Министерство образования Республики Беларусь

Учреждение образования

«Гомельский государственный университет им. Франциска Скорины»

**Отчет по лабораторной работе №6 на тему:**

**«Виды сортировки»**

Выполнил: Никитин П. Д.

Проверил: Соколов С. И.

2020

**Введение**

Иногда данные, которые мы храним или извлекаем в приложении, могут находится в беспорядочном состояние. И иногда возникает необходимость упорядочивания данные прежде чем их можно будет эффективно использовать. За все эти годы учеными было создано множество алгоритмов сортировки для организации данных. В этой статье мы рассмотрим наиболее популярные алгоритмы сортировки, разберем, как они работают, и напишем их на Python. Мы также сравним, как быстро они сортируют элементы в списке. Для простоты реализации алгоритмов сортировать числа будем в порядке их возрастания.

**Пузырьковая сортировка.** Этот самый простой алгоритм сортировки который выполняет итерации по списку, сравнивая элементы попарно и меняя их местами, пока более крупные элементы не перестанут «всплывать» до конца списка, а более мелкие элементы не будут оставаться «снизу».

**Сортировка выбором**. Этот алгоритм сегментирует список на две части: отсортированные и несортированные. Он постоянно удаляет наименьший элемент из несортированного сегмента списка и добавляет его в отсортированный сегмент.

**Сортировка вставками**. Как и Сортировка выбором, этот алгоритм сегментирует список на отсортированные и несортированные части. Он перебирает несортированный сегмент и вставляет просматриваемый элемент в правильную позицию отсортированного списка.

**Пирамидальная сортировка**. Этот популярный алгоритм сортировки, как сортировки вставками и выбором, сегментирует список на отсортированные и несортированные части. Он преобразует несортированный сегмент списка в структуру данных типа куча (heap), чтобы мы могли эффективно определить самый большой элемент.

**Сортировка слиянием.** Этот алгоритм «разделяй и властвуй» разбивает список пополам и продолжает разбивать список на пары, пока в нем не будут только одиночные элементы. Соседние элементы становятся отсортированными парами, затем отсортированные пары объединяются и сортируются с другими парами. Этот процесс продолжается до тех пор, пока мы не получим отсортированный список со всеми элементами несортированного списка.

**Быстрая сортировка**. Это то же алгоритм «разделяй и властвуй» и его наиболее часто используют их описанных в этой статье. При правильной настройке он чрезвычайно эффективен и не требует дополнительного пространства памяти как сортировка слиянием. Мы разделяем список вокруг элемента точка опоры, сортируя значения вокруг этой точки.

**Задание:**

Чтобы понять, как быстро работают рассмотренные алгоритмы, необходимо сгенерировать список из N чисел со значениями от 0 до 1000. Затем необходимо определить время, необходимое для завершения каждого алгоритма. Повторить это 10 раз, чтобы можно было более надежно установить производительность сортировки. Таким образом, для каждого вида сортировки (в том числе и для встроенной сортировки) провести сортировку массивов при N = 100, 1000, 3000, 5000, 7000, 10000, 20000, 50000 элементов. Для каждого вида сортировки провести не менее 10 измерений времени сортировки. Для каждого вида сортировки построить график зависимости времени сортировки от размера массива (оценить зависимость нотации большого О от N). Сравнить полученные графики с теоретическими и сделать выводы.

**Решение:**

Напишем программу на Python гарнирующую список из N чисел со значениями от 0 до 1000. Затем программа определяет время, необходимое для завершения каждого алгоритма. Повторим 10 раз, чтобы можно было более надежно установить производительность сортировки.

Таким образом, для каждого вида сортировки (в том числе и для встроенной сортировки) проведены сортировки массивов при N = 100, 1000, 3000, 5000, 7000, 10000, 20000, 50000 элементов. Также для каждого вида сортировки проведены 10 измерений времени сортировки.

**Таблица 1 ­ - Сложность алгоритмов сортировки**

|  |  |
| --- | --- |
| Алгоритм сортировки | Сложность |
| Пузырьком | n2 |
| Выбором | n2 |
| Вставками | n2 |
| Кучей | n log n |
| Слиянием | n log n |
| Быстрая | n log n |
| Встроенная | неизвестно |

Для каждого вида сортировки построим график зависимости времени сортировки от размера массива.

Графики разделены на три группы ввиду, большой разности в масштабах.

Рисунок 1 – Сортировка пузырьком, выбором, вставками

Рисунок 2 – Сортировка быстрая, встроенная

Рисунок 3 – Сортировка кучей и слиянием

Как видно, самой быстрой оказалась сортировка встроенным методом, что происходит по тому, что встроенные библиотеки Python написаны на языке программирования С, что делает их очень быстрыми. А самой медленной оказалась сортировка пузырьком.

Сравним полученные графики с теоретическими.

Рисунок 4 – Теоритический вид графика n2

Рисунок 5 – Теоритический вид графика n log n

Графики имеют форму соответствующую, теоритическим.

## Заключение:

В процессе данной лабораторной работы, была написана программа реализующая тестирование времени выполнения разных методов сортировки. Результаты были сопоставлены с теоритической базой. Полученные графики абсолютно соответствуют теоритическим. Для каждого вида сортировки было проведено не менее 10 измерений времени сортировки. Также для каждого вида сортировки был построен график зависимости времени сортировки от размера массива.