Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 6
по дисциплине «Анализ алгоритмов»

Тема Поиск в словаре

Студент Калашников С.Д.

Группа ИУ7-53Б

Преподаватель Волкова Л.Л., Строганов Ю.В.

Москва, 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является получение навыка поиска по словарю при ограничении на значение признака, заданном при помощи лингвистической переменной. Для достижения поставленной цели необходимо выполнить следующие задачи:

1) формализовать объект по варианту и его признак;

2) составить анкета для заполнения респондентом;

3) провести анкетирование респондентов;

4) построить функцию принадлежности термам числовых значений признака, описываемого лингвистической переменной, на основе статистической обработки мнений респондентов, выступающих в роли экспертов;

5) описать алгоритм поиска в словаре объектов;

6) описать структуру данных словаря;

7) реализовать описанный алгоритм поиска в словаре;

8) описать и обосновать результаты в виде отчёта о выполненной лабораторной работе, выполненном как расчётно-пояснительная записка к работе.

1 Аналитическая часть

В данном разделе будут рассмотрены словарь как структура данных и алгоритм полного перебора, а также представлены требования к разрабатываемой программе.

1.1 Словарь как структура данных

Словарь — абстрактный тип данных (интерфейс к хранилищу данных), позволяющий хранить пары вида «(ключ, значение)» и поддерживающий операции добавления пары, а также поиска и удаления пары по ключу:

1) *insert(k, v)*;

2) *find(k)*;

3) *remove(k)*.

В паре *(k, v)*: *v* называется значением, ассоциированным с ключом *k*. Где *k* — это ключ, а *v* — значение. Семантика и названия вышеупомянутых операций в разных реализациях ассоциативного массива могут отличаться.

Операция поиска *find(k)* возвращает значение, ассоциированное с заданным ключом, или некоторый специальный объект, означающий, что значения, ассоциированного с заданным ключом, нет. Две другие операции ничего не возвращают (за исключением, возможно, информации о том, успешно ли была выполнена данная операция).

Словарь с точки зрения интерфейса удобно рассматривать как обычный массив, в котором в качестве индексов можно использовать не только целые числа, но и значения других типов — например, строки (именно по этой причине словарь также иногда называют «ассоциативным массивом»).

1.2 Алгоритм полного перебора

Алгоритмом полного перебора называют метод решения задачи, при котором по очереди рассматриваются все возможные варианты. В случае реализации алгоритма в рамках данной работы будут последовательно перебираться ключи словаря до тех пор, пока не будет найден нужный.

Трудоёмкость алгоритма зависит от того, присутствует ли искомый ключ в словаре, и, если присутствует – насколько он далеко от начала массива ключей. Пусть на старте алгоритм затрагивает k_0 операций, а при сравнении k_1 операций.

Пусть алгоритм нашёл элемент на первом сравнении (лучший случай), тогда будет затрачено $k_0 + k_1$ операций, на втором — $k_0 + 2 \cdot k_1$, на последнем (худший случай) — $k_0 + N \cdot k_1$. Если ключа нет в массиве ключей, то мы сможем понять это, только перебрав все ключи, таким образом трудоёмкость такого случая равно трудоёмкости случая с ключом на последней позиции. Трудоёмкость в среднем может быть рассчитана как математическое ожидание по формуле (1.1), где Ω — множество всех возможных случаев.

$$\sum_{i \in \Omega} p_i \cdot f_i = k_0 + k_1 \cdot \left(1 + \frac{N}{2} - \frac{1}{N+1}\right) \tag{1.1}$$

1.3 Требования к программе

К разрабатываемой в данной работе программе предъявляется ряд требований:

1) на вход будет подаваться строка, на основании которой производится поиск;

2) на выходе — результат поиска в словаре;

3) программа не должна аварийно завершаться при отсутствии ключа в словаре.

1.4 Вывод

В данном разделе были рассмотрены словарь как структура данных и алгоритм полного перебора, а также приведены требования к разрабатываемой программе.

2 Конструкторская часть

Вывод

В данном разделе были описаны используемые структуры и приведены схемы алгоритмов.

3 Технологическая часть

В данном разделе будут рассмотрены средства реализации, а также представлены листинги алгоритмов.

3.1 Средства реализации

В данной работе для реализации был выбран язык программирования $c\#$. В текущей лабораторной работе требуется замерить время работы реализаций алгоритмов. Для этого используется класс $Stopwatch$ из библиотеки $System.Diagnostics$.

3.2 Сведения о модулях программы

Программа состоит из следующих классов:

— $AntAlgorithm$ — класс, реализующий муравьиный алгоритм;

— $Program$ — класс, реализующий основную программу;

— $BruteForce$ — класс, реализующий алгоритм полного перебора;

3.3 Реализация алгоритмов

В листингах 3.1–3.2 представлены реализации алгоритмов.

Листинг 3.1 — Реалитзация алгоритма полного перебора

Листинг 3.2 — Реализация муравьиного алгоритма

3.4 Функциональное тестирование

Вывод

В данном разделе были рассмотрены листинги используемых алгоритмов, проведено функциональное тестирование.

4 Исследовательская часть

В данном разделе будет проведен сравнительный анализ времени работы реализаций алгоритмов при различных ситуациях на основе полученных данных.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялись замеры времени, представлены далее:

— операционная система Windows 11 Pro Версия 22H2 (22621.674) [1];

— память 16 ГБ;

— процессор 11th Gen Intel(R) Core(TM) i5-11400 2.59 ГГц [2], 6 физических и 12 логических ядер.

При тестировании компьютер был включен в сеть электропитания. Во время замеров процессорного времени устройство было нагружено только встроенными приложениями окружения, а также системой тестирования.

4.2 Время выполнения реализаций алгоритмов

На рис. 4.1 и рис. 4.2 показаны результаты замеров времени.

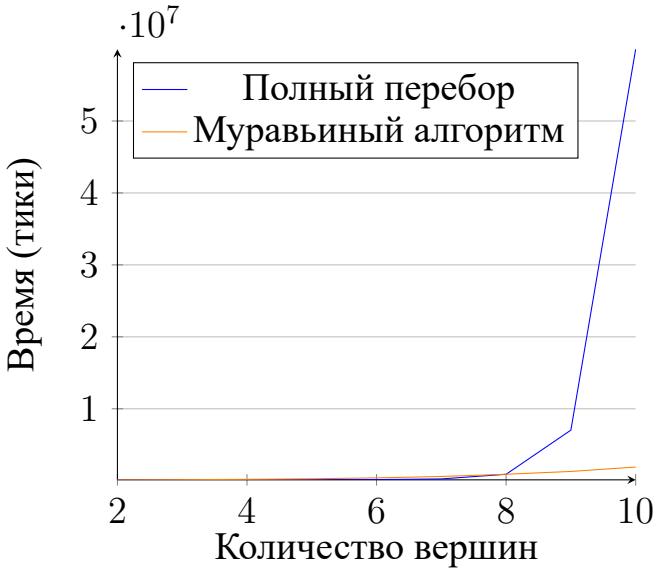

Рис. 4.1 – Сравнение времени работы алгоритмов при увеличении размера графа

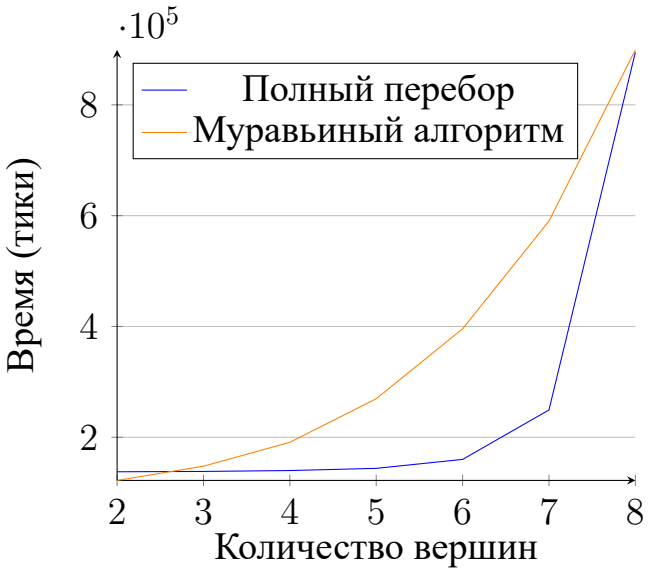

Рис. 4.2 – Сравнение времени работы алгоритмов на малых размерах графа

4.3 Вывод

По полученным результатам иследования можно сделать вывод, что

ЗАКЛЮЧЕНИЕ

Цель, которая была поставлена в начале лабораторной работы, была достигнута: изучена задача коммивояжера и реализован алгоритм полного перебора и муравьиный алгоритм для ее решения.

В ходе выполнения лабораторной работы были решены все задачи:

1) исследована задача коммивояжера;

2) описаны алгоритм полного перебора и муравьиный алгоритм для решения задачи коммивояжера;

3) реализованы данные алгоритмы;

4) проведена параметризация муравьиного алгоритма на нескольких классах данных;

5) проведены замеры времени для реализованных алгоритмов;

6) выполнен анализ полученных результатов;

7) по итогам работы составлен отчет.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Windows 11, version 22H2 [Эл. ресурс]*. Режим доступа: https://clck.ru/32NCXx (дата обращения: 14.10.2022).
2. *Процессор Intel® Core™ i7 [Эл. ресурс]*. Режим доступа: https://clck.ru/yeQa8 (дата обращения: 14.10.2022).