Государственное бюджетное общеобразовательное учреждение

лицей № 393 Кировского района Санкт-Петербурга

(ГБОУ лицей № 393 Кировского района Санкт-Петербурга)

Руководство программиста

**Генератор задания №1 ЕГЭ**

Автор проекта: Ермак Михаил, ученик 10 «А» класса

Руководитель проекта: Смирнова Надежда Алексеевна, учитель информатики

Санкт-Петербург

2024

Содержание

[Аннотация 3](#_Toc98109664)

[Постановка задачи 4](#_Toc98109665)

[Формализация алгоритма 5](#_Toc98109666)

[Листинг программы 6](#_Toc98109667)

[Тестовые примеры 23](#_Toc98109668)

[Описание размещения 24](#_Toc98109669)

[Требования к программным и аппаратным средствам 25](#_Toc98109670)

# Аннотация

Данный документ представляет собой краткое описание самой программы, её интерфейсов, размещения, требования к аппаратным и программным средствам.

Проект "Генератор заданий номер 1 ЕГЭ по информатике на Python" представляет собой программу, которая генерирует случайные задания из раздела номер 1 ЕГЭ по информатике и выводит их пользователю. Программа написана на языке программирования Python и использует различные математические операции, условные операторы и циклы для создания разнообразных заданий. Проект поможет учащимся подготовиться к сдаче ЕГЭ по информатике, практикуя решение различных задач

# Постановка задачи

Целью моего проекта было создать программу, которая будет автоматически генерировать уникальные задания номер 1 ЕГЭ по информатике, так же необходимо реализовать графический интерфейс с помощью библиотеки PyQT5 для того, чтобы пользователю было удобнее пользоваться этой программой.

# Формализация алгоритма

1. Библиотека PYQT5 для реализации графического интерфейса программы.
2. Библиотека random для генерации и перемешивания чисел.
3. Библиотека matplotlib, network для генерации изображения графа.
4. Библиотека collections для реализации очереди.
5. Функция final\_generation, включающиеся в нее функции: matrix\_generation, matrix\_check, saving\_matrix, отвечают за функцию создания графа, создания изображения этого графа.

# Листинг программы

from PyQt5.QtWidgets import QApplication, QMainWindow  
from PyQt5.uic import loadUi  
from PyQt5.QtGui import QPixmap  
import sys  
import random  
import networkx as nx  
import matplotlib.pyplot as plt  
from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg as FigureCanvas  
from collections import deque  
  
  
class MainWindow(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 loadUi('ui.ui', self)  
 self.pushButton.clicked.connect(self.final\_generation)  
 self.radioButton.toggled.connect(self.on\_radio\_button\_toggled)  
 self.radioButton\_2.toggled.connect(self.on\_radio\_button\_toggled)  
 self.radioButton.setChecked(True)  
 self.pushButton\_2.clicked.connect(self.show\_answer)  
 self.pushButton\_2.hide()  
 self.label\_5.setVisible(False)  
 self.show()  
  
 # Проверяем, какой тип задания выбрал пользователь, если нет, то 1  
  
 type\_of\_task = 1  
 def on\_radio\_button\_toggled(self):  
 global type\_of\_task  
 sender = self.sender()  
 if sender.isChecked():  
 if sender == self.radioButton:  
 type\_of\_task = 1  
 elif sender == self.radioButton\_2:  
 type\_of\_task = 2  
  
 def show\_answer(self):  
 self.label\_5.setVisible(True)  
  
 def final\_generation(self):  
 global answer\_for\_type2  
 # Генерация матрицы размером size со случайным заполнением 0 и 1  
 def matrix\_generation(size):  
 array = [[0 for \_ in range(size)] for \_ in range(size)]  
 for i in range(size):  
 for j in range(i + 1, size):  
 array[i][j] = random.randint(0, 1)  
 array[j][i] = array[i][j]  
 return array  
  
 # Проверка матрицы на требование: не менее 2 и не более 4 "1" в каждой строке матрицы  
 def matrix\_check(array):  
 for row in array:  
 count = sum(row)  
 if not (count >= 2 and count <= 3):  
 return False  
 return True  
  
 # Финальное создание задания и сохранение графа  
 def saving\_matrix(resultmatrix):  
 # G - граф  
 G = nx.Graph()  
  
 # Случайно перемешиваем буквы из списка letters  
 letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']  
 random.shuffle(letters)  
 nums = []  
 for i in range(len(resultmatrix)):  
 G.add\_node(i)  
 nums.append(i)  
  
 # Создаем словарь вида Номер точки на графе: Буквенное обозначение  
 answer = dict(zip(nums, letters))  
  
 # Случайно выбираем буквы, номер которых нужно будет найти пользователю. Генерация происходит до тех пор, пока буквы будут различными  
 required\_letter\_1 = random.choice(letters)  
 required\_letter\_2 = random.choice(letters)  
 while not(required\_letter\_1 != required\_letter\_2):  
 required\_letter\_2 = random.choice(letters)  
  
 # Создаем словарь вида Буквенное обозначение: Номер точки на графе  
 answer\_2 = dict(zip(letters, nums))  
  
 # Задаем ребра графа  
 for i in range(len(resultmatrix)):  
 for j in range(i + 1, len(resultmatrix)):  
 if resultmatrix[i][j] == 1:  
 G.add\_edge(i, j)  
  
 # Создание нового графика с помощью библиотеки Matplotlib. fig представляет собой объект Figure, а ax - объект Axes, который представляет область построения на графике.  
 fig, ax = plt.subplots()  
  
 # Вычисление позиции узлов графа G алгоритмом spring\_layout  
 pos = nx.spring\_layout(G)  
  
 # Рисуем граф  
 nx.draw(G, pos, with\_labels=True, font\_weight='bold', font\_size=10, labels=answer)  
  
 # Создаем фигуру canvas с помощью FigureCanvasQTAgg  
 canvas = FigureCanvas(fig)  
  
 # Присваимваем label\_3 фигуру canvas  
 self.label\_3.setPixmap(QPixmap(canvas.grab()))  
 self.label\_3.setVisible(True)  
 answer\_text = "Ответы:\n"  
 for node, letter in answer.items():  
 answer\_text += f"П{node+1}: {letter}, "  
 self.label\_2.setText(f'На рисунке справа схема дорог Н-ского района изображена в виде графа, звёздочка в ячейке таблицы обозначает наличие дороги между двумя пунктами. Так как таблицу и схему рисовали независимо друг от друга, то нумерация населённых пунктов в таблице никак не связана с буквенными обозначениями на графе. Определите, какие номера населённых пунктов в таблице могут соответствовать населённым пунктам {required\_letter\_1} и {required\_letter\_2} на схеме. В качестве ответа перечислите найденные номера в любом порядке.')  
 self.label\_5.setText(f'Ответ: {answer\_2[required\_letter\_1]+1}, {answer\_2[required\_letter\_2]+1}')  
 self.pushButton\_2.show()  
 self.label\_5.setVisible(False)  
 return answer  
  
  
 resulting\_matrix = matrix\_generation(8)  
 while not matrix\_check(resulting\_matrix):  
 resulting\_matrix = matrix\_generation(8)  
  
  
 saving\_matrix(resulting\_matrix)  
  
 if type\_of\_task == 1:  
 for i in range(8):  
 for j in range(8):  
 if resulting\_matrix[i][j] == 1:  
 resulting\_matrix[i][j] = '\*'  
 else:  
 resulting\_matrix[i][j] = '-'  
 else:  
 answer = saving\_matrix(resulting\_matrix)  
 required\_number\_1 = random.randint(0, 7)  
 required\_number\_2 = random.randint(0, 7)  
 while not (required\_number\_1 != required\_number\_2):  
 required\_number\_2 = random.randint(0, 7)  
 adj\_list = {}  
 for i in range(8):  
 for j in range(8):  
 if resulting\_matrix[i][j] == 1:  
 resulting\_matrix[i][j] = random.randint(1, 9)  
 else:  
 resulting\_matrix[i][j] = '-'  
  
 for i in range(len(resulting\_matrix)):  
 for j in range(len(resulting\_matrix[i])):  
 if resulting\_matrix[i][j] != '-':  
 if i not in adj\_list:  
 adj\_list[i] = []  
 adj\_list[i].append((j, int(resulting\_matrix[i][j])))  
  
 def bfs(adj\_list, start, end):  
 visited = set()  
 queue = deque([(start, 0)]) # Добавляем начальную вершину и длину пути  
 visited.add(start)  
 while queue:  
 node, distance = queue.popleft()  
 if node == end:  
 return distance  
 if node in adj\_list:  
 for neighbor, edge\_distance in adj\_list[node]:  
 if neighbor not in visited:  
 queue.append((neighbor, distance + edge\_distance))  
 visited.add(neighbor)  
 path\_length = bfs(adj\_list, required\_number\_1, required\_number\_2)  
 self.label\_2.setText(  
 f'На рисунке справа схема дорог Н-ского района изображена в виде графа, звёздочка в ячейке таблицы обозначает наличие дороги между двумя пунктами. Так как таблицу и схему рисовали независимо друг от друга, то нумерация населённых пунктов в таблице никак не связана с буквенными обозначениями на графе. Определите протяженность пути между населённым пунктам {answer[required\_number\_1]} и {answer[required\_number\_2]} на схеме. В ответе запишите целое число.')  
 self.label\_5.setText(f'Ответ: {path\_length}')  
  
 result\_text = " П1: П2: П3: П4: П5: П6: П7: П8: \n"  
 for i, row in enumerate(resulting\_matrix, 1):  
 result\_row = ''  
 for j in row:  
 result\_row += (' '+ str(j) + ' ')  
 result\_text += f"П{i}: {result\_row}\n"  
  
 self.label\_4.setText(result\_text)  
 self.label\_2.setVisible(True)  
 self.label.setVisible(False)  
 self.pushButton.setText('Далее')  
 self.pushButton.move(10, 466)  
 self.radioButton.move(130, 465)  
 self.radioButton\_2.move(130, 485)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 window = MainWindow()  
 window.show()  
 app.exec\_()

# Тестовые примеры

После запуска программы, пользователь видит надпись “Добро пожаловать!”, кнопку “Начать!” и 2 кнопки “1 тип” и “2 тип”. Чтобы начать генерацию, пользователь должен выбрать один из предложенных типов задания и нажать на кнопку “Начать!”. После этого, программа выводит текст задания, граф, таблицу, в соответствии с выбранным типом задания. После решения выведенной задачи, пользователю предлагается нажать на кнопку “Проверить ответ”, чтобы сравнить получившийся ответ с правильным. После этого, пользователь может выбрать один из двух типов заданий и после нажатия на кнопку “Далее”, сгенерировать новое задание.

# Описание размещения

Для запуска программы «Генератор задания №1» требуется файл с программой и установленный на компьютер модули PyQT5, matplotli для языка программирования Python.

# Требования к программным и аппаратным средствам

Язык программирования: Python

Версия программы для программирования: IDLE (Python 3.9 32-bit)

Версия офисной программы, с помощью которой была подготовлена документация: Microsoft Word

Операционная система: Windows 11 Pro (дата установки – 14.08.2023)

Характеристика компьютера, на котором точно работает программа, программа может работать и на слабых устройствах:

1. AMD Ryzen 5 3600
2. RAM 16gb 3200mhz
3. AMD Radeon RX6600