

МИНОБРНАУКИ РОССИИ  
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ  
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
"ЛЭТИ" ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)  
КАФЕДРА ВТ

ОТЧЁТ  
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1  
НА ТЕМУ:  
"МЕТОДЫ НЕИНФОРМИРОВАННОГО (СЛЕПОГО) ПОИСКА"  
"1 ВАРИАНТ."

В ДИСЦИПЛИНЕ "ВВЕДЕНИЕ В ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ"

Выполнил: \_\_\_\_\_ Радабольский Владислав Сергеевич (группа 0308)

\_\_\_\_\_ Лесниченко Александр Олегович (группа 0308)

\_\_\_\_\_ Косневич Давид Андреевич (группа 0308)

Преподаватель: \_\_\_\_\_ Ельчанинов М.Н.

Санкт-Петербург  
2023

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
ПОСТАНОВКА ЗАДАЧИ.....	4
ОПИСАНИЕ СТРУКТУР ДАННЫХ.....	5
ОПИСАНИЕ АЛГОРИТМОВ .....	6
РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ .....	7
СРАВНИТЕЛЬНЫЕ ОЦЕНКИ СЛОЖНОСТЕЙ .....	9
ВЫВОД.....	10

## ВВЕДЕНИЕ

**Цель работы:** практическое закрепление понимания общих идей поиска в пространстве состояний и стратегий слепого поиска.

### Содержание:

1. Освоение теоретических основ слепого поиска (лекции, книга Рассел С, Норвиг П. Искусственный интеллект: современный подход, 2-е изд., М. «Вильямс», 2006, стр. 110–152). Стратегии, критерии, алгоритмы. Разбор алгоритмов различных стратегий поиска на конкретном примере в пошаговом режиме.

2. Рассмотрение принципов формализации задачи поиска и алгоритмов поиска для различных стратегий слепого поиска на конкретном примере. Программная реализация стратегий слепого поиска для конкретной задачи.

3. Теоретическая и экспериментальная оценка временной и емкостной сложности.

### Вариант 1:

Начальное состояние:

[5, 8, 3,  
4, 0, 2,  
7, 6, 1]

Конечное состояние:

[1, 2, 3,  
4, 5, 6,  
7, 8, 0]

Стратегии поиска: В глубину (dfs), В ширину (bfs)

## ПОСТАНОВКА ЗАДАЧИ

Состояние представляется массивом из 9 элементов, в котором пустое место обозначается нулем.

Для реализации алгоритмов был выбран язык программирования Python, так как он гораздо проще остальных ООП-языков синтаксически, а также в него встроен сборщик мусора, так что при написании алгоритмов не придётся отвлекаться на ручное управление памятью. Хочется отметить, что язык использует свой менеджмент памяти, поэтому показанные результаты можно интерпретировать с большой погрешностью, но все же они подходят для оценки временной и емкостной сложностей.

## ОПИСАНИЕ СТРУКТУР ДАННЫХ


Название структуры	Описание
Хэшированное множество	Используется для хранения хэшей состояний. Таким образом, за $O(1)$ можно проверить, что такое состояние уже существует.
Node	Пользовательская структура, описывающая узел. Опишем каждое из полей: – <code>current_state</code> - начальное состояние; <code>parent_node</code> - указатель на родительский узел; <code>previous_action</code> - действие, которое было применено к родительскому узлу для формирования данного узла; <code>path_cost</code> – стоимость пути; <code>depth</code> – глубина узла; <code>node_id</code> – уникальный идентификатор узла; <code>nodes_count</code> – статический счетчик вершин;
Очередь (Tree)	Структура, реализованная для хранения вершин дерева (как и рекомендовалось в мет. книге)
Словарь	Словарь используется для хранения узлов. В качестве ключа в словаре используется глубина, по ключу можно получить список (на основе массива), который будет содержать узлы на этом уровне.
Action	Структура для управления передвижением пустого элемента.

## ОПИСАНИЕ АЛГОРИТМОВ

Название алгоритма	Описание
<b>DFS (В глубину)</b>	<p>Алгоритм идет "внутрь" графа, до того момента как ему становится некуда идти, затем возвращается в предыдущую вершину и снова идет от нее до тех пор, пока есть куда идти. И так далее.</p> <p>Поскольку мы обходим каждого «соседа» каждого узла, игнорируя тех, которых посещали ранее, мы имеем время выполнения, равное <math>O(V + E)</math>.</p>
<b>BFS (В ширину)</b>	<p>Сначала мы проходимся по всем вершинам смежным со стартовой, потом по всем, смежным со смежными стартовой и так далее.</p> <p>Время выполнения BFS также составляет <math>O(V + E)</math>, а поскольку мы используем очередь, вмещающую все вершины</p>

## РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

Алгоритм обхода в ширину:



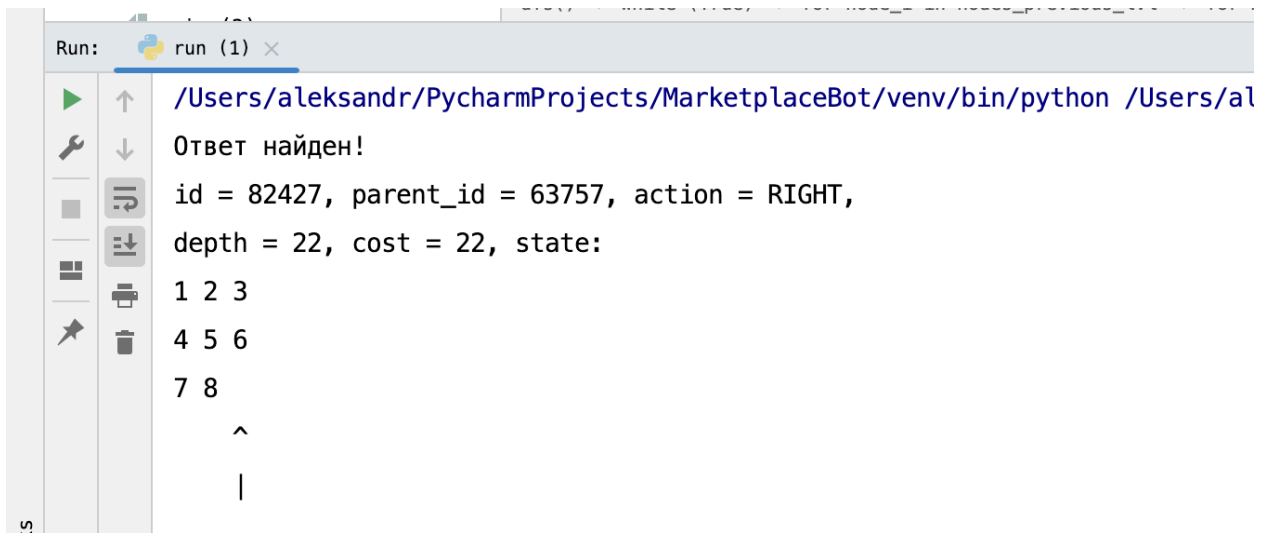
```
Run: run (1) x
id = 0, parent_id = 0, action = None,
depth = 0, cost = 0, state:
5 8 3
4 2
7 6 1
^
|

Итого узлов: 82429
Итого итераций: 82428
Потраченное время процессора: 867.048 миллисекунд
Памяти использовано: 51654656 байтов
← ОБХОД В ШИРИНУ (BFS) →

Process finished with exit code 0
```

Рис. 1

Конечное состояние:



```
Run: run (1) x
/Users/aleksandr/PycharmProjects/MarketplaceBot/venv/bin/python /Users/al
Ответ найден!
id = 82427, parent_id = 63757, action = RIGHT,
depth = 22, cost = 22, state:
1 2 3
4 5 6
7 8
^
|
```

Рис. 2

## Алгоритм обхода в глубину:



Рис. 3

## Конечное состояние:

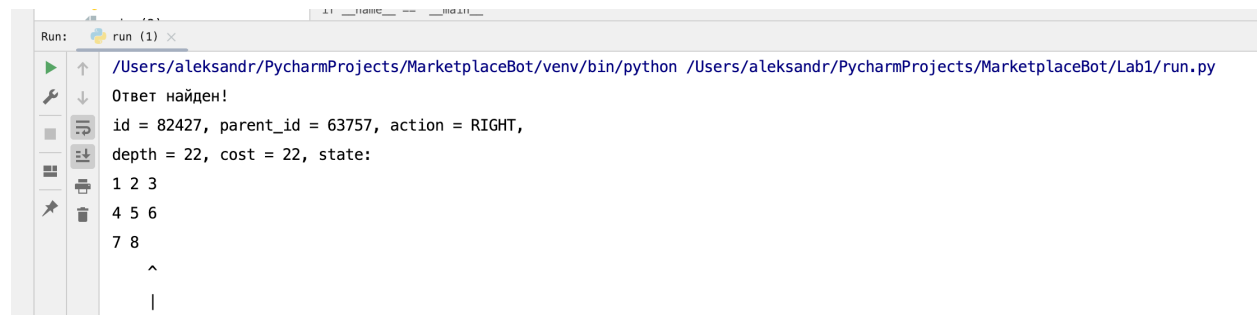


Рис. 4



## СРАВНИТЕЛЬНЫЕ ОЦЕНКИ СЛОЖНОСТЕЙ

Алгоритм	Время, [м/с]
DFS	886.303, 82428 итераций, 48.9 Мб
BFS	867.048, 82429 итераций, 49.3 Мб

## ВЫВОД

В ходе лабораторной работы №1 были разработаны запланированные структуры данных, реализованы алгоритмы поиска в ширину и глубину. Рассчитаны временные сложности алгоритмов и записана сравнительная оценка сложности.

Листинг представлен в репозитории GitHub:

[https://github.com/alexandrLes/eltech\\_labs/tree/IntroductionToAI](https://github.com/alexandrLes/eltech_labs/tree/IntroductionToAI)