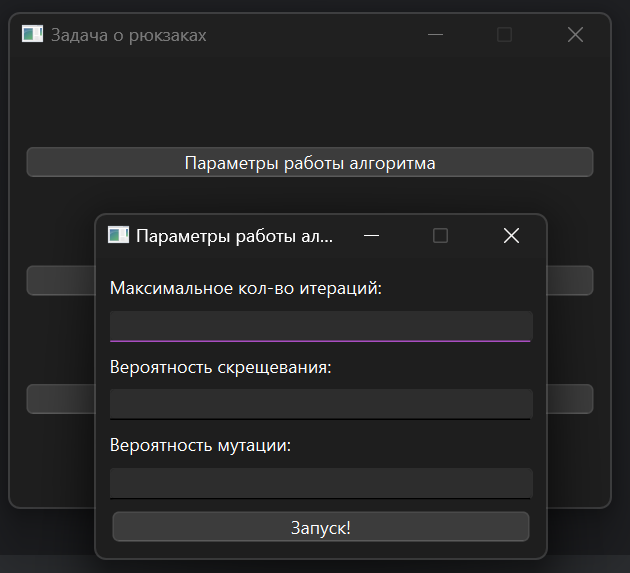


Рисунок 2 - Окно при выборе параметров работы алгоритма

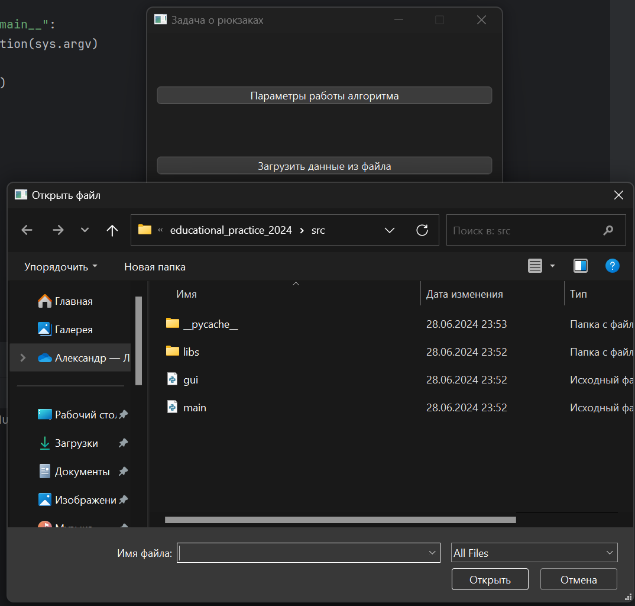


Рисунок 3 - Окно при выборе ввода данных из файла

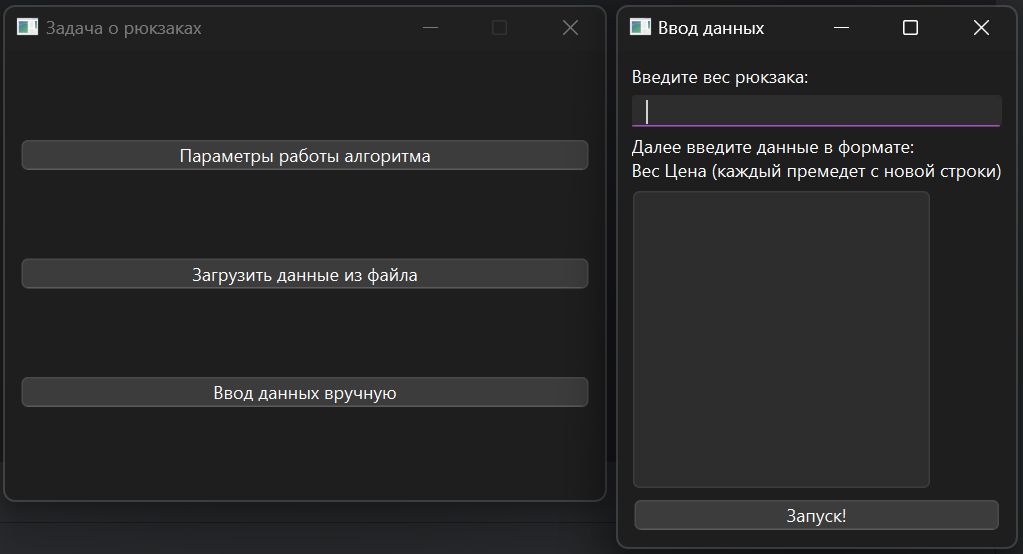


Рисунок 4 - Окно при выборе ввода данных вручную

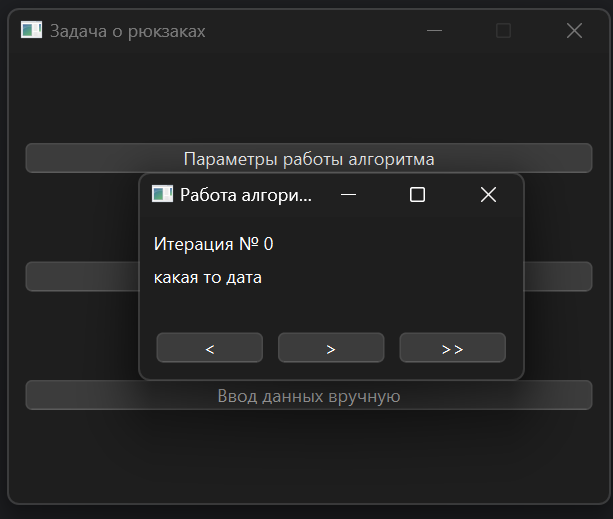


Рисунок 5 - Окно работы алгоритма

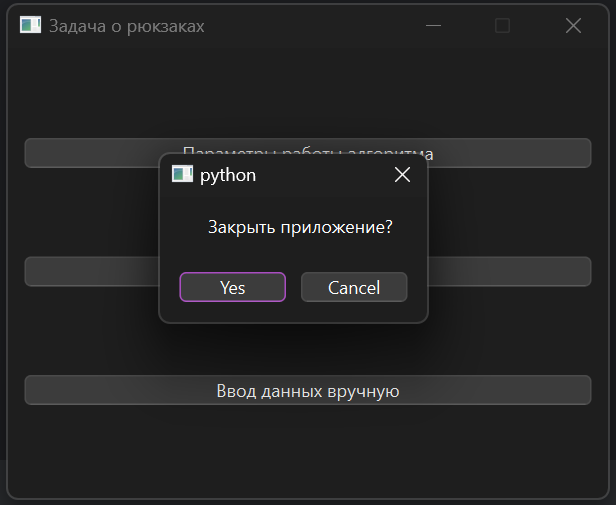


Рисунок 6 - Окно подтверждения выхода из программы

**Вывод.**

Изучены генетические алгоритмы. Разработан генетический алгоритм, метрика его качества, прототип GUI, а также способ представления данных. Создан прототип программы, решающую задачу о неограниченном рюкзаке с помощью генетического алгоритма.

# Приложение А Исходный код программы

Название файла: gui.py

from PyQt6.QtCore import Qt  
from PyQt6.QtGui import QPixmap, QCloseEvent  
from PyQt6.QtWidgets import QApplication, QWidget, QMainWindow, QPushButton, QVBoxLayout, QLineEdit, QLabel, QDialog, \  
 QFileDialog, QPlainTextEdit, QHBoxLayout, QMessageBox  
  
  
class Window(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_() # наследуем от QMainWindow  
  
 self.inputFileName = '~/'  
  
 # Окно ввода вручную  
 self.inputWindow = InputWindow()  
 # Окно работы алгоритма  
 self.iterationWindow = IterationWindow()  
 # Окно параметров алгоритма  
 self.algParamsWindow = AlgParamsWindow()  
  
 # Название окна  
 self.setWindowTitle("Задача о рюкзаках")  
 # Размер окна  
 self.setFixedSize(400, 300)  
  
 # Вертикальная разметка виджетов  
 layout = QVBoxLayout()  
 # Массив виджетов (все виджеты, которые будут на этом окне)  
 main\_window\_widgets = []  
  
 # Создаем кнопку ввода параметров алгоритма  
 # Добавляем в виджеты  
 # Привязываем к событию clicked ивент paramsButtonEvent  
 params\_button = QPushButton("Параметры работы алгоритма")  
 main\_window\_widgets.append(params\_button)  
 params\_button.clicked.connect(self.paramsButtonEvent)  
  
 # Создаем кнопку закгрузки данных из файла  
 # Добавляем в виджеты  
 # Привязываем к событию clicked ивент browseEvent  
 browse\_file\_button = QPushButton("Загрузить данные из файла")  
 browse\_file\_button.clicked.connect(self.browseEvent)  
 main\_window\_widgets.append(browse\_file\_button)  
  
 # Создаем кнопку ввода вручную  
 # Добавляем в виджеты  
 # Привязываем к событию clicked ивент paramsButtonEvent  
 input\_button = QPushButton("Ввод данных вручную")  
 input\_button.clicked.connect(self.InputButtonEvent)  
 main\_window\_widgets.append(input\_button)  
  
 # Добавляем в разметку все наши элементы  
 for w in main\_window\_widgets:  
 layout.addWidget(w)  
  
 # Превращаем нашу разметку в один большой виджет  
 widget = QWidget()  
 widget.setLayout(layout)  
  
 # Устанавливаем центральный виджет окна. Виджет будет расширяться по умолчанию,  
 # заполняя всё пространство окна.  
 self.setCentralWidget(widget)  
  
 # Обработка сигнала clicked для кнопки 1  
 def paramsButtonEvent(self):  
 self.algParamsWindow.show()  
  
 # Обработка сигнала clicked для кнопки 2  
 # Открываем диалог (выбор файла)  
 def browseEvent(self):  
 file\_name = QFileDialog.getOpenFileName(self, 'Открыть файл', '~/')  
 self.inputFileName = file\_name[0]  
 print(self.inputFileName)  
  
 self.iterationWindow.show()  
  
 # Обработка сигнала clicked для кнопки 3  
 def InputButtonEvent(self):  
 self.inputWindow.show()  
  
 # Ивент запуска окна с алгоритмом  
 def startAlgorithm(self):  
 self.iterationWindow.show()  
  
 # Ивент закрытия главного окна  
 def closeEvent(self, event: QCloseEvent) -> None:  
 closing\_MB = QMessageBox(self)  
 closing\_MB.setText("Закрыть приложение?")  
 closing\_MB.setStandardButtons(QMessageBox.standardButtons(closing\_MB).Yes |  
 QMessageBox.standardButtons(closing\_MB).Cancel)  
 closed = closing\_MB.exec()  
  
 if closed == QMessageBox.standardButtons(closing\_MB).Yes:  
 event.accept()  
 close\_all\_windows()  
 else:  
 event.ignore()  
  
  
# Функция закрытия всех окон  
def close\_all\_windows():  
 win\_list = QApplication.allWindows()  
 for w in win\_list:  
 w.close()  
  
  
class AlgParamsWindow(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.setWindowTitle("Параметры работы алгоритма")  
 self.setFixedSize(300, 200)  
 self.move(1000, 300)  
  
 self.iterationWindow = IterationWindow()  
  
 layout = QVBoxLayout()  
 widgets = []  
  
 self.label1 = QLabel("Максимальное кол-во итераций:")  
 self.label1.adjustSize()  
 widgets.append(self.label1)  
  
 self.label2 = QLabel("Вероятность скрещевания:")  
 self.label2.adjustSize()  
  
 self.label3 = QLabel("Вероятность мутации:")  
 self.label3.adjustSize()  
  
 self.inputText1 = QLineEdit()  
 widgets.append(self.inputText1)  
 widgets.append(self.label2)  
 self.inputText2 = QLineEdit()  
 widgets.append(self.inputText2)  
 widgets.append(self.label3)  
 self.inputText3 = QLineEdit()  
 widgets.append(self.inputText3)  
  
 self.startButton = QPushButton("Запуск!")  
 self.startButton.adjustSize()  
 self.startButton.clicked.connect(self.startAlgorithm)  
 widgets.append(self.startButton)  
  
 for w in widgets:  
 layout.addWidget(w)  
  
 widget = QWidget()  
 widget.setLayout(layout)  
  
 self.setCentralWidget(widget)  
  
 def startAlgorithm(self):  
 self.close()  
 self.iterationWindow.show()  
  
  
class InputWindow(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.iterationAlg = IterationWindow()  
  
 self.setWindowTitle("Ввод данных")  
 # self.setFixedSize(300, 500)  
 self.move(1000, 300)  
  
 layout = QVBoxLayout()  
 layout.setSpacing(5)  
 # layout.addStretch(1)  
 # layout.setContentsMargins(0, 0, 0, 0)  
 widgets = []  
  
 self.label1 = QLabel("Введите вес рюкзака: ")  
 widgets.append(self.label1)  
  
 self.inputAmount = QLineEdit()  
 widgets.append(self.inputAmount)  
  
 self.label2 = QLabel("Далее введите данные в формате:\nВес Цена (каждый премедет с новой строки)")  
 self.label2.adjustSize()  
 widgets.append(self.label2)  
  
 self.inputData = QPlainTextEdit()  
 self.inputData.setFixedSize(200, 200)  
 widgets.append(self.inputData)  
  
 self.startButton = QPushButton("Запуск!")  
 self.startButton.clicked.connect(self.startAlg)  
 widgets.append(self.startButton)  
  
 for w in widgets:  
 layout.addWidget(w)  
  
 widget = QWidget()  
 widget.setLayout(layout)  
  
 self.setCentralWidget(widget)  
  
 def startAlg(self):  
 self.close()  
 self.iterationAlg.show()  
  
  
class IterationWindow(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
  
 self.iteration = 0  
 self.someData = list()  
 self.iterationLine = "Итерация № "  
  
 self.setWindowTitle("Работа алгоритма")  
 # self.setFixedSize(300, 500)  
 # self.move(1000, 300)  
  
 layout = QVBoxLayout()  
 widgets = []  
  
 self.iterationLabel = QLabel((self.iterationLine + str(self.iteration)))  
 widgets.append(self.iterationLabel)  
 self.dataLabel = QLabel("какая то дата")  
 widgets.append(self.dataLabel)  
  
 self.graphicLabel = QLabel()  
 self.graphicAlgorithm = QPixmap('/Users/raregod/Downloads/cat.jpg')  
 smaller\_pixmap = self.graphicAlgorithm.scaled(250, 250, Qt.AspectRatioMode.KeepAspectRatio)  
 self.graphicLabel.setPixmap(smaller\_pixmap)  
 widgets.append(self.graphicLabel)  
  
 self.hLayout = QHBoxLayout()  
 HWidgets = []  
  
 self.backButton = QPushButton("<")  
 HWidgets.append(self.backButton)  
 self.forwardButton = QPushButton(">")  
 HWidgets.append(self.forwardButton)  
 self.finishButton = QPushButton(">>")  
 HWidgets.append(self.finishButton)  
  
 for w in HWidgets:  
 self.hLayout.addWidget(w)  
  
 for w in widgets:  
 layout.addWidget(w)  
  
 layout.addLayout(self.hLayout)  
 widget = QWidget()  
 widget.setLayout(layout)  
  
 self.setCentralWidget(widget)

Название файла: main.py

import sys # Только для доступа к аргументам командной строки  
  
from PyQt6.QtWidgets import QApplication  
  
from src.gui import Window  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
  
 wind = Window()  
 wind.show()  
  
 app.exec()