МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

Інформатики та програмної інженерії

(повна назва кафедри, циклової комісії)

КУРСОВА РОБОТА

З «Основ програмування. Частина 2. Модульне програмування»

(назва дисципліни)

на тему: 8-puzzle

Студентки 1-го курсу, групи ІП-22

Семенова Єлизавета Олександрівна

Спеціальності 121 «Інженерія програмного забезпечення »

Керівник ст.вик. Головченко М.М.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Кількість балів: _____

Національна оцінка _____

Члени комісії

_____ к.т.н. доц. Муха І.П.
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

_____ ас. доц. Вовк Є.А.
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ - 2023 рік

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра інформатики та програмної інженерії

Дисципліна Основи програмування

Напрям "ІПЗ"

Курс 1 Група ІП-22 Семестр 2

ЗАВДАННЯ

на курсову роботу студента

Семенової Єлизавети Олександрівни

(прізвище, ім’я, по батькові)

1. Тема роботи 8-puzzle методом A* та RBFS
2. Строк здачі студентом закінченої роботи 25.06.2023
3. Вихідні дані до роботи Додаток А Технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці) Постановка задачі, теоретичні відомості, опис алгоритмів, опис програмного забезпечення, тестування програмного забезпечення, інструкція користувача, аналіз і узагальнення результатів

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

6. Дата видачі завдання 12.02.2023

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	12.02.2023	
2.	Підготовка ТЗ	25.02.2023	
3.	Пошук та вивчення літератури з питань курсової роботи	28.03.2023	
4.	Розробка сценарію роботи програми	12.04.2023	
6.	Узгодження сценарію роботи програми з керівником	18.04.2023	
5.	Розробка (вибір) алгоритму рішення задачі	24.04.2023	
6.	Узгодження алгоритму з керівником	02.05.2023	
7.	Узгодження з керівником інтерфейсу користувача	02.05.2023	
8.	Розробка програмного забезпечення	08.05.2023	
9.	Налагодження розрахункової частини програми	08.05.2023	
10.	Розробка та налагодження інтерфейсної частини програми	08.05.2023	
11.	Узгодження з керівником набору тестів для контрольного прикладу	16.05.2023	
12.	Тестування програми	19.05.2023	
13.	Підготовка пояснювальної записки	25.05.2023	
14.	Здача курсової роботи на перевірку	06.06.2023	

Студент

(підпис)

Керівник _____ Головченко М.М.

(підпис) (прізвище, ім’я, по батькові)

“12” лютого 2023 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 97 сторінок, 30 рисунків, 14 таблиць, 3 посилання.

Мета роботи: розробка якісного та надійного ПЗ для розв’язання головоломки 8-puzzle за допомогою алгоритмів A* та RBFS.

Вивчено алгоритми A* та RBFS.

Виконана програмна реалізація алгоритмів A* та RBFS. Розроблено застосунок з графічним інтерфейсом для розв’язання головоломки 8-puzzle одним із заданих двох методів. Роботу застосунку протестовано на різних значеннях та у різних умовах.

8-PUZZLE, АЛГОРИТМ A*, АЛГОРИТМ RBFS, МАНХЕТТЕНСЬКА ВІДСТАНЬ, РОЗВ’ЯЗНІСТЬ 8-PUZZLE.

ЗМІСТ

ВСТУП

Головоломка 8-puzzle є класичною задачею у сфері Штучного Інтелекту. Вона представляє собою дошку розміром 3x3 з відсутнім квадратом та числами від 1 до 8, розташованими у довільному порядку. Мета полягає у впорядкуванні чисел за зростанням шляхом переміщення плиток та досягнення кінцевої конфігурації.

Застосування алгоритмів A* та RBFS для розв'язання головоломки 8-puzzle має декілька важливих переваг. По-перше, ці алгоритми забезпечують оптимальні рішення, що означає, що знаходять найкоротший шлях до цільової конфігурації. По-друге, вони можуть працювати зі значними обсягами даних та складними конфігураціями, що робить їх ефективними для розв'язання складних головоломок.

Крім того, розв'язання головоломки 8-puzzle є важливим завданням у багатьох галузях. В штучному інтелекті ці алгоритми можуть бути застосовані для пошуку оптимальних шляхів у складних системах, таких як навігація роботів або планування маршрутів. В оптимізації розподілених систем вони можуть бути використані для розподілу завдань між вузлами та мінімізації часу виконання. У галузі робототехніки алгоритми розв'язання головоломки 8-puzzle можуть бути застосовані для планування рухів та уникнення перешкод. Крім того, в комп'ютерних іграх вирішення головоломки 8-puzzle дозволяє створити складні та цікаві головоломки для гравців.

Оскільки головоломка 8-puzzle є важкою задачею зі значною кількістю можливих станів та шляхів, розробка програмного забезпечення, яке ефективно розв'язує її, є актуальною та цікавою в галузі комп'ютерних наук.

1 ПОСТАНОВКА ЗАДАЧІ

Розробити програмне забезпечення, що буде розв’язувати головоломку 8-puzzle за допомогою таких методів:

- метод A*;
- метод RBFS.

Вхідними даними для даної роботи є початковий та кінцевий стани головоломки, задані користувачем або згенеровані програмою.

Вихідними даними програми є покроковий шлях розв’язку головоломки за допомогою обраного алгоритму. Ці дані виводяться за допомогою графічного інтерфейсу, а також текстового файлу.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ

Задача гри у вісім (8-puzzle) складається з дошки 3×3 з вісьмома пронумерованими фішками і з одним порожнім місцем. В програмі кожен стан дошки визначається масивом, що є певним порядком чисел 1-8 та порожнього місця, позначеного пробілом (« »).

Оскільки в кожній задачі ми знаємо початковий та кінцевий стани, то є можливість перевірити розв’язність цих конфігурацій. Це можна зробити, порахувавши кількість інверсій у кожному стані, виключаючи порожнє місце, і порівнявши ці значення. Якщо обидва стани мають однакову парність інверсій (обидва парні або обидва непарні), то задача має розв'язок. В іншому випадку, розв'язку для такої задачі не існує.

У цій роботі ми шукатимемо розв’язок задач наступними методами:

- A*;
- RBFS.

A* (читається як «А зірочка») - різновид пошуку за першим найкращим збігом. Для оцінки вузлів у ньому використовується функція $g(n)$ - вартість досягнення поточного вузла і $h(n)$ - вартість проходження від поточного вузла до цілі. Отже загальна оцінка має такий вигляд:

$$f(n) = g(n) + h(n)$$

$h(n)$- прийнятна евристична функція. У нашому випадку буде використовуватись евристика Манхеттенської відстані - сума відстаней між кожним елементом і його правильною позицією.

Як можна побачити на Рисунку 2.1, за допомогою використання поточної вартості шляху та евристики алгоритм A* на кожному кроці обирає найбільш вигідний вузол, тобто вузол, в якого сума цих значень буде найменшою. При цьому зберігається інформація про всі відкриті вузли.

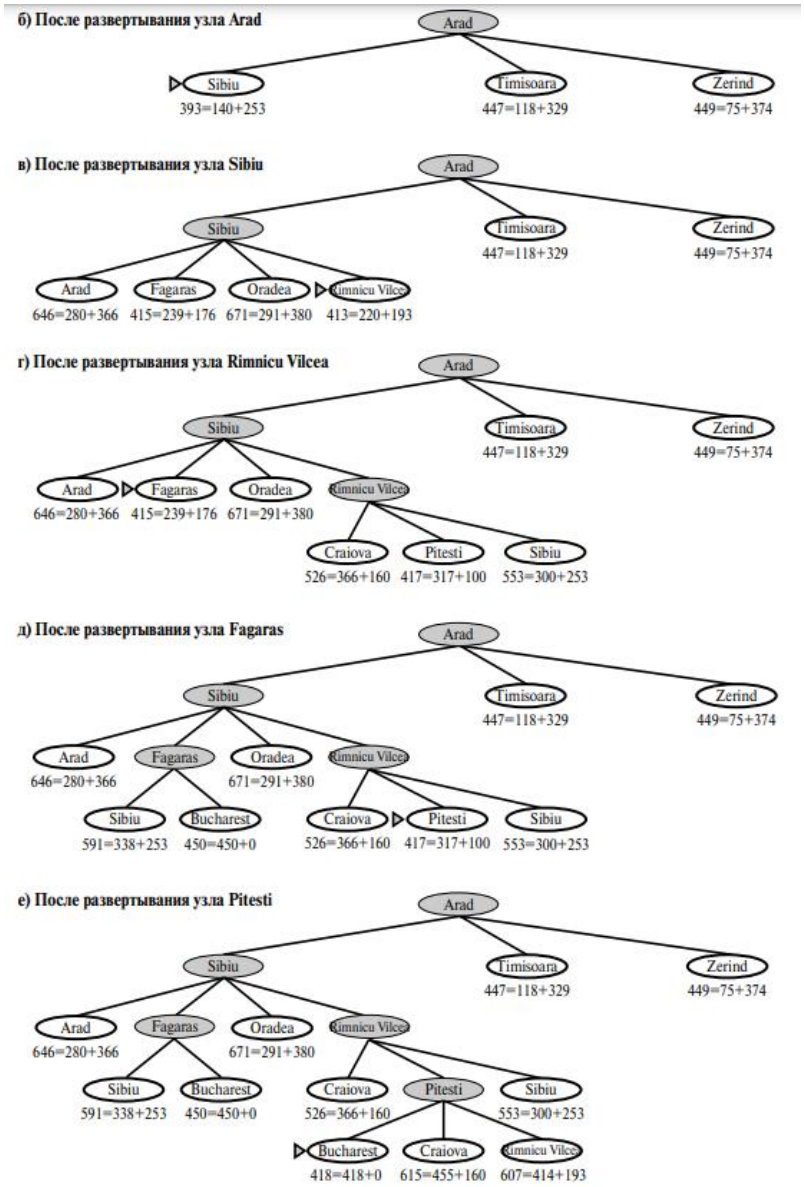


Рис. 2.1 - Пошук найкоротшого шляху до міста Бухарест за допомогою алгоритму A*. Кожен вузол відмічений відповідним значенням f = g + h, де h – відстань від міста до Бухаресту по прямій.

Алгоритм пошуку за першим найкращим збігом (Recursive Best-First Search - RBFS) - це простий рекурсивний алгоритм, який шукає найкращий шлях у графі. Він зберігає лише вузли поточного шляху і використовує значення f-функції для прийняття рішень. Якщо поточний вузол перевищує

ліміт f-значення найкращого альтернативного шляху, алгоритм відміняє поточний етап рекурсії і продовжує пошук з альтернативного шляху.

Після відміни етапу рекурсії, алгоритм заміняє значення f-функції кожного вузла на найкраще значення серед його дочірніх вузлів. Це дозволяє запам'ятати f-значення найкращого листового вузла і в разі потреби розгорнути це піддерево знову. Приклад роботи алгоритму наведено на Рисунку 2.2.

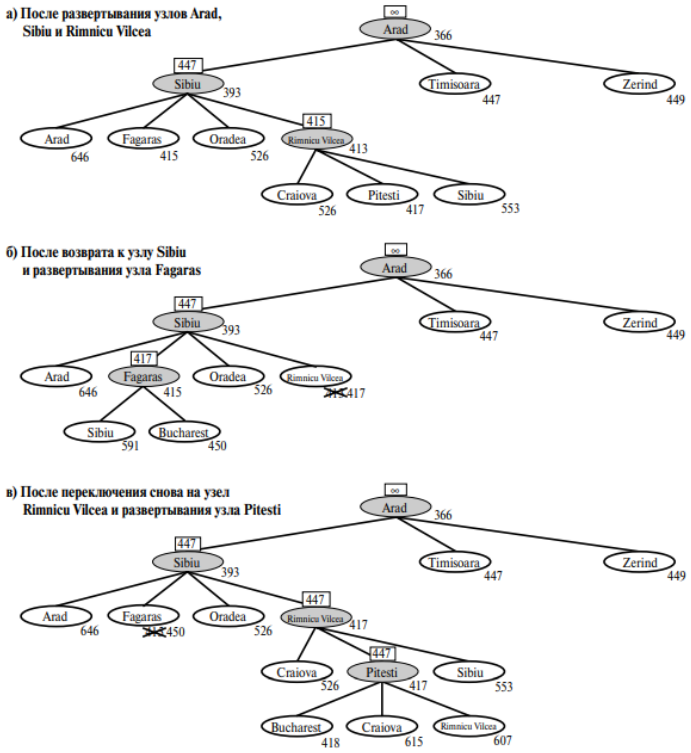


Рис. 2.2 - Пошук найкоротшого шляху до міста Бухарест за допомогою алгоритму RBFS. Кожен вузол відмічений f-лімітом поточного етапу.

Отже, алгоритм RBFS використовує значно менше пам’яті, ніж A*, адже зберігаються дані лише вузлів з поточного шляху, а не всіх, що були відкриті. Але при цьому алгоритм A* зазвичай швидший, бо не має потреби знов відкривати вузли, які вже були розглянуті.

3 ОПИС АЛГОРИТМІВ

Перелік всіх основних змінних і їхнє призначення наведено в таблиці 3.1.

Таблиця 3.1 — Основні змінні та їхні призначення

Змінна	Призначення
start	Початковий стан головоломки
end	Кінцевий стан головоломки
path	Шлях розв’язку головоломки
is_valid	Прапорець, що позначає коректність введених даних
is_unique	Прапорець, що позначає унікальність введених даних
start_node	Корінь дерева пошуку розв’язку
node	Поточний вузол
children	Нащадки вузла
successors	Структура, що зберігає дані про вузли, їх оціночну функцію та іншу інформацію, яка потрібна для прийняття рішень про наступний крок
count	Лічильник, що враховує кількість створених вузлів та використовується як додатковий критерій вибору кращого вузла
explored	Список відвіданих вузлів
f_limit	Ліміт f-значення найкращого альтернативного шляху
best_node	Вузол з найменшим f-значенням
alternative	Вузол з другим найменшим f-значенням

3.1. Загальний алгоритм

1. ПОЧАТОК

2. ЯКЩО обрано ввести дані задачі, ТО перейти до пункту 3. ІНАКШЕ згенерувати валідні дані задачі, вивести їх у вікні Випадкова задача та перейти до пункту 4

3. Зчитати початковий та кінцевий стани з системи інтерфейсу:

3.1. Позначити прапорці is_unique та is_valid як True

3.2. ЦИКЛ проходу по всіх елементах макету.

3.2.1. Отримання значення з елементу. ЯКЩО значення валідне, ТО повертається це значення. ІНАКШЕ повертається значення None

3.2.2. ЯКЩО значення елементу - None, ТО змінюємо значення прапорця is_valid на False та переходимо до наступного елемента

3.2.3. ЯКЩО таке ж значення вже було зчитано, ТО змінюємо значення прапорця is_unique на False

3.3. ЯКЩО усі елементи унікальні та валідні, то повернути значення введених станів у масиви start та end та перейти до пункту 5. ІНАКШЕ стани не змінюються.

4. Згенерувати початковий та кінцевий стани:

4.1. ПОВТОРИТИ Згенерувати два масиви start та end з випадковим розташуванням чисел від 1 до 8 ПОКИ парність інверсій цих масивів не буде однакова

4.2. Додати у випадкове місце в масиві start елемент « »

4.3. Додати у випадкове місце в масиві end елемент « »

4.4. Вивести згенеровані стани

4.5. Перейти до пункту 5

5. Розв’язок задачі:

5.1. ЯКЩО не введено початковий і кінцевий стани, ТО вивести помилку

5.2. ЯКЩО обрано алгоритм A*, ТО присвоїти масиву path результат виконання алгоритму A*. ІНАКШЕ ЯКЩО обрано алгоритм RBFS, то присвоїти масиву path результат виконання алгоритму RBFS. ІНАКШЕ вивести помилку

5.3. ЦИКЛ проходу по всіх елементах масиву path:

5.3.1. Вивести елемент у поле виведення розв’язку головоломки та зачекати 1 секунду

5.4. Вивести повідомлення про завершення роботи

6. Записати дані в файл:

6.1. ЯКЩО жодного розв’язку не було знайдено, ТО вивести помилку

6.2. Відкрити файл

6.3. ЦИКЛ проходу по всіх елементах масиву path:

6.3.1. Записати елемент у файл

6.4. Вивести повідомлення про закінчення роботи

7. КІНЕЦЬ

3.2. Алгоритм A*
1. ПОЧАТОК

2. Присвоїти count = 0

3. Створення порожнього масиву explored

4. Створення початкового вузла start_node з заданими початковою та кінцевою конфігурацією, відсутнім батьківським вузлом та дією, і вагою пройденого шляху 0

5. ЯКЩО початковий вузол є нерозв’язним (**start_node.is_solvable()** повертає значення False), ТО повернути None

6. Створення черги з пріоритетом successors

7. Додавання в чергу з пріоритетом successors кортеж, що складається з значення оціночної функції початкового вузла, лічильника count і початкового вузла

8. ПОКИ successors не порожня:

8.1. Запис елементу черги successors з найвищим пріоритетом в змінну node та видалення цього елементу з черги

8.2. Присвоїти node = node[2]

8.3. Додати node.state до масиву explored

8.4. ЯКЩО node дорівнює кінцевому стану, ТО повернути відновлений шлях від початкового стану до поточного (**node.find_solution()**)

8.5. Створити масив children та заповнити його нащадками вузла node (**node. generate_child()**)

8.6. ЦИКЛ проходу по всіх нащадках поточного вузла (child):

8.6.1. ЯКЩО node.state не знаходиться в explored, ТО збільшити count на 1 та додати до черги successors кортеж, що складається зі значення оціночної функції вузла child (**child.evaluation_function**), лічильника count і вузла child

9. Повернути None

10. КІНЕЦЬ

3.3. Алгоритм RBFS
1. ПОЧАТОК

2. Створення початкового вузла start_node з заданими початковою та кінцевою конфігурацією, відсутнім батьківським вузлом та дією, і вагою пройденого шляху 0

3. ЯКЩО початковий вузол є нерозв’язним (**start_node.is_solvable()** повертає значення False), ТО повернути None

4. Присвоїти node результат виконання функції RBFS з аргументами start_node та sys.maxsize

5. Присвоїти node = node[0]

6. Повернути відновлений шлях від початкового стану до поточного (**node.find_solution()**)

RBFS(node, f_limit):

1. ПОЧАТОК

2. Створення порожнього масиву successors

3. ЯКЩО node дорівнює кінцевому стану, ТО повернути node та None

4. Створити масив children та заповнити його нащадками вузла node (**node. generate_child()**)

5. ЯКЩО вузол не має нащадків, ТО повернути None та sys.maxsize

6. Зменшити значення count на 1

7. ЦИКЛ проходу по всіх нащадках поточного вузла (child):

    7.1. Збільшити значення count на 1

    7.2. Додати до масиву successors кортеж, що складається зі значення оціночної функції вузла child (**child.evaluation_function**), лічильника count і вузла child

8. ПОКИ довжина successors!= 0:

    8.1. Відсортувати масив successors

    8.2. Присвоїти best_node = successors [0][2]

8.3. ЯКЩО значення оціночної функції вузла best_node (**best_node.evaluation_function**) більше, ніж f_limit, ТО повернути None та best_node.evaluation_function

8.4. ЯКЩО нащадків більше 1, ТО присвоїти alternative = successors[1][0] та f_limit присвоїти мінімальне значення між поточним значенням f_limit та alternative

8.5. Присвоїти result та best_node.evaluation_function значення, яке повертатиме функція RBFS з аргументами best_node та f_limit

8.6. Присвоїти successors[0] кортеж, що складається зі значення оціночної функції вузла best_node (**best_node.evaluation_function**), лічильника count і вузла best_node

8.7. ЯКЩО значення result не None, ТО вийти з циклу

9. Повернути result та None

10. КІНЕЦЬ

4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Діаграма класів програмного забезпечення

На Рисунку 4.1 зображено 8 класів представлених в програмі та класи з бібліотеки PyQt5, від яких наслідуються класи програми.

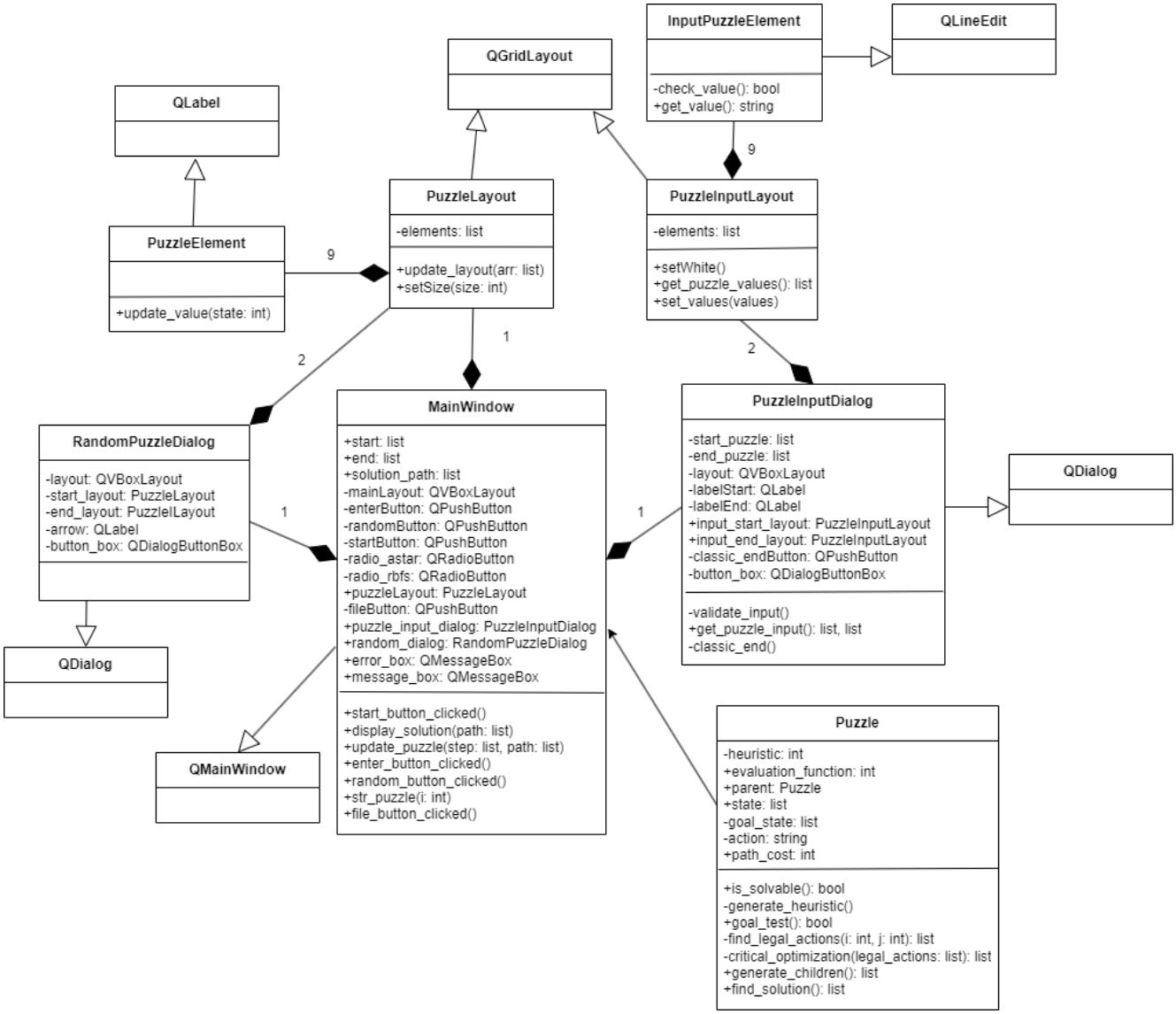


Рисунок 4.1. - Діаграма класів.

PuzzleElement - клас, що відповідає за виведення елементу пазла. Наслідується від класу QLabel.

InputPuzzleElement - клас, що відповідає за введення елементу пазла, а також за перевірку на його коректність. Наслідується від класу QLineEdit.

PuzzleLayout - клас, що відповідає за створення поля для виведення головоломки. Наслідується від класу QGridLayout та включає клас PuzzleElement.

PuzzleInputLayout - клас, що відповідає за створення поля для введення головоломки та перевірки елементів головоломки на унікальність. Наслідується від класу QGridLayout та включає клас PuzzleInputElement.

PuzzleInputDialog - клас, що відповідає за створення вікна для введення початкового і кінцевого станів головоломки. Також має метод для створення кінцевого стану з класичною конфігурацією. Також має перевірку на валідність введених значень. Клас наслідується від класу QDialog та включає клас PuzzleInputLayout.

RandomPuzzleDialog - клас, що відповідає за створення вікна для виведення згенерованих початкового і кінцевого станів головоломки. Клас наслідується від класу QDialog та включає клас PuzzleLayout.

MainWindow - клас, що відповідає за створення і роботу з головним вікном. Наслідується від класу QMainWindow. Включає класи PuzzleLayout, PuzzleInputDialog та RandomPuzzleDialog. Має методи, що виконуються після натискання кнопок для введення, генерації, розв’язання та запису в файл задачі. Має асоціативний зв’язок з класом Puzzle.

Puzzle - клас, що відповідає за утворення вузла в дереві пошуку розв’язку задачі. Зберігає поточний та кінцевий стан, евристику та оціночну функцію, батьківський об’єкт та дію, яка була виконана для його утворення. Має методи для перевірки чи є задача розв’язною і чи відповідає поточний стан кінцевому, генерації евристики, можливих подій та нащадків, а також відтворення шляху від листового вузла дерева пошуку до кореня. Має асоціативний зв’язок з класом MainWindow.

4.2. Опис методів програмного забезпечення

4.2.1. Стандартні методи

У таблиці 4.1 наведено використані у програмі стандартні методи.

Таблиця 4.1 - Стандартні методи

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
1	list	append	Додає елемент до кінця списку	item - елемент, який необхідно додати до списку	Відсутні	builtins
2	list	pop	Видаляє та повертає останній елемент списку.	index (за замовченням останній) - індекс елемента, який потрібно видалити із списку.	Значення видаленого елементу	builtins

Продовження таблиці 4.1

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
3	list	insert	Вставляє елемент на певну позицію в списку.	Index - індекс, на який потрібно вставити елемент; value - значення елемента, який потрібно вставити.	Відсутні	builtins
4	list	index	Повертає індекс першого входження елемента в списку.	value - значення елемента, для якого потрібно знайти індекс; start (опціональний) - індекс, з якого починається пошук; end (опціональний) - індекс, на якому закінчується пошук.	Ціле число - індекс першого входження шуканого елемента в списку.	builtins

Продовження Таблиці 4.1

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголо-вний файл
5	list	sort	Сортує елементи списку за зростанням.	key (опціональ-ний) - функція, яка визначає ключ сортування; reverse (опціональ-ний)- логічне значення чи потрібно сортувати у зворотному порядку.	Відсутні. Метод модифікує сам список.	builtins
6	Відсу-тній	copy	Створює копію об'єкту	Відсутні	Повертає новий об'єкт, який є копією вихідного об'єкта.	copy
7	list	remove	Видаляє перший елемент зі списку, який має задане значення.	value - значення елемента, який потрібно видалити зі списку.	Відсутні	builtins

Продовження Таблиці 4.1

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
8	list	reverse	Змінює порядок елементів списку на зворотній.	Відсутні	Відсутні. Метод модифікує сам список.	builtins
9	Відсутній	any	Перевіряє, чи є хоча б один елемент в ітерабельному об'єкті, який вважається істинним.	Ітерабельний об'єкт (список, кортеж, множина тощо).	True, якщо принаймні один елемент у об'єкті є істинним; False, якщо всі елементи є хибними або якщо об'єкт порожній.	builtins
10	Відсутній	join	Об'єднує елементи в ітерабельному об'єкті в один рядок, розділені заданим роздільником.	delim - роздільник (рядок); об'єкт (список, кортеж, рядок тощо), який містить елементи для об'єднання.	Повертає новий рядок, в якому елементи об'єкту об'єднані за допомогою заданого роздільника.	builtins

Продовження Таблиці 4.1

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
11	Відсутній	map	Застосовує задану функцію до кожного елемента в ітерабельному об'єкті і повертає результат у вигляді ітератора.	функцію, яка буде застосовуватися до кожного елемента; об'єкт (список, кортеж, рядок тощо), який містить елементи, до яких буде застосовуватися функція.	Повертає ітератор, який містить результати застосування функції до кожного елемента ітерабельного об'єкту.	builtins
12	Відсутній	range	Генерує послідовність цілих чисел від початкового значення до кінцевого значення з певним кроком.	початкове значення (за замовчуванням 0), кінцеве значення (не включається) і крок (за замовчуванням 1).	Повертає ітератор, який генерує послідовність цілих чисел.	builtins

Продовження Таблиці 4.1

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
13	Відсутній	len	Повертає кількість елементів у колекції або довжину об'єкта.	колекція або об'єкт, для якого потрібно обчислити кількість елементів або довжину.	ціле число, яке відповідає кількості елементів у колекції або довжині об'єкта.	builtins
14	str	isdigit	Перевіряє, чи складається рядок лише з цифрових символів.	Відсутні	булеве значення яке показує, чи складається рядок лише з цифр.	builtins
15	Відсутній	min	Повертає найменше значення з переданого ітерабельного об'єкту або серед переданих аргументів.	один або більше аргументів або один ітерабельний об'єкт	найменше значення з переданого ітерабельного об'єкту або серед переданих аргументів	builtins

Продовження Таблиці 4.1

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
16	Відсутній	abs	Повертає абсолютне значення числа.	Приймає один аргумент - число.	Повертає абсолютне значення числа (додатне значення без знаку мінус)	builtins
17	Відсутній	int	Перетворює об'єкт на ціле число.	Приймає один аргумент, яким може бути рядок, число з плаваючою комою або інше число	Повертає ціле число, отримане з переданого аргументу	builtins
18	Відсутній	str	Перетворює об'єкт на рядкове значення.	Приймає один аргумент, яким може бути будь-який об'єкт	Повертає рядкове значення, отримане з переданого аргументу	builtins

Продовження Таблиці 4.1

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
19	Відсутній	open	Відкриває файл у вказаному режимі і повертає об'єкт файлу для подальшої роботи з ним	filename - рядок, що містить ім'я або шлях до файлу; mode (за замовченням "r") - режим відкриття файлу; encoding (опціональний) - кодування файлу.	Повертає об'єкт файлу	builtins
20	file	write	Записує рядок або дані до відкритого файлу.	рядком або байтові дані	Відсутні	builtins
21	file	close	Закриття файлу та звільнення ресурсів, пов'язаних із зчитуванням або записом у файл.	Відсутні	Відсутні	builtins

Продовження Таблиці 4.1

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
22	random	randint	Генерує випадкове ціле число в заданому діапазоні	a - мінімальне значення діапазону; b - максимальне значення діапазону	згенероване випадкове ціле число з діапазону [a, b]	random
23	random	sample	Генерує випадкову вибірку елементів з вказаного джерела без повторень.	population - джерело, з якого буде вибрана вибірка; k - розмір вибірки	Список розміру k, що містить випадкову вибірку елементів з джерела без повторень	random
24	Priority Queue	put	Додає елемент до пріоритетної черги з урахуванням пріоритету	item - елемент; priority - пріоритет елементу	Відсутні	queue
25	Priority Queue	get	Повертає та видаляє елемент з найвищим пріоритетом з пріоритетної черги	Відсутні	елемент з найвищим пріоритетом	queue

Продовження Таблиці 4.1

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
26	QApplication	exec	Запуск основного циклу обробки подій для додатку	Відсутні	Відсутні	PyQt5.QtWidgets
27	QWidget	show	Відображення вікна або віджету на екрані.	Відсутні	Відсутні	PyQt5.QtWidgets
28	Відсутній	setFixedSize	Встановлює фіксований розмір вікна або віджета.	width - ширина, в пікселях; height - висота, в пікселях	Відсутні	PyQt5.QtWidgets
28	Відсутній	setStyleSheet	Встановлює стилізацію (CSS) для віджета.	styleSheet - рядок, що містить правила стилів у форматі CSS	Відсутні	PyQt5.QtWidgets
30	Відсутній	setAlignment	Встановлює вирівнювання вмісту віджета	alignment (Qt.AlignmentFlag або int)	Відсутні	PyQt5.QtWidgets
31	Відсутній	setText	Встановлює текст для віджета, якій підтримує відображення тексту.	text (str) - текст, який потрібно встановити для віджета	Відсутні	PyQt5.QtWidgets

Продовження Таблиці 4.1

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголо-вний файл
32	Відсут-ній	font	Встановлює шрифт для віджета або отримує поточний шрифт віджета	font (QFont або QFontInfo) - шрифт, який потрібно встановити для віджета	font (QFont) - поточний шрифт	PyQt5.QtGui
33	Відсут-ній	SetPointSize	Встановлює розмір шрифту для віджета	size (int) - розмір шрифту	Відсутні	PyQt5.QtGui
34	Відсут-ній	setFont	Встановлює шрифт для віджета	font (QFont або QFontInfo) - шрифт, який потрібно встановити	Відсутні	PyQt5.QtWidgets
35	QLayout	setSpacing	Встановлює відступ між елементами віджета	spacing (int) - значення відступу	Відсутні	PyQt5.QtWidgets

Продовження Таблиці 4.1

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
36	QLayout	addLayout	Додає вкладений контейнерний об'єкт з розміщенням до поточного контейнерного об'єкта	layout (QLayout) - контейнерний об'єкт з розміщенням; stretch (int) - коефіцієнт розтягування	Відсутні	PyQt5.QtWidgets
37	QLayout	addWidget	Додає віджет до поточного контейнерного об'єкта	widget (QWidget) - віджет, який потрібно додати; stretch (int) - коефіцієнт розтягування	Відсутні	PyQt5.QtWidgets
38	QLayout	itemAt	Повертає вказівник на віджет, який знаходиться на певній позиції в контейнері	x (int) - горизонтальна позиція; y (int) - вертикальна позиція	Вказівник на віджет або None	PyQt5.QtWidgets
39	Відсутній	setWindowTitle	Встановлює заголовок вікна або віджета.	title (str) - заголовок, який потрібно встановити	Відсутні	PyQt5.QtWidgets

Продовження Таблиці 4.1

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
40	Відсутній	accept	Прийняти дію або результат	Відсутні	Відсутні	PyQt5.QtWidgets
41	Відсутній	reject	Відхилити дію або результат	Відсутні	Відсутні	PyQt5.QtWidgets
42	Відсутній	connect	Підключити обробник події до сигналу	Обробник події: Функція або метод, який буде викликатись при спрацюванні сигналу clicked	Відсутні	PyQt5.QtWidgets
43	QLineEdit	text	Отримати текстове значення елементу інтерфейсу	Відсутні	Значення тексту елементу інтерфейсу	PyQt5.QtWidgets
44	QRadioButton	isChecked	Перевірити, чи вибрано прапорець або прапорцівий перемикач	Відсутні	Логічне значення, що показує, чи вибрано прапорець	PyQt5.QtWidgets

Продовження Таблиці 4.1

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголо-вний файл
45	QTimer	singleShot	Запускає виконання функції або методу один раз з затримкою	delay (int) - затримка у мс; member - функція або метод, який буде виконаний після закінчення затримки	Відсутні	PyQt5.QtCore
46	QApplication	processEvents	Обробляє всі події, що знаходяться у черзі подій QApplication	Відсутні	Відсутні	PyQt5.QtWidgets

4.2.2. Користувацькі методи

У Таблиці 4.2 наведено користувацькі методи, розроблені в програмному забезпеченні

Таблиця 4.2 - Користувацькі методи

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголо-вний файл
1	PuzzleElement	update_value	Оновлення значення в елементі контейнеру	state - нове значення	Відсутні	gui

Продовження Таблиці 4.2

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголо-вний файл
2	Puzzle Layout	update_ layout	Оновлення значень усіх елементів контейнеру	arr - масив з новим значеннями	Відсутні	gui
3	Puzzle Layout	setSize	Зміна розміру усіх елементів контейнеру	size(int) - новий розмір елементів контейнеру в пікселях	Відсутні	gui
4	InputP uzzleEl ement	check_ value	Перевіряє чи значення в комірці вводу є коректним для задачі	Відсутні	Логічне значення, що позначає чи коректні вхідні дані	gui
5	InputP uzzleEl ement	get_ value	Повертає значення з комірки, якщо воно коректне	Відсутні	Строка зі введеним значенням або None	gui
6	PuzzleI nputLa yout	setWhit e	Змінює колір усіх елементів на білий	Відсутні	Відсутні	gui

Продовження Таблиці 4.2

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
7	PuzzleInputLayout	get_puzzle_values	Повертає значення зчитані з елементів контейнеру	Відсутні	Повертає масив значень, якщо усі значення коректні та унікальні, або None	gui
8	PuzzleInputLayout	set_values	Встановлює значення для усіх елеменів з контейнера	Масив з даними, які встановлюються в елементи контейнера	Відсутні	gui
9	PuzzleInputDialog	classic_end	Встановлює в контейнері кінцевого стану класичну кінцеву конфігурацію	Відсутні	Відсутні	gui
10	PuzzleInputDialog	validate_input	Перевіряє чи є введені дані коректними і, якщо так, закриває вікно вводу	Відсутні	Відсутні	gui

Продовження Таблиці 4.2

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
11	PuzzleInputDialog	get_puzzle_input	Повертає введені значення початкового та кінцевого станів	Відсутні	Масиви з початковим та кінцевим станами	gui
12	MainWindow	start_button_clicked	Запускає пошук рішення заданої задачі та виведення знайденого шляху розв’язку	Відсутні	Відсутні (шлях розв’язку змінюється під час виконання методу)	gui
13	MainWindow	display_solution	Покроково виводить у контейнер розв’язку стани головоломки, які є в шляху розв’язку	масив зі шляхом розв’язку задачі	Відсутні	gui
14	MainWindow	update_puzzle	Оновлює значення у контейнері розв’язку	масив зі шляхом розв’язку задачі; поточний стан в розв’язку	Відсутні	gui

Продовження Таблиці 4.2

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
15	MainWindow	enter_button_clicked	Запуск вікна введення початкового і кінцевого станів, а також отримання даних з цього вікна	Відсутні	Відсутні (початковий і кінцевий стани змінюються під час виконання методу)	gui
16	MainWindow	random_button_clicked	Генерація випадкових початкового і кінцевого станів для задачі та їх виведення у спеціальному вікні	Відсутні	Відсутні (початковий і кінцевий стани змінюються під час виконання методу)	gui
17	MainWindow	str_puzzle	Перетворює поточний крок шляху розв’язання на строку	i(int) - номер поточного кроку	Стока, що зображає стан головоломки	gui
18	MainWindow	file_button_clicked	Запис знайденого шляху розв’язку у файл	Відсутні	Відсутні	gui

Продовження Таблиці 4.2

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголо-вний файл
19	Puzzle	is_ solvable	Перевіряє чи є задача розв’язною	Відсутні	Логічне значення, яке позначає чи розв’язна задача	puzzle
20	Puzzle	generate _heurist ic	Розрахувати для стану головоломки евристику	Відсутні	Відсутні (значення евристики змінює-ється під час виконання методу)	puzzle
21	Puzzle	goal_ test	Перевіряє чи відповідає стан головоломки кінцевому	Відсутні	Логічне значення, яке позначає чи відповідає значення кінцевому стану	puzzle

Продовження Таблиці 4.2

№	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів	Заголовний файл
22	Puzzle	find_ legal_ actions	Знайти усі можливі кроки для стану головоломки	i(int) - вертикальне розташування пустого місця; j(int) - горизонтальне розташування пустого елементу	список можнивих кроків	puzzle
23	Puzzle	critical_ optimization	Прибирає зі списку можливих кроків крок, який є протилежним попередньому	legal_actions - список можливих кроків	повертає оновлене значення списку legal_actions	puzzle
24	Puzzle	generate_children	Генерує нащадків для стану головоломки	Відсутні	масив з нащадками головоломки	puzzle
25	Puzzle	find_ solution	Відновлює шлях пошуку розв’язку головоломки від кінцевого стану до початкового	Відсутні	масив зі станами усіх кроків розв’язку головоломки від початкового стану до кінцевого	puzzle

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1. План тестування

Перед перевіркою коректності роботи програми складемо план проведення тестів.

а) Тестування валідності введених значень.

1) Тестування при введенні в головоломку некоректного символу.

2) Тестування при введенні в головоломку некоректного числа.

3) Тестування при введенні однакових чисел в головоломку.

б) Тестування початку виконання задач без наявних даних.

1) Тестування при відсутності початкового і кінцевого станів.

2) Тестування при відсутності обраного алгоритму.

3) Тестування запису у файл при відсутності знайденого розв’язку.

в) Тестування перевірки на можливість виконання задачі.

г) Тестування коректності роботи методу 1.

д) Тестування коректності роботи методу 2.

5.2. Приклади тестування

Для перевірки коректності роботи програми проведемо ряд тестів, користуючись планом тестування (пункт 5.1).

У таблицях 5.1 - 5.3 наведено результати тестування програми при введені некоректних вхідних даних.

У програмі передбачено перетворення порожньої клітинки на елемент зі значенням « ».

Таблиця 5.1. Тестування при введенні в головоломку некоректного символу.

Мета тесту	Перевірити можливість вводу некоректних вхідних даних
Початковий стан програми	Відкрите вікно введення початкового та кінцевого станів головоломки
Вхідні дані	3 2 1 4 5 r 7 8 « » 1 2 3 4 5 6 7 8 « »
Схема проведення тесту	Поелементне ручне заповнення початкового стану. Заповнення кінцевого стану програмно після натискання кнопки «Класична кінцева розстановка». Натискання кнопки «ОК»
Очікуваний результат	Позначення некоректного елементу та не прийняття вводу
Стан програми після проведення випробувань	Клітинки з некоректним вводом позначено червоним, вікно для вводу відкрите

Таблиця 5.2. Тестування при введенні в головоломку некоректного числа.

Мета тесту	Перевірити можливість вводу некоректних вхідних даних
Початковий стан програми	Відкрите вікно введення початкового та кінцевого станів головоломки
Вхідні дані	3 2 1 90 5 6 7 8 « » 1 2 3 4 5 6 7 8 « »
Схема проведення тесту	Поелементне ручне заповнення початкового стану. Заповнення кінцевого стану програмно після натискання кнопки «Класична кінцева розстановка». Натискання кнопки «ОК»
Очікуваний результат	Позначення некоректного елементу та не прийняття вводу
Стан програми після проведення випробувань	Клітинки з некоректним вводом позначено червоним, вікно для вводу відкрите

Таблиця 5.3. Тестування при введенні однакових чисел в головоломку.

Мета тесту	Перевірити можливість вводу некоректних вхідних даних
Початковий стан програми	Відкрите вікно введення початкового та кінцевого станів головоломки
Вхідні дані	3 2 1 3 5 6 7 8 « » 1 2 3 4 5 6 7 8 « »
Схема проведення тесту	Поелементне ручне заповнення початкового стану. Заповнення кінцевого стану програмно після натискання кнопки «Класична кінцева розстановка». Натискання кнопки «ОК»

Продовження Таблиці 5.3

Очікуваний результат	Позначення некоректного елементу та не прийняття вводу
Стан програми після проведення випробувань	Клітинки з некоректним вводом позначено червоним, вікно для вводу відкрите

У таблицях 5.4 - 5.6 наведено результати тестування початку виконання задачі без наявних даних.

Таблиця 5.4. Тестування при відсутності початкового і кінцевого станів.

Мета тесту	Перевірити реакцію програми на відсутність вхідних даних
Початковий стан програми	Відкрите головне вікно програми. Початковий та кінцевий стани головоломки не обрані.
Вхідні дані	-
Схема проведення тесту	Початковий та кінцевий стан не введені жодним способом. Натискання кнопки «Вирішити задачу»
Очікуваний результат	Повідомлення про помилку
Стан програми після проведення випробувань	Видано помилку «Помилка! Оберіть початковий і кінцевий стани»

Таблиця 5.5. Тестування при відсутності обраного алгоритму.

Мета тесту	Перевірити реакцію програми на відсутність обраного методу виконання задачі
Початковий стан програми	Відкрите головне вікно програми. Алгоритм для розв’язання головоломки не обрано.
Вхідні дані	-
Схема проведення тесту	Алгоритм не обрано. Натискання кнопки «Вирішити задачу»
Очікуваний результат	Повідомлення про помилку
Стан програми після проведення випробувань	Видано помилку «Помилка! Оберіть один з алгоритмів»

Таблиця 5.6. Тестування запису у файл при відсутності знайденого розв’язку.

Мета тесту	Перевірити реакцію програми на відсутність розв’язку під час запису в файл
Початковий стан програми	Відкрите головне вікно програми. Шлях не було знайдено або кнопка «Вирішити задачу» не натискалась.
Вхідні дані	-
Схема проведення тесту	Не здійснено натискання кнопки «Вирішити задачу». Натискання кнопки «Записати в файл»
Очікуваний результат	Повідомлення про помилку
Стан програми після проведення випробувань	Видано помилку «Помилка! Жодного розв'язку знайдено не було»

У таблиці 5.7 наведено результати тестування програми при введені головоломки, що не має розв’язку.

Таблиця 5.7. Тестування перевірки на можливість виконання задачі.

Мета тесту	Перевірити реакцію програми нерозв’язні вхідні дані
Початковий стан програми	Відкрите головне вікно програми. Введено початковий та кінцевий стани. Обрано алгоритм A*
Вхідні дані	1 2 3 5 4 6 7 8 « » 1 2 3 4 5 6 7 8 « »
Схема проведення тесту	Поелементне ручне заповнення початкового стану. Заповнення кінцевого стану програмно після натискання кнопки «Класична кінцева розстановка». Натискання кнопки «ОК» Обрано радіокнопку «A*» Натискання кнопки «Вирішити задачу»
Очікуваний результат	Повідомлення про неможливість виконання задачі
Стан програми після проведення випробувань	Видано повідомлення « Задача не має розв'язку »

У таблицях 5.8 - 5.9 наведено результати тестування коректності роботи алгоритмів A* та RBFS.

Таблиця 5.8. Тестування роботи алгоритму A*.

Мета тесту	Перевірити коректність роботи алгоритму A*
Початковий стан програми	Відкрите головне вікно програми. Введено початковий та кінцевий стани. Обрано алгоритм A*
Вхідні дані	1 3 4 8 6 2 7 « » 5 1 2 3 8 « » 4 7 6 5
Схема проведення тесту	Поелементне ручне заповнення початкового та кінцевого станів. Натискання кнопки «ОК» Обрано радіокнопку «A*» Натискання кнопки «Вирішити задачу»
Очікуваний результат	Виведення такого шляху розв'язку: 1 3 4 8 6 2 7 « » 5 1 3 4 8 « » 2 7 6 5 1 3 4 8 2 « » 7 6 5 1 3 « » 8 2 4 7 6 5 1 « » 3 8 2 4 7 6 5 1 2 3 8 « » 4 7 6 5
Стан програми після проведення випробувань	У полі для виведення головоломки з перервою в 1 с змінюються такі стани: 1 3 4 8 6 2 7 « » 5 1 3 4 8 « » 2 7 6 5 1 3 4 8 2 « » 7 6 5 1 3 « » 8 2 4 7 6 5 1 « » 3 8 2 4 7 6 5 1 2 3 8 « » 4 7 6 5

Таблиця 5.9. Тестування роботи алгоритму RBFS.

Мета тесту	Перевірити коректність роботи алгоритму RBFS
Початковий стан програми	Відкрите головне вікно програми. Введено початковий та кінцевий стани. Обрано алгоритм RBFS
Вхідні дані	2 3 « » 1 4 5 7 8 6 4 1 3 7 2 5 8 « » 6
Схема проведення тесту	Поелементне ручне заповнення початкового та кінцевого станів. Натискання кнопки «ОК» Обрано радіокнопку «RBFS» Натискання кнопки «Вирішити задачу»
Очікуваний результат	Виведення такого шляху розв’язку: 2 3 « » 1 4 5 7 8 6 2 « » 3 1 4 5 7 8 6 2 4 3 1 « » 5 7 8 6 2 4 3 « » 1 5 7 8 6 « » 4 3 2 1 5 7 8 6 4 « » 3 2 1 5 7 8 6 4 1 3 2 « » 5 7 8 6 4 1 3 « » 2 5 7 8 6 4 1 3 7 2 5 « » 8 6 4 1 3 7 2 5 8 « » 6
Стан програми після проведення випробувань	У полі для виведення головоломки з перервою в 1 с змінюються такі стани: 2 3 « » 1 4 5 7 8 6 2 « » 3 1 4 5 7 8 6 2 4 3 1 « » 5 7 8 6 2 4 3 « » 1 5 7 8 6 « » 4 3 2 1 5 7 8 6 4 « » 3 2 1 5 7 8 6 4 1 3 2 « » 5 7 8 6 4 1 3 « » 2 5 7 8 6 4 1 3 7 2 5 « » 8 6 4 1 3 7 2 5 8 « » 6

6 ІНСТРУКЦІЯ КОРИСТУВАЧА

6.1. Робота з програмою

Для роботи з програмою достатньо завантажити файл 8PuzzleSolver.exe та запустити його. Відкриється головне вікно програми (Рисунок 6.1).

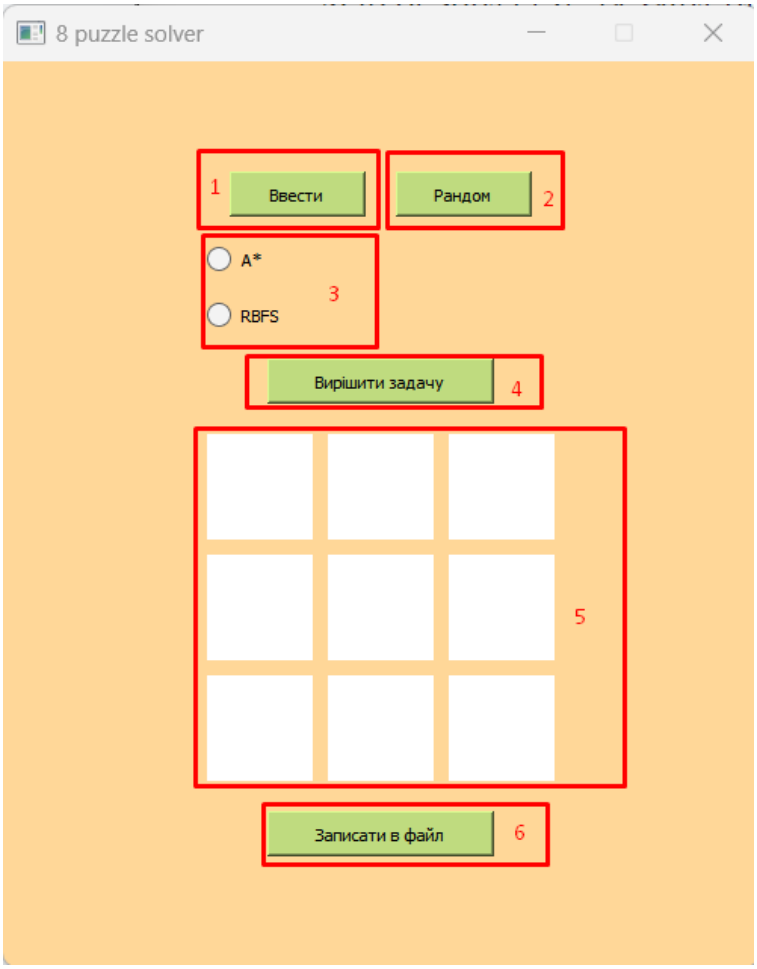


Рисунок 6.1. - Головне вікно програми

Вікно складається з таких елементів: кнопка для введення головоломки (1), кнопка для випадкової генерації розв'язної головоломки (2), радіокнопки для вибору методу розв'язку головоломки (3), кнопка запуску розв'язання головоломки (4), поле для виведення розв'язку головоломки (5), кнопка для запису розв'язку в файл (6).

При натисканні на кнопку «Ввести» відкривається вікно для введення початкового та кінцевого станів головоломки (Рисунок 6.2).

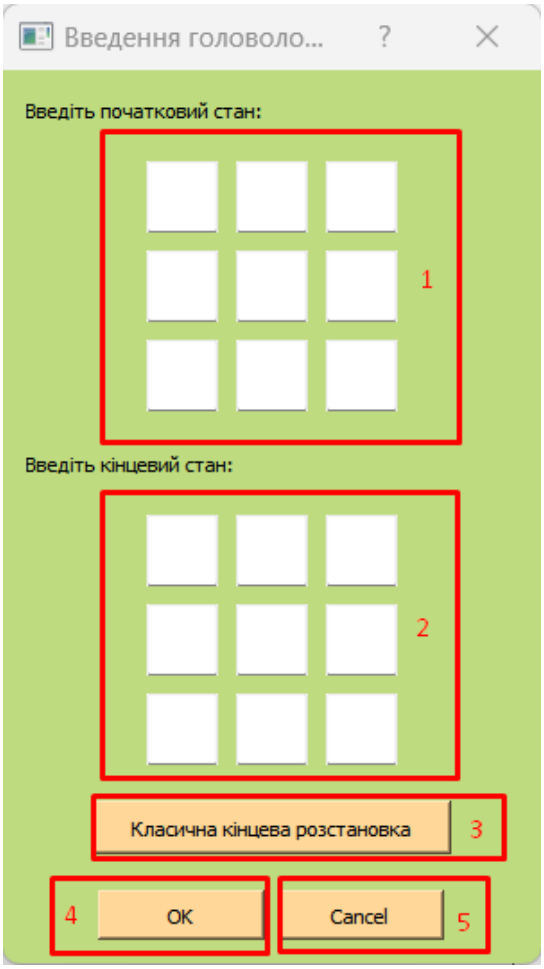


Рисунок 6.2. - Вікно введення головоломки

Це вікно складається з полів для введення початкового (1) та кінцевого (2) станів, кнопки для генерації класичного кінцевого стану (3), кнопки для підтвердження вводу (4) і для скасування вводу (5). У поля потрібно потрібно ввести початкову та кінцеву розстановки пазла. В клітинки, що відповідають фішкам, потрібно ввести відповідні номери від 1 до 8, а клітинку, що відповідає в розстановці порожньому місцю, залишити порожньою (Рисунок 6.3). Якщо кінцевий стан відповідає класичному (1 2 3 4 5 6 7 8 « »), то можна натиснути кнопку «Класична розстановка», щоб заповнити поле кінцевого стану класичним кінцевим станом задачі 8-puzzle (Рисунок 6.4). Після введення обох станів слід натиснути кнопку «ОК»,

якщо хочете підтвердити введення, або кнопку «Cancel», якщо хочете відмінити введення.

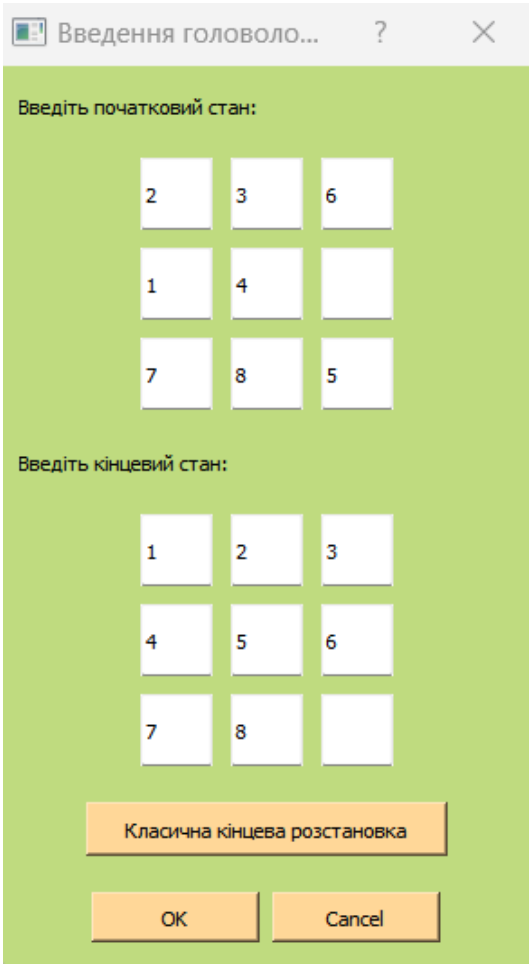


Рисунок 6.3. - Введення початкового стану

та автоматичне кінцевого стану класичною розстановкою

Якщо у введених даних є помилка, то при натисканні кнопки «ОК» відповідні клітинки з помилкою змінять колір на червоний, вікно введення не буде закрите та поточні початковий та кінцевий стани в системі не зміняться (Рисунок 6.5).

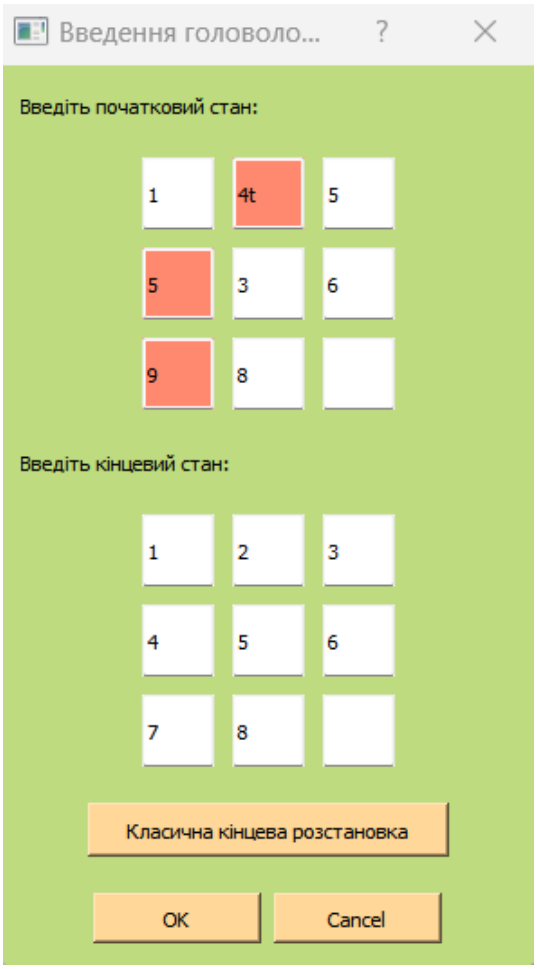


Рисунок 6.4. - Позначення клітинок з некоректним вводом

При натисканні на головному вікні кнопки «Рандом» буде згенеровано початковий та кінцевий стани розв’язної задачі та відкривається вікно для виведення цих станів (Рисунок 6.6).

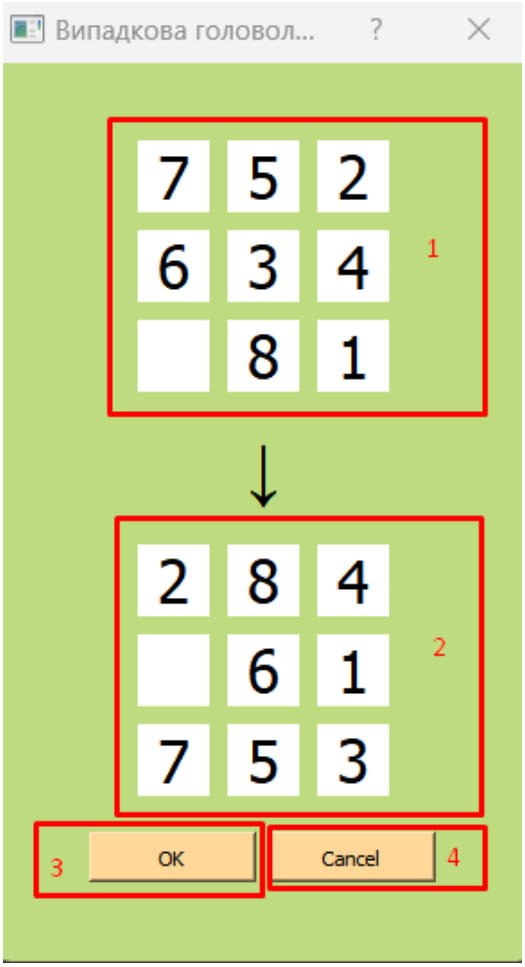


Рисунок 6.6. - Вікно виведення згенерованої головоломки

Вікно складається з полів для виведення початкового (1) та кінцевого (2) станів, кнопка для підтвердження генерації (4) і для скасування генерації (5).

Якщо хочете підтвердити згенеровані значення, то натискайте кнопку «ОК». Якщо хочете відмінити генерацію головоломки натискайте кнопку «Cancel».

Коли будь-яким з описаних вище способів було задано початковий та кінцевий стани, то у полі для виведення головоломки на головному вікні буде виведено початковий стан. Якщо ж встановлення станів було відмінене, то стан поля для виводу головоломки не зміниться.

Після визначення початкового та кінцевого станів потрібно визначити за допомогою якого алгоритму буде відбуватись розв’язання задачі (Рисунок 6.7). Радіокнопка «A*» відповідає за алгоритм A*, радіокнопка «RBFS» - RBFS.

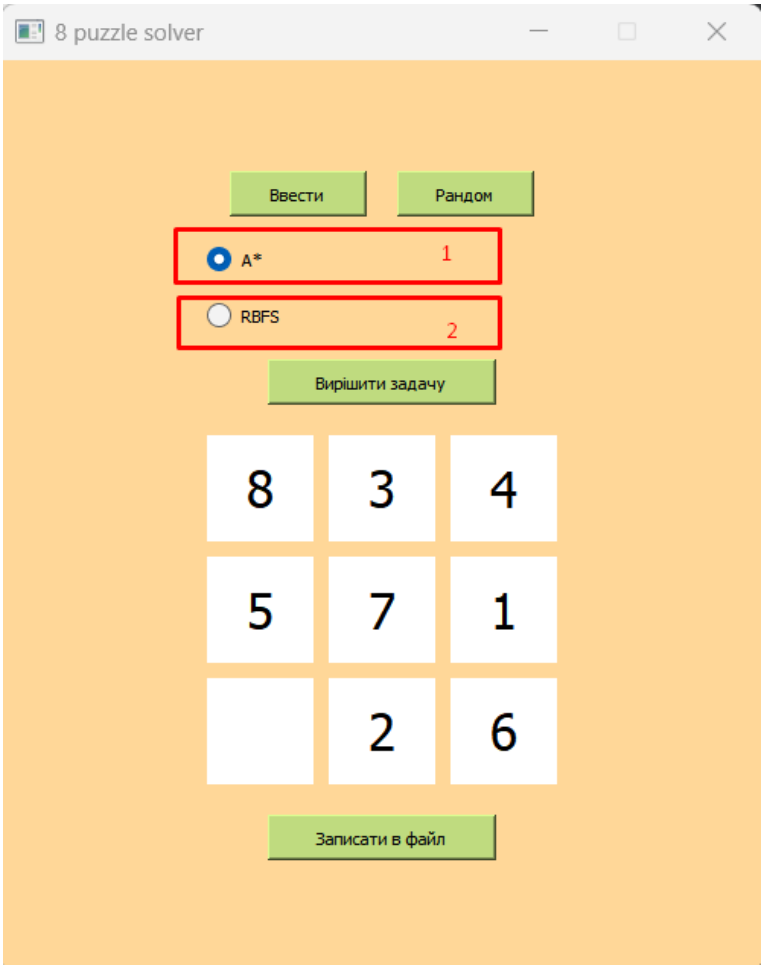


Рисунок 6.7. - Вибір алгоритму

Після того як вхідні дані та метод розв’язання було визначено потрібно натиснути на кнопку «Вирішити задачу». Це запустить алгоритм пошуку шляху розв’язання. Знайдений шлях буде виведено в поле для виведення розв’язку головоломки в головному вікні програми за допомогою анімації - кожен хід буде робитись з перервою в 1 секунду, після чого буде виведено повідомлення «Готово!» (Рисунки 6.8 - 6.13).

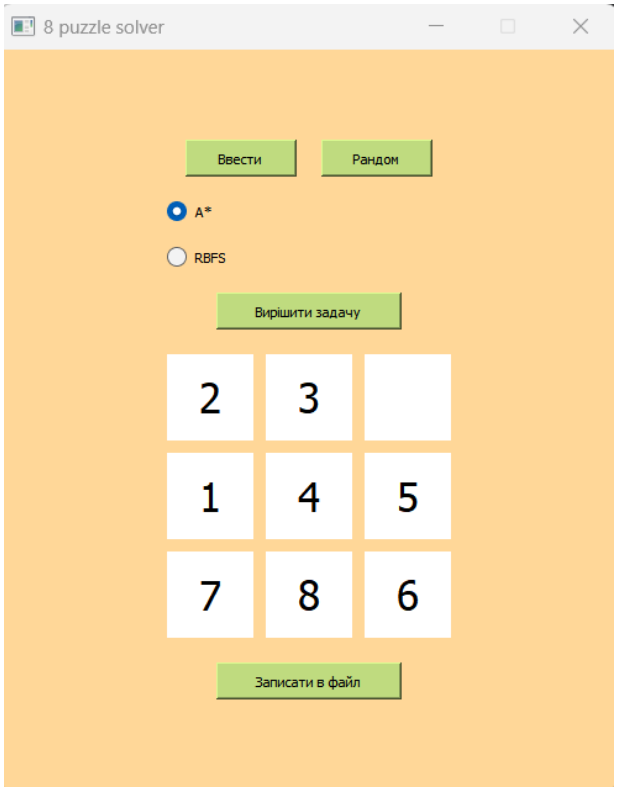


Рисунок 6.8. - Початкова розстановка

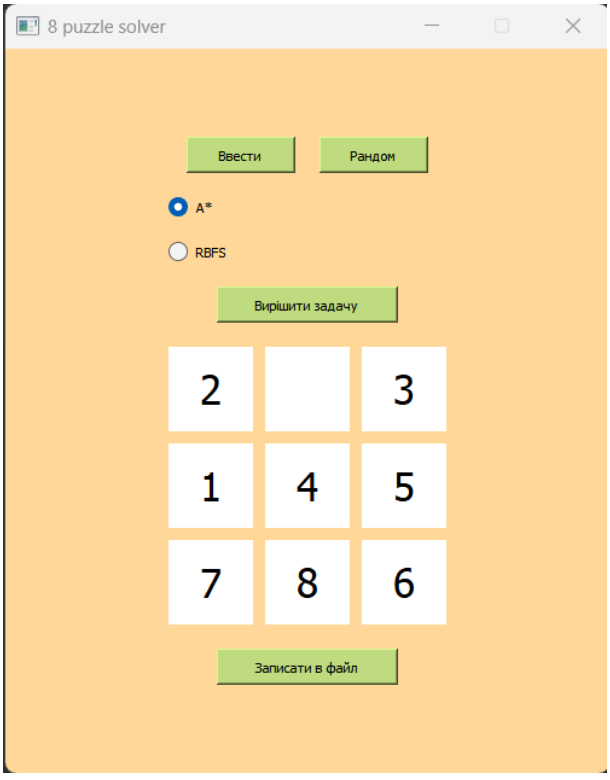


Рисунок 6.9. - Крок 1

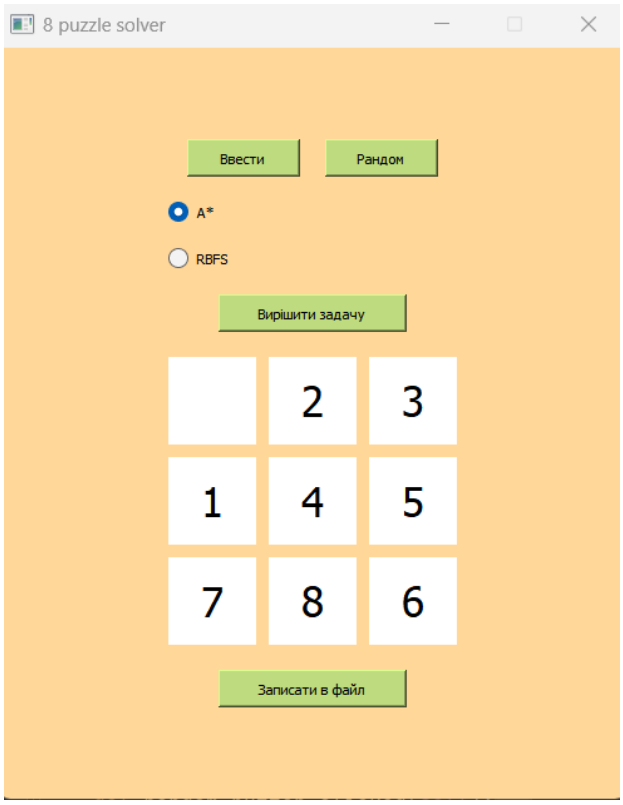


Рисунок 6.10. - Крок 2

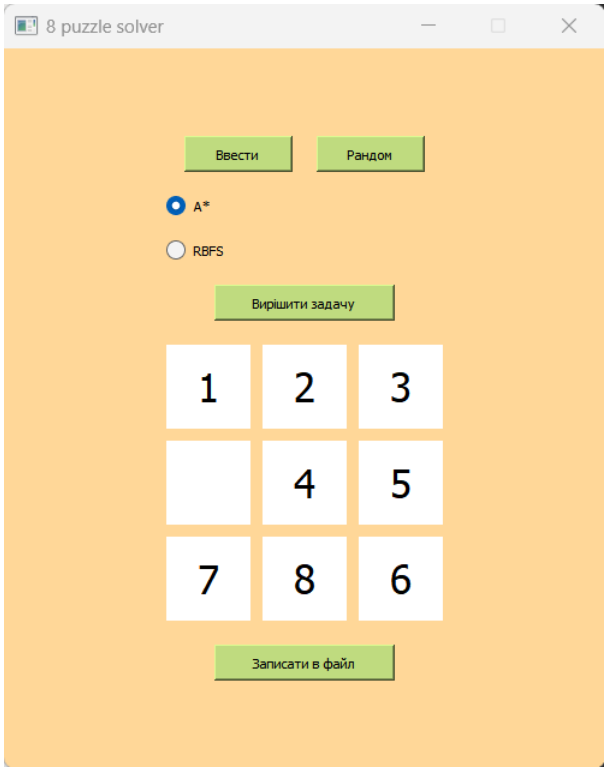


Рисунок 6.11. - Крок 3

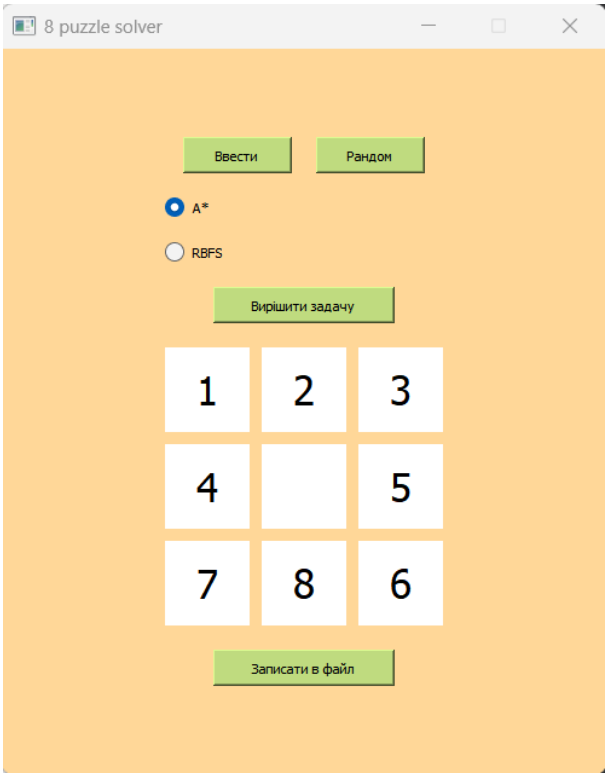


Рисунок 6.12. - Крок 4

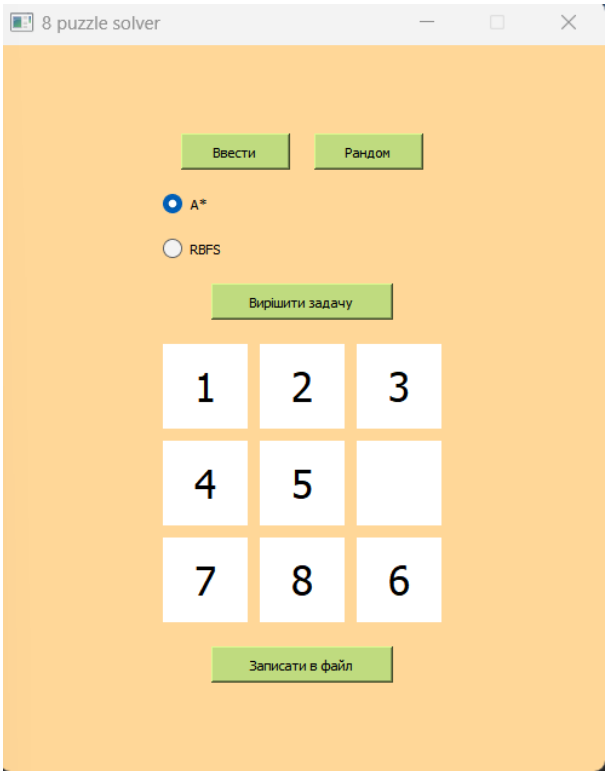


Рисунок 6.13. - Крок 5

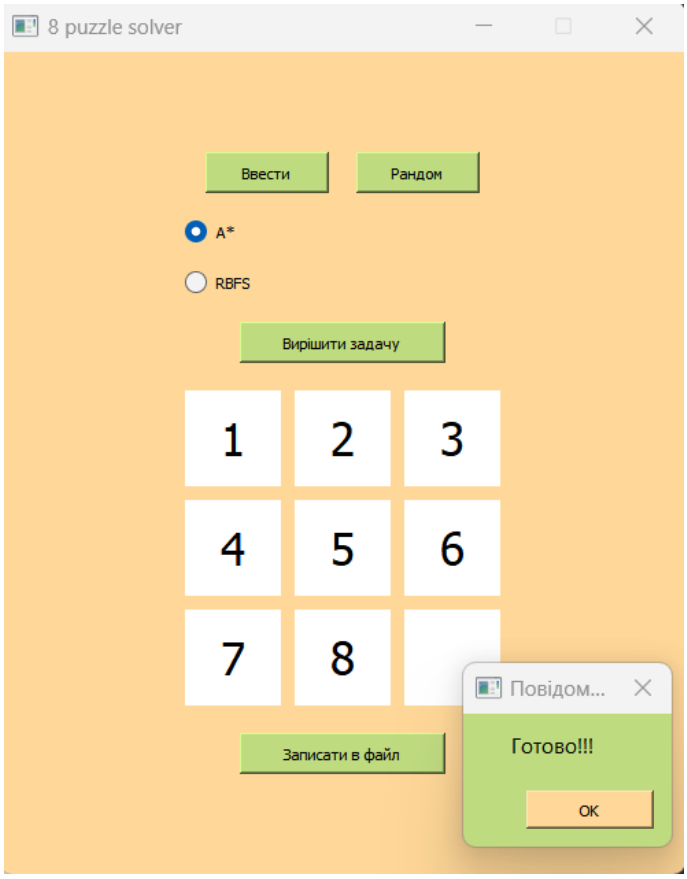


Рисунок 6.14. - Кінцева розстановка та повідомлення «Готово!»

За бажанням користувач може зберегти хід розв’язку в текстовий файл. Для цього необхідно натиснути на кнопку «Записати в файл». Після цього розв’язання буде записане у файл з назвою «solution.txt» та буде виведено повідомлення «Розв’язок записано у файл solution.txt».

6.2. Формат вхідних та вихідних даних

На вхід програми користувач подає початковий та кінцевий стани головоломки. Кожен елемент стану головоломки - це унікальне ціле число від 1 до 8 або символ « ».

Результатом виконання є шлях розв’язку задачі, а також кількість відкритих вузлів у дереві пошуку. Кожен елемент станів, що знаходяться в шляху розв’язку задачі, - це унікальне ціле число від 1 до 8 або символ « ».

6.3. Системні вимоги

Системні вимоги до програмного забезпечення наведені в Таблиці 6.1

Таблиця 6.1 - Системні вимоги до програмного забезпечення

	Мінімальні	Рекомендовані
Операційна система	Windows 10	Windows 10 (з останніми оновленнями) або новіші системи Windows
Процесор	Intel i5-1135g7 2.4 GHz або AMD Ryzen 5 4500U 2.38 GHz	Intel i3-2100 3.1 GHz або AMD Athlon X2 245 2.9 GHz
Оперативна пам'ять	100 MB RAM	200 MB RAM
Відеоадаптер	AMD Radeon Graphics 512 MB (або сумісний аналог)	
Дисплей	800x600	1024x768 або краще
Прилади введення	Клавіатура та комп’ютерна миша	

7 АНАЛІЗ І УЗАГАЛЬНЕННЯ РЕЗУЛЬТАТІВ

Основна задача цієї курсової роботи - реалізація програми для вирішення головоломки 8-puzzle за допомогою алгоритмів A* та RBFS.

Під час роботи програми не було помічено жодних критичних ситуацій. Помилки виникають при введені некоректних станів головоломки, а спроби куристувача почати певний процес без необхідних вхідних даних, а також при введені нерозв’язної задачі.

Для перевірки та доведення достовірності результатів виконання програми скористаюсь веб-сайтом 8-Puzzle Solver.

Для цього введемо в програму задачу. Яка зображена на Рисунку 7.1.

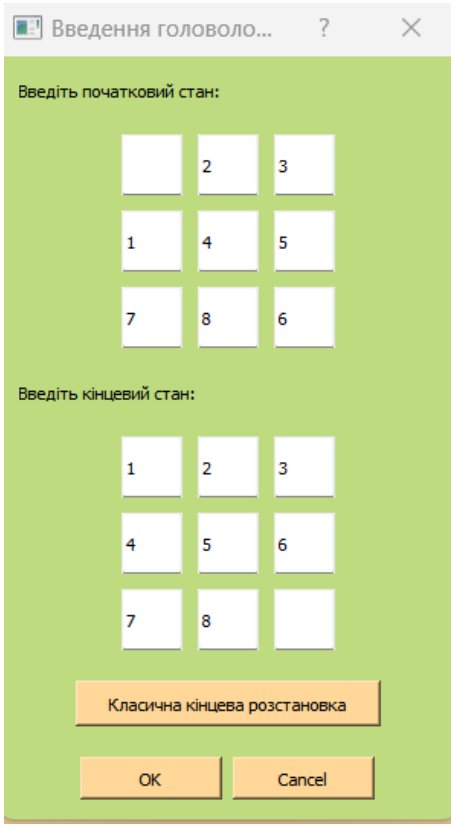


Рисунок 7.1. - Початкова та кінцева розстановки задачі

Спочатку розв’яжемо цю задачу за допомогою алгоритму A*. Хід розв’язку зображено на Рисунках 7.2 - 7.6.

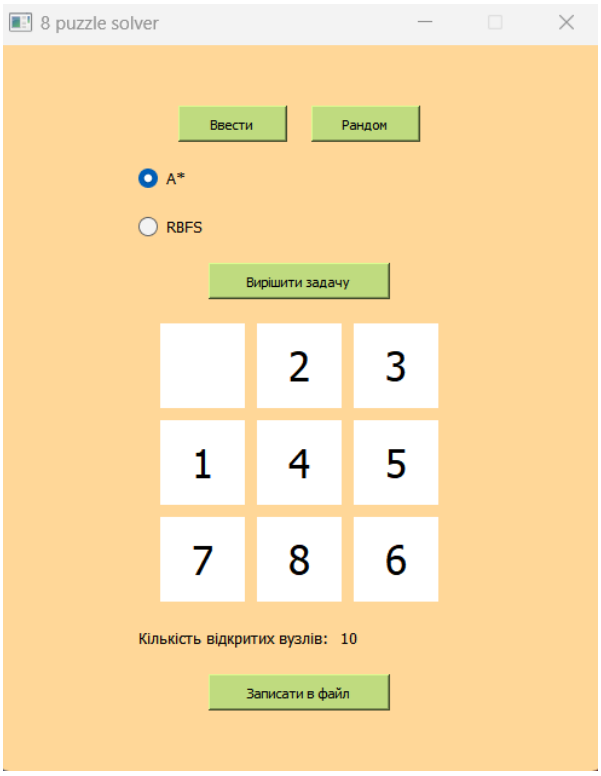


Рисунок 7.2 - Початковий стан при алгоритмі A*

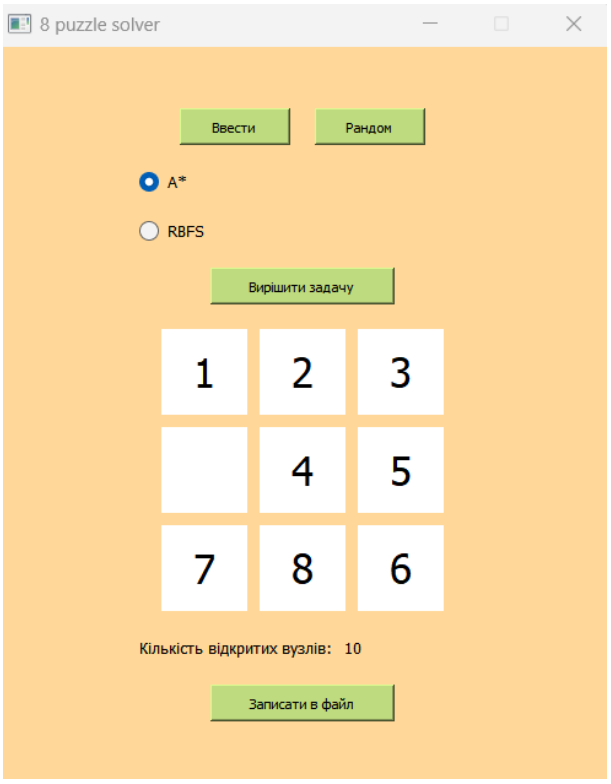


Рисунок 7.3 - Крок 1 при алгоритмі A*

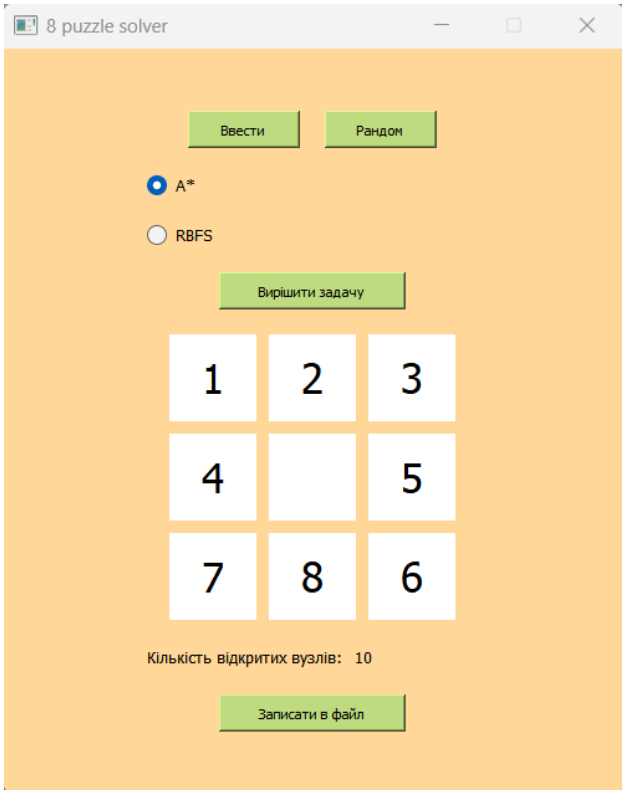


Рисунок 7.4 - Крок 2 при алгоритмі A*

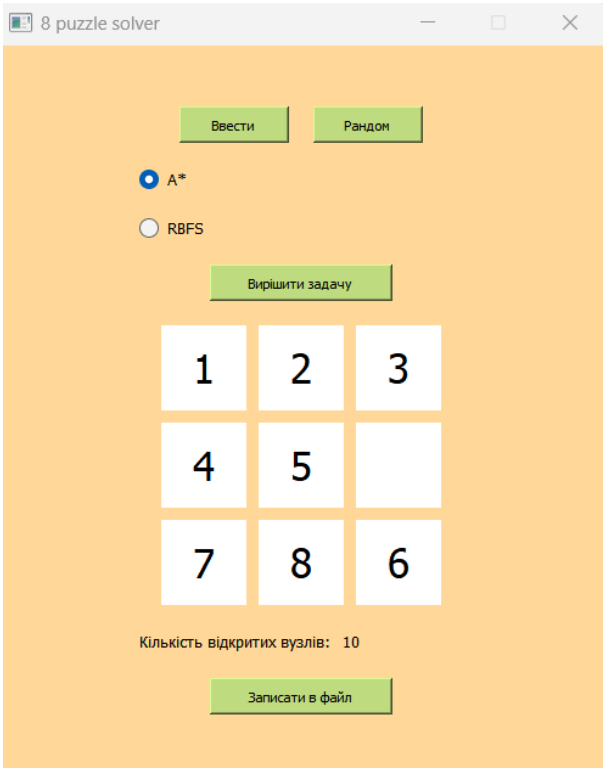


Рисунок 7.5 - Крок 3 при алгоритмі A*

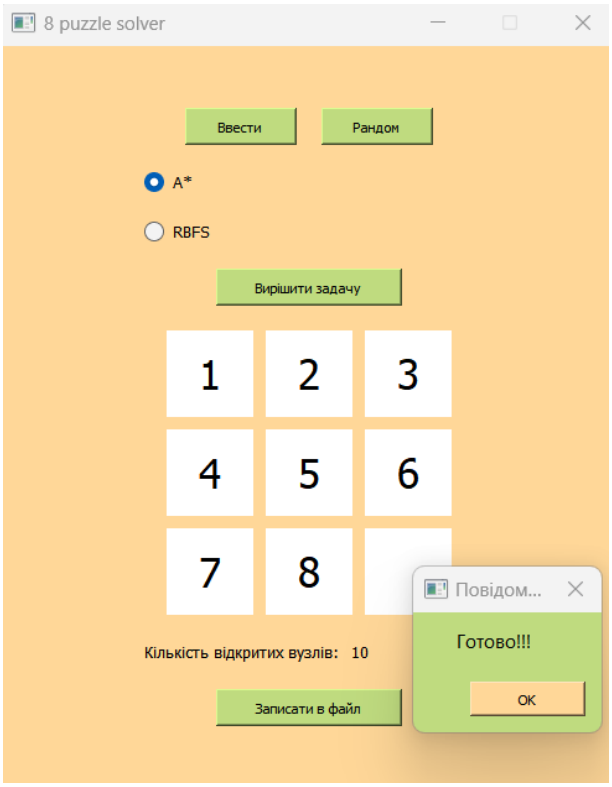


Рисунок 7.6 - Кінцевий результат при алгоритмі A*

Також розв’яжемо цю задачу за допомогою алгоритму RBFS. Хід розв’язку задачі за допомогою цього алгоритму зображено на Рисунках 7.7 - 7.11.

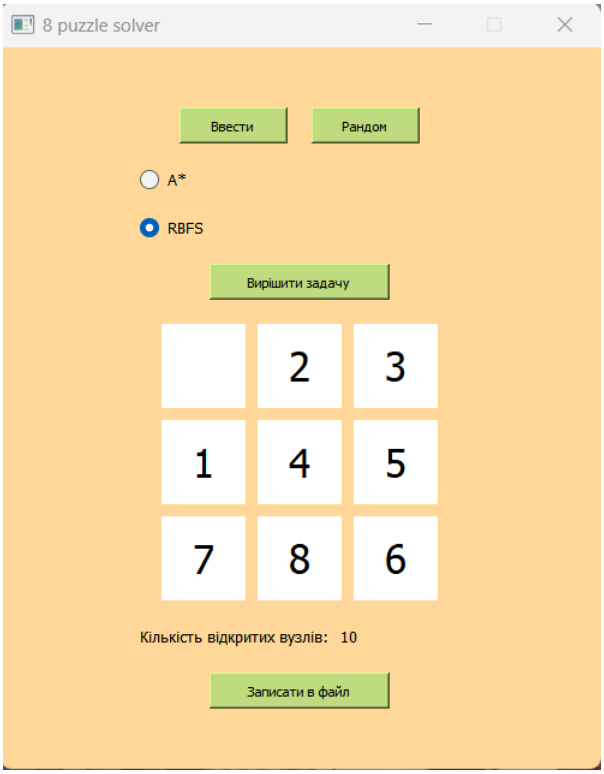


Рисунок 7.7 - Початковий стан при алгоритмі RBFS

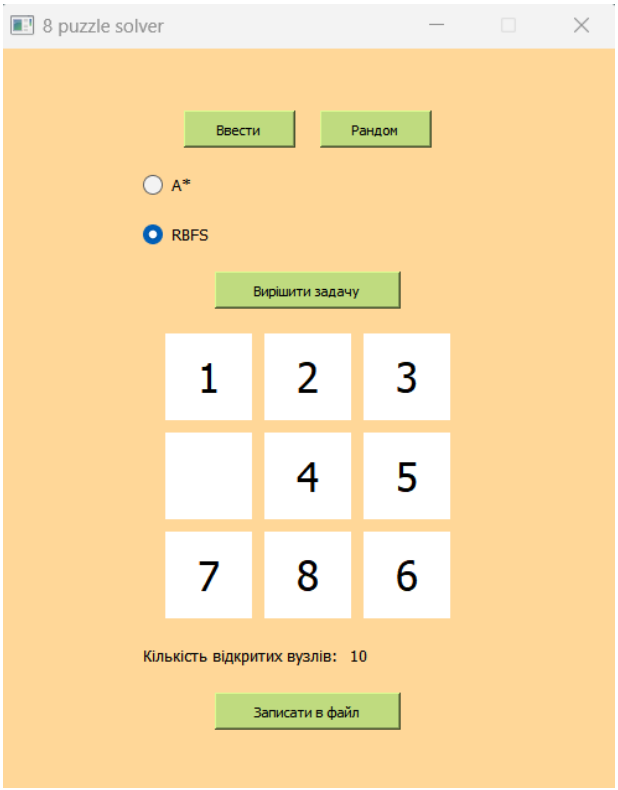


Рисунок 7.8 - Крок 1 при алгоритмі RBFS

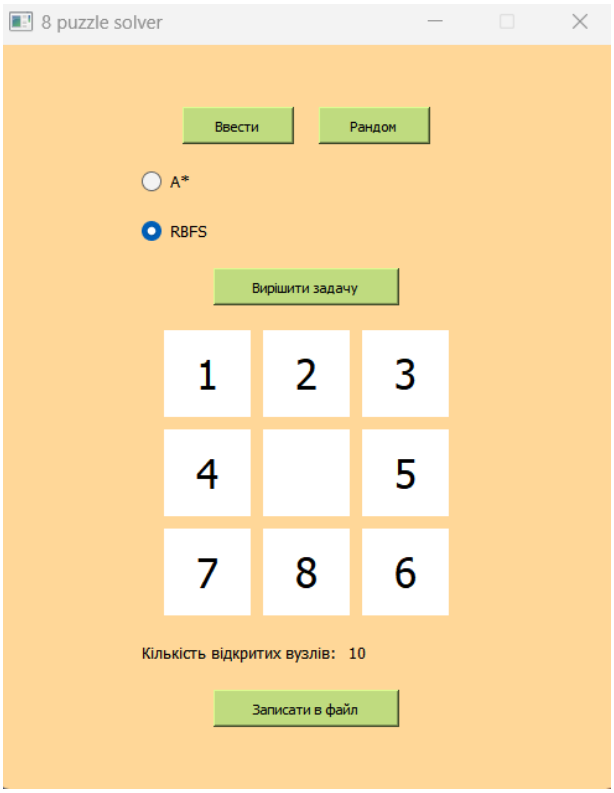


Рисунок 7.9 - Крок 2 при алгоритмі RBFS

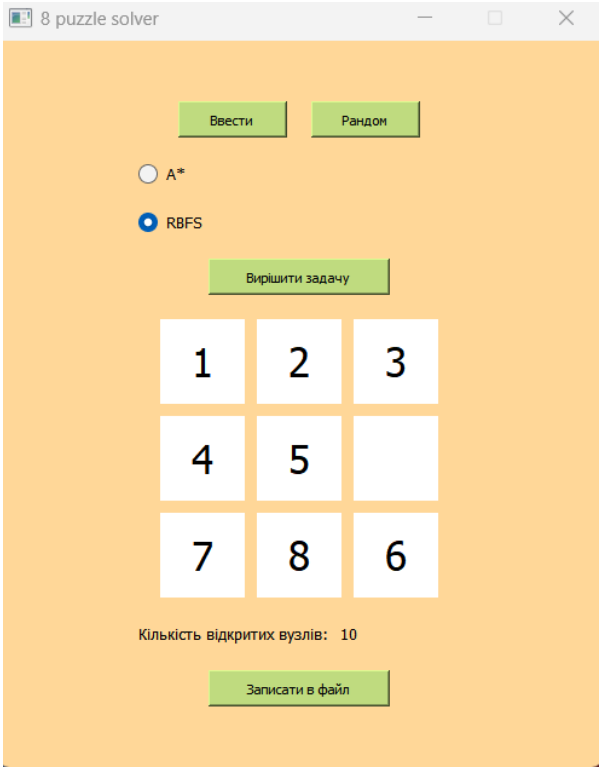


Рисунок 7.10 - Крок 3 при алгоритмі RBFS

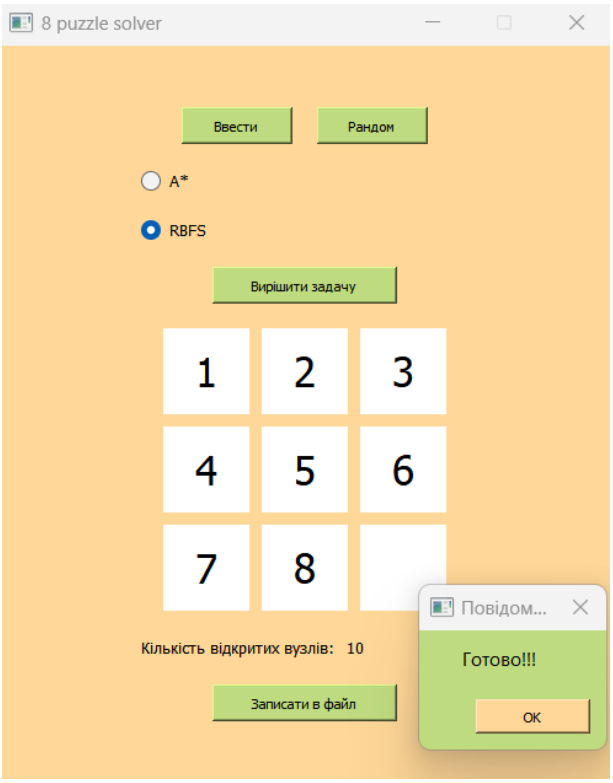


Рисунок 7.11 - Кінцевий стан при алгоритмі RBFS

Результати обох алгоритмів збігаються з розв’язанням цієї задачі на веб-сайті 8-Puzzle Solver (Рисунок 7.12), отже алгоритми працюють коректно.

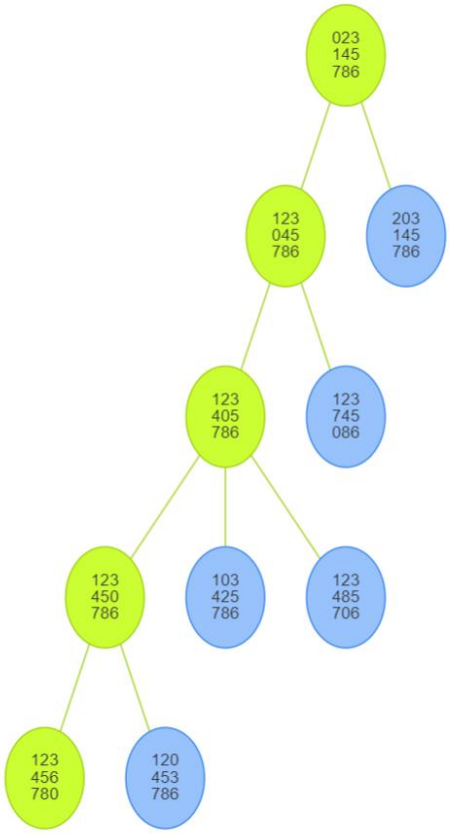


Рисунок 7.12 - Результат розв’язання задачі

на веб-сайті 8-Puzzle Solver

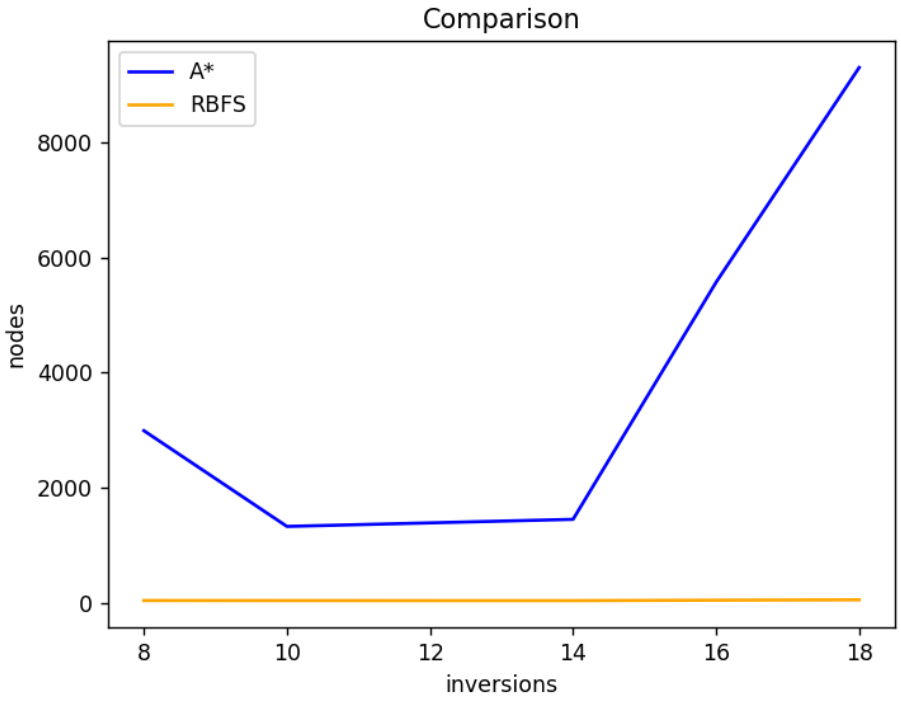
Для тестування ефективності алгоритмів, було згенеровано 5 випадкових початкових станів та класичний кінцевий стан.

Результати тестування ефективності алгоритмів наведено в Таблиці 7.1.

Таблиця 7.1 - Тестування ефективності методів

Кількість інверсій	Параметри тестування	Алгоритм	
		A*	RBFS
8	Кількість відкритих вузлів	2993	45
10	Кількість відкритих вузлів	1331	44
14	Кількість відкритих вузлів	1454	43
16	Кількість відкритих вузлів	5569	51
20	Кількість відкритих вузлів	9295	57

Візуалізація даних таблиці 7.1 наведено на Рисунку 7.13.



Рисунку 7.13 - Графік залежності кількості відкритих вузлів від кількості інверсій в початковому стані

Результати тестування спонукають до таких висновків:

1) У більшості випадків між кількістю інверсій та кількістю відкритих вузлів є залежність: чим більша кількість інверсій, тим більша кількість згенерованих вузлів, а отже і довший пошук.
2) Кількість відкритих вузлів у алгоритмі RBFS значно менша за кількість відкритих вузлів у A*. Це пов’язано з тим, що RBFS закриває піддерево, як тільки знайде більш ефективний шлях.

ВИСНОВКИ

Метою даної курсової роботи було створення програмного забезпечення з інтуїтивно зрозумілим графічним інтерфейсом, призначеного для розв'язання головоломки 8-puzzle з використанням алгоритмів A* та RBFS.

Основне завдання додатку було сформульовано в розділі "Постановка задачі". В 2 розділі було висвітлено принцип роботи обох алгоритмів, а в 3 розділі було наведено загальний принцип роботи програми та псевдокод обох алгоритмів. Розділ "Опис програмного забезпечення" містить діаграму класів, їх опис та опис методів. У 5 розділі програму було протестовано на різних значеннях, а також були досліджені реакції програми на ймовірні помилкові ситуації, спричинені неправильнім введенням, спробою запустити роботу програми без вхідних даних або за відсутності розв’язку задачі. У 6 розділі було наведено детальну інструкцію користувача для роботи з програмою. Аналіз і узагальнення результатів було проведено у 7 розділі курсової роботи.

Результатом виконання курсової роботи є якісне програмне забезпечення для розв'язання головоломки 8-puzzle. Програма на вибір користувача генерує вхідні дані або зчитує їх з графічного інтерфейса, перевіряючи коректність вводу та вказуючи на помилкові значення. Реалізовано обидва алгоритми пошуку розв’язку задачі, анімацію шляху розв’язку задачі та його запис у файл.

У результаті тестування програми помилок у реалізації не виявлено, програма чітко та коректно реагує на будь-які помилкові ситуації. Програма знаходить оптимальний розв’язок задачі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Документація до мови програмування Python. 2023. *Python 3.11.4 documentation.* URL: https://docs.python.org/3 (дата звернення 23.03.2023)
2. Документація до бібліотеки PyQt5. 2023. QT for Python. URL: https://doc.qt.io/qtforpython-6 (дата звернення 18.04.2023)
3. Рассел С. Дж., Норвиг, П. Искусственный интеллект: современный подход = Artificial Intelligence: A Modern Approach / Пер. с англ. и ред. К. А. Птицына. — 2-е изд.. — М.: Вильямс, 2006. — С. 157—162. — 3000 экз. Экз

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра
інформатики та програмної інженерії

Затвердив

Керівник Головченко М.М.

«6» березня 2023 р.

Виконавець:

Студент Семенова Є. О.

«6» червня 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: 8-puzzle

з дисципліни:

«Основи програмування»

Київ 2023

1. Мета: Метою курсової роботи є розробка якісного програмного забезпечення, яке реалізує головоломку “8-puzzle”.
2. Дата початку роботи: « 12 » березня 2023 р.
3. Дата закінчення роботи: «31» травня 2023 р.
4. Вимоги до програмного забезпечення.

1) Функціональні вимоги:
– Графічний інтерфейс
– Випадкова генерація коректного початкового стану в головоломці
– Вибір кінцевого стану користувачем
– Перевірка на правильність вводу
– Розв’язання задачі за допомогою методів A* та RBFS
– Вибір алгоритму розв’язання користувачем
– Відображення кроків, які необхідні для розв’язання
– Збереження результатів роботи у текстовий файл
– Відображення характеристик алгоритмів, які визначають їх практичну складність

2) Нефункціональні вимоги:
– Можливість використання програми на Windows 10 та Windows11.
– Реалізація програми на мові Python
– Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.
ГОСТ 19.106 - 78 - Вимоги до програмної документації.
ГОСТ 7.1 - 84 та ДСТУ 3008 - 2015 - Розробка технічної документації.

5. Стадії та етапи розробки:
1) Об'єктно-орієнтований аналіз предметної області задачі (до 12.03.2023 р.)
2) Об'єктно-орієнтоване проектування архітектури програмної системи (до 10.04.202_р.)
3) Розробка програмного забезпечення (до 08.05.2023 р.)
4) Тестування розробленої програми (до 19.05.2023 р.)
5) Розробка пояснювальної записки (до 25.05.2023 р.).
6) Захист курсової роботи (до 06.06.2023 р.).
6. Порядок контролю та приймання. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду програмного забезпечення

«8 Puzzle Solver»

(Найменування програми (документа))

USB-флешка

(Вид носія даних)

20 арк., 90 Мб

(Обсяг програми (документа), арк.)

студента групи ІП-22 І курсу
Семенової О. Є.

main.py

```
from gui import MainWindow
from PyQt5.QtWidgets import *

app = QApplication([])
window = MainWindow()
window.show()
app.exec()
```

gui.py

```
from PyQt5.QtCore import *
from PyQt5.QtGui import QFont
from PyQt5.QtWidgets import *
from random import sample
from random import randint

from astar import Astar
from puzzle import count_inversions
from rbfs import RBFS


class PuzzleElement(QLabel):
    def __init__(self, num):
        super().__init__()
        self.setFixedSize(70, 70)
        self.setStyleSheet("background-color: white;")
        self.setText(str(num))
```

```
        font = self.font()
        font.setPointSize(20)
        self.setFont(font)

        self.setAlignment(Qt.AlignCenter)

    def update_value(self, state):
        self.setText(str(state))


class PuzzleLayout(QGridLayout):
    def __init__(self, arr):
        super().__init__()
        self.setSpacing(10)
        self.setVerticalSpacing(10)
        self.__elements = []
        for i in range(9):
            element = PuzzleElement(arr[i])
            self.addWidget(element, i // 3, i % 3)
            self.__elements.append(element)

    def update_layout(self, arr):
        for i in range(9):
            self.__elements[i].update_value(arr[i])

    def setSize(self, size):
        for i in range(9):
            self.__elements[i].setFixedSize(size, size)
```

```
class InputPuzzleElement(QLineEdit):
    def __init__(self):
        super().__init__()
        self.setFixedSize(40, 40)
        self.setStyleSheet("background-color: white;")


    def __check_value(self):
        if (self.text().isdigit() and int(self.text()) < 9 and int(self.text()) != 0) or self.text() == "":
            self.setStyleSheet("background-color: white;")
            return True
        else:
            self.setStyleSheet("background-color: #FE896E;")
            return False


    def get_value(self):
        if self.__check_value():
            return self.text()
        else:
            return None



class PuzzleInputLayout(QGridLayout):
    def __init__(self):
        super().__init__()
        self.setSpacing(10)
        self.setVerticalSpacing(10)
```

```
    self.__elements = []
    for i in range(9):
        element = InputPuzzleElement()
        self.addWidget(element, i // 3, i % 3)
        self.__elements.append(element)

def setWhite(self):
    for element in self.__elements:
        element.setStyleSheet("background-color: white;")

def get_puzzle_values(self):
    values = []
    is_unique = True
    is_valid = True
    for element in self.__elements:
        value = element.get_value()
        if value is None:
            is_valid = False
            continue

        if (value != '' and int(value) in values) or (value == '' and ' ' in values):
            is_unique = False
            element.setStyleSheet("background-color: #FE896E;")
        values.append(int(value) if value.isdigit() else ' ')
    if is_unique and is_valid:
        return values
    else:
        return None
```

```
    def set_values(self, values):
        for i in range(9):
            self.__elements[i].setText(str(values[i]))


class PuzzleInputDialog(QDialog):
    __start_puzzle = None
    __end_puzzle = None

    def __init__(self):
        super().__init__()
        self.setFixedSize(QSize(300, 500))
        self.setStyleSheet("background-color: #BFDB7F;")
        self.setWindowTitle("Введення головоломки")
        layout = QVBoxLayout(self)

        labelStart = QLabel("Введіть початковий стан:")
        labelEnd = QLabel("Введіть кінцевий стан:")
        self.input_start_layout = PuzzleInputLayout()
        self.input_start_layout.setAlignment(Qt.AlignHCenter)
        self.input_end_layout = PuzzleInputLayout()
        self.input_end_layout.setAlignment(Qt.AlignHCenter)

        classic_endButton = QPushButton("Класична кінцева розстановка")
        classic_endButton.setStyleSheet("background-color: #FFD798;")
        classic_endButton.setFixedSize(QSize(200, 30))
        classic_endButton.clicked.connect(self.__classic_end)
```

```
        button_box = QDialogButtonBox(QDialogButtonBox.Ok | QDialogButtonBox.Cancel)
        button_box.setStyleSheet("background-color: #FFD798;")
        button_box.accepted.connect(self.__validate_input)
        button_box.rejected.connect(self.reject)

        layout.setSpacing(20)
        layout.addWidget(labelStart)
        layout.addLayout(self.input_start_layout)
        layout.addWidget(labelEnd)
        layout.addLayout(self.input_end_layout)
        layout.addWidget(classic_endButton)
        layout.itemAt(4).setAlignment(Qt.AlignHCenter)
        layout.addWidget(button_box)
        layout.itemAt(5).setAlignment(Qt.AlignHCenter)

    def __classic_end(self):
        self.input_end_layout.set_values([1, 2, 3, 4, 5, 6, 7, 8, ''])

    def __validate_input(self):
        self.input_start_layout.setWhite()
        self.input_end_layout.setWhite()
        self.__start_puzzle = self.input_start_layout.get_puzzle_values()
        self.__end_puzzle = self.input_end_layout.get_puzzle_values()
        if self.__start_puzzle is not None and self.__end_puzzle is not None:
            self.accept()

    def get_puzzle_input(self):
        return self.__start_puzzle, self.__end_puzzle
```

```
class RandomPuzzleDialog(QDialog):
    def __init__(self, start, end):
        super().__init__()
        self.setFixedSize(QSize(300, 500))
        self.setStyleSheet("background-color: #BFDB7F;")
        self.setWindowTitle("Випадкова головоломка")
        layout = QVBoxLayout(self)
        layout.setSpacing(20)
        layout.setAlignment(Qt.AlignVCenter)

        layout.addLayout(PuzzleLayout(start))
        layout.itemAt(0).setSize(40)
        layout.itemAt(0).setAlignment(Qt.AlignHCenter)

        arrow = QLabel("↓")
        font = QFont("Arial", 24)
        arrow.setFont(font)
        layout.addWidget(arrow)
        layout.itemAt(1).setAlignment(Qt.AlignHCenter)

        layout.addLayout(PuzzleLayout(end))
        layout.itemAt(2).setSize(40)
        layout.itemAt(2).setAlignment(Qt.AlignHCenter)

        button_box = QDialogButtonBox(QDialogButtonBox.Ok |
QDialogButtonBox.Cancel)
        button_box.setStyleSheet("background-color: #FFD798;")
```

```
        button_box.accepted.connect(self.accept)
        button_box.rejected.connect(self.reject)
        layout.addWidget(button_box)
        layout.itemAt(3).setAlignment(Qt.AlignHCenter)


class MainWindow(QMainWindow):
    start = None
    goal = None
    solution_path = None

    def __init__(self):
        super().__init__()

        self.setWindowTitle("8 puzzle solver")
        self.setStyleSheet("background-color: #FFD798;")
        self.setFixedSize(QSize(500, 600))
        font = QFont()
        font.setPointSize(8)

        mainWidget = QWidget()
        mainLayout = QVBoxLayout(mainWidget)

        enterButton = QPushButton("Ввести")
        randomButton = QPushButton("Рандом")
        startButton = QPushButton("Вирішити задачу")
        enterButton.setStyleSheet("background-color: #BFDB7F;")
        randomButton.setStyleSheet("background-color: #BFDB7F;")
        startButton.setStyleSheet("background-color: #BFDB7F;")
```

```
enterButton.setFixedSize(QSize(90, 30))
randomButton.setFixedSize(QSize(90, 30))
startButton.setFixedSize(QSize(150, 30))
enterButton.clicked.connect(self.enter_button_clicked)
randomButton.clicked.connect(self.random_button_clicked)
startButton.clicked.connect(self.start_button_clicked)

buttonsLayout = QHBoxLayout()
buttonsLayout.setSpacing(20)
buttonsLayout.addWidget(enterButton)
buttonsLayout.addWidget(randomButton)
buttonsLayout.setAlignment(Qt.AlignHCenter)

radio_astar = QRadioButton("A*")
radio_astar.setFont(font)
radio_rbfs = QRadioButton("RBFS")
radio_rbfs.setFont(font)
self.radio_astar = radio_astar
self.radio_rbfs = radio_rbfs

radioLayout = QHBoxLayout()
radioLayout.addWidget(radio_astar)
radioLayout.addWidget(radio_rbfs)

self.puzzleLayout = PuzzleLayout([" ", " ", " ", " ", " ", " ", " ", " ", " "])

costLayout = QHBoxLayout()
costLayout.setSpacing(10)
costMessage = QLabel("Кількість відкритих вузлів:")
```

```
costMessage.setFont(font)
self.costLabel = QLabel()
self.costLabel.setFont(font)
self.costLabel.setFixedSize(100, 20)
costLayout.addWidget(costMessage)
costLayout.addWidget(self.costLabel)

fileButton = QPushButton("Записати в файл")
fileButton.setStyleSheet("background-color: #BFDB7F;")
fileButton.setFixedSize(QSize(150, 30))
fileButton.clicked.connect(self.file_button_clicked)

mainLayout.setAlignment(Qt.AlignHCenter)
mainLayout.setSpacing(20)
mainLayout.addLayout(buttonsLayout)
mainLayout.addWidget(radio_astar)
mainLayout.addWidget(radio_rbfs)
mainLayout.addWidget(startButton)
mainLayout.itemAt(3).setAlignment(Qt.AlignHCenter)
mainLayout.addLayout(self.puzzleLayout)
mainLayout.itemAt(4).setAlignment(Qt.AlignHCenter)
mainLayout.addLayout(costLayout)
mainLayout.addWidget(fileButton)
mainLayout.itemAt(6).setAlignment(Qt.AlignHCenter)
self.setCentralWidget(mainWidget)

self.error_box = QMessageBox()
self.error_box.setWindowTitle("Помилка")
```

```
        self.error_box.setStyleSheet("QMessageBox { background-color: #FE896E;}"
                                     "QPushButton { background-color: #FFD798; }")
        self.error_box.addButton(QMessageBox.Ok)

        self.message_box = QMessageBox()
        self.message_box.setWindowTitle("Повідомлення")
        self.message_box.setStyleSheet("QMessageBox { background-color: #BFDB7F;}"
                                       "QPushButton { background-color: #FFD798; }")
        self.message_box.addButton(QMessageBox.Ok)

    def start_button_clicked(self):
        if self.start is None or self.goal is None:
            self.error_box.setText("Помилка! Оберіть початковий і кінцевий стани")
            self.error_box.exec()
            return
        if self.radio_astar.isChecked():
            self.solution_path, cost = Astar(self.start, self.goal)
        elif self.radio_rbfs.isChecked():
            self.solution_path, cost = RBFS(self.start, self.goal)
        else:
            self.error_box.setText("Помилка! Оберіть один з алгоритмів")
            self.error_box.exec()
            return
        if self.solution_path is None:
            self.error_box.setText("Задача не має розв'язку")
            self.error_box.exec()
            return
```

```
    path = self.solution_path[:]
    self.costLabel.setText(str(cost))
    self.display_solution(path)

def display_solution(self, path):
    if not path:
        self.message_box.setText("Готово!!!")
        QTimer.singleShot(1500, lambda: self.message_box.exec())
        return

    step = path.pop(0)
    QTimer.singleShot(1000, lambda: self.update_puzzle(step, path))
    QApplication.processEvents()

def update_puzzle(self, step, path):
    self.puzzleLayout.update_layout(step)
    QApplication.processEvents()
    self.display_solution(path)

def enter_button_clicked(self):
    puzzle_input_dialog = PuzzleInputDialog()

    if puzzle_input_dialog.exec_() == QDialog.Accepted:
        self.start, self.goal = puzzle_input_dialog.get_puzzle_input()
        self.puzzleLayout.update_layout(self.start)
        self.costLabel.setText("")

def random_button_clicked(self):
    start = sample(range(1, 9), 8)
```

```
        end = sample(range(1, 9), 8)
        while count_inversions(start) % 2 != count_inversions(end) % 2:
            end = sample(range(1, 9), 8)
        start.insert(randint(0, 8), " ")
        end.insert(randint(0, 8), " ")

        random_dialog = RandomPuzzleDialog(start, end)

        if random_dialog.exec_() == QDialog.Accepted:
            self.start = start[:]
            self.goal = end[:]
            self.puzzleLayout.update_layout(self.start)
            self.costLabel.setText("")

    def str_puzzle(self, i):
        state = self.solution_path[i][:]
        state[state.index(" ")] = "_"
        return str(' '.join(map(str, state[0:3])) + '\n' + ' '.join(map(str, state[3:6])) +
'\n' + ' '.join(map(str, state[6:9])))

    def file_button_clicked(self):
        if self.solution_path is None:
            self.error_box.setText("Помилка! Жодного розв'язку знайдено не
було")
            self.error_box.exec()
            return
        with open("solution.txt", 'w') as file:
            file.write(self.str_puzzle(0))
            for i in range(1, len(self.solution_path)):
```

```
                file.write("\n\n  | \n \\|/ \n\n")
                file.write(self.str_puzzle(i))
        file.close()

        self.message_box.setText("Розв'язок записано у файл solution.txt")
        self.message_box.exec()
```

astar.py

```
from queue import PriorityQueue
from puzzle import Puzzle


def Astar(initial_state, goal_state):
    count = 1
    explored = []
    start_node = Puzzle(initial_state, goal_state, None, None, 0)
    if not start_node.is_solvable():
        return None
    successors = PriorityQueue()
    successors.put((start_node.evaluation_function, -count, start_node))

    while not successors.empty():
        node = successors.get()
        node = node[2]
        explored.append(node.state)
        if node.goal_test():
            return node.find_solution(), count
```

```
        children = node.generate_children()
        for child in children:
            is_explored = any(child.state == explored_state for explored_state in explored)
            if not is_explored:
                count += 1
                successors.put((child.evaluation_function, -count, child))

    return None
```

rbfs.py

```
from puzzle import Puzzle
from sys import maxsize

open_nodes = 0

def RBFS(initial_state, goal_state):
    global open_nodes
    open_nodes = 1
    start_puzzle = Puzzle(initial_state, goal_state, None, None, 0)
    if not start_puzzle.is_solvable():
        return None
    node = RBFS_search(start_puzzle, maxsize)
    node = node[0]
    return node.find_solution(), open_nodes


def RBFS_search(node, f_limit):
    global open_nodes
```

```
    successors = []

    if node.goal_test():
        return node, None
    children = node.generate_children()
    if not len(children):
        return None, maxsize
    count = -1
    for child in children:
        open_nodes += 1
        count += 1
        successors.append((child.evaluation_function, count, child))
    while len(successors):
        successors.sort()
        best_node = successors[0][2]
        if best_node.evaluation_function > f_limit:
            open_nodes -= len(children)
            return None, best_node.evaluation_function
        if len(successors) > 1:
            alternative = successors[1][0]
            result, best_node.evaluation_function = RBFS_search(best_node,
min(f_limit, alternative))
        else:
            result, best_node.evaluation_function = RBFS_search(best_node,
f_limit)
        successors[0] = (best_node.evaluation_function, successors[0][1],
best_node)
        if result is not None:
            break
```

```
    return result, None
```

puzzle.py

```
def count_inversions(state):
    inversions = 0
    for i in range(len(state)):
        for j in range(i + 1, len(state)):
            if state[i] > state[j]:
                inversions += 1
    return inversions

class Puzzle:
    __heuristic = None
    evaluation_function = None

    def __init__(self, state, goal, parent, action, path_cost):
        self.parent = parent
        self.__goal_state = goal
        self.state = state
        self.__action = action
        if parent:
            self.path_cost = parent.path_cost + path_cost
        else:
            self.path_cost = path_cost

        self.__generate_heuristic()
        self.evaluation_function = self.__heuristic + self.path_cost
```

```
def is_solvable(self):
    start_state = [num for num in self.state if num != " "]
    end_state = [num for num in self.__goal_state if num != " "]

    start_inversions = count_inversions(start_state)
    goal_inversions = count_inversions(end_state)

    if start_inversions % 2 == goal_inversions % 2:
        return True
    else:
        return False

def __generate_heuristic(self):
    self.__heuristic = 0
    for num in range(1, 9):
        distance = abs(self.state.index(num) - self.__goal_state.index(num))
        i = int(distance / 3)
        j = int(distance % 3)
        self.__heuristic = self.__heuristic + i + j

def goal_test(self):
    if self.state == self.__goal_state:
        return True
    return False

def __find_legal_actions(self, i, j):
    legal_action = ['U', 'D', 'L', 'R']
    if i == 0:
```

```
            legal_action.remove('U')
        elif i == 2:
            legal_action.remove('D')
        if j == 0:
            legal_action.remove('L')
        elif j == 2:
            legal_action.remove('R')

        legal_action = self.__critical_optimization(legal_action)
        return legal_action

    def __critical_optimization(self, legal_actions):
        if self.__action is not None:
            if self.__action == 'U':
                legal_actions.remove('D')
            elif self.__action == 'D':
                legal_actions.remove('U')
            elif self.__action == 'L':
                legal_actions.remove('R')
            elif self.__action == 'R':
                legal_actions.remove('L')
        return legal_actions

    def generate_children(self):
        children = []
        x = self.state.index(" ")
        i = int(x / 3)
        j = int(x % 3)
        legal_actions = self.__find_legal_actions(i, j)
```

```
    for action in legal_actions:
        new_state = self.state.copy()
        if action == 'U':
            new_state[x], new_state[x - 3] = new_state[x - 3], new_state[x]
        elif action == 'D':
            new_state[x], new_state[x + 3] = new_state[x + 3], new_state[x]
        elif action == 'L':
            new_state[x], new_state[x - 1] = new_state[x - 1], new_state[x]
        elif action == 'R':
            new_state[x], new_state[x + 1] = new_state[x + 1], new_state[x]
        children.append(Puzzle(new_state, self.__goal_state, self, action, 1))
    return children

def find_solution(self):
    solution = [self.state]
    path = self
    while path.parent is not None:
        path = path.parent
        solution.append(path.state)
    solution.reverse()
    return solution
```