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

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 «Записи с вариантами. Обработка таблиц»

Студент Городский Юрий Николаевич

Группа ИУ7 – 32Б

2023 г.

Оглавление

Условие задачи

Создать таблицу, содержащую не менее 40-ка записей (тип - запись с вариантами (объединениями)). Упорядочить данные в ней по возрастанию ключей, двумя алгоритмами сортировки, где ключ – любое невариантное поле (по выбору программиста), используя:

- саму таблицу
- массив ключей.

(Возможность добавления и удаления записей в ручном режиме обязательна).

Имеются описания: Тип жилье = (дом, общежитие, съемное);

- Данные :
  - Фамилия, имя, группа, пол (м, ж), возраст, средний балл за сессию, дата поступления
- адрес:
  - дом : (улица, №дома, №кв);
  - общежитие : (№общ., №комн.)
  - съемное : (улица, №дома, №кв, стоимость);

Ввести общий список студентов.

Вывести список студентов, указанного года поступления, живущих

в съемном жилье.

Техническое задание

Входные данные:

1. **Файл с данными:** текстовый файл, каждое поле новой записи находится на новой строке.
2. **Номер команды:** целое число в диапазоне от 1 до 15
3. **Новая запись таблицы:** значение каждого поля вводятся через enter
4. **Ограничения полей (обозначены в интерфейсе):**
   1. Фамилия — строка до 10 символов
   2. Имя — строка до 10 символов

3. Группа — строка до 10 символов
4. Пол — значение 0(ж) или 1(м)
5. Возраст — целое число >= 0
6. Средняя оценка за сессию — вещественное число от 1 до 5
7. Дата поступления — целое число >= 0
8. Улица — строка до 10 символов
9. № дома, № квартиры, № общежития, № комнаты — целое число >= 0
10. Стоимость — дробное число

Выходные данные:

1. Таблица с данными в текущем состоянии (в зависимости от команды)
2. Таблица вида ключ — значение, где ключ — номер записи в таблице с данными, а значение
3. Файл с сохраненной таблицей, каждое поле каждой записи находится на новой строке
4. Файл с результатами временных замеров каждая строка которых содержит количество записей в таблице, время сортировки, количество затраченной памяти в байтах.

Функции программы

1. Прочитать из файла
2. Вывести таблицу
3. Добавить запись
4. Удалить запись
5. Загрузить исходную таблицу в буфер
6. Отсортировать по оценкам таблицу с помощью quick sort
7. Отсортировать по оценкам таблицу с помощью bubble sort
8. Отсортировать по оценкам массив ключей с помощью quick sort
9. Отсортировать по оценкам массив ключей с помощью bubble sort
10. Вывести массив ключей
11. Вывести таблицу по массиву ключей

12. Поиск студентов, живущих в съемном жилье, с заданным годом поступления

13. Сохранить таблицу

14. Замерный эксперимент

15. Выйти

Аварийные ситуации:

1. Некорректный ввод команды (введено не число или число не находится в диапазоне от 1 до 15): сообщение «Неверная команда»
2. Неверно введено введено имя файла (длина названия больше 50 символов): сообщение «Ошибка: не удалось считать имя файла»
3. Не удалось открыть файл: сообщение «Не удалось открыть файл»
4. Неправильный формат файла: сообщение «Неправильный формат файла»
5. Неправильное диапазон значение поля: сообщение «Неверное значение [имя поля]»
6. Не удалось считать значение поля: «Не удалось считать [имя поля]»

Обращение к программе

Запуск через терминал (./app.exe)

Структура данных

```
// Типы жилья
enum
{
    HOUSE, // Дом
    DORMITORY, // Общежитие
    RENTAL // Съемное жилье
};

// Дом
typedef struct
{
    char sreet[STREET_LEN + 1]; // Улица
    unsigned int house_number;  // Номер дома
    unsigned int flat_number;   // Номер квартиры
} house_t;

// Общежитие
typedef struct
{
    unsigned int dormitory_number; // Номер общежития
    unsigned int room_number;      // Номер комнаты
} dormitory_t;

// Съемное жилье
typedef struct
{
    char sreet[STREET_LEN + 1]; // Название улицы
```

```
    unsigned int house_number;  // Номер дома
    unsigned int flat_number;   // Номер квартиры
    double cost;                // Стоимость
} rental_t;

// Студент
typedef struct
{
    size_t n;                     // Номер строки
    unsigned int housing_type;    // Тип жилья(Дом, общежитие, съемное жилье)
    char surname[SURNAME_LEN +1]; // Фамилия
    char name[NAME_LEN + 1];      // Имя
    char group[GROUP_LEN + 1];    // Группа
    unsigned int sex;             // Пол (м,ж - 1, 0)
    unsigned int age;             // Возраст
    double average_mark;          // Средний былл за сессию
    unsigned int entry_date;      // Год поступления
    union                         // Место проживания
    {
        house_t house;            // Дом
        dormitory_t dormitory;    // Общежитие
        rental_t rental;          // Съемное жилье
    } housing;
} student_t;
```

Для хранения ключей используются структуры

```
typedef struct
{
    size_t n;             // Индекс в таблице
    double average_mark;  // Значение средней оценки
} student_key_t;
```

Описание алгоритма

1. Выводится меню команд, вводится и выполняется команда
2. Если введено значение 15 программа завершает свою работу

Тесты

Таблица 1: Положительные тесты

№	Описание	Входные данные	Выходные данные
1	Прочитать из файла	Название файла с корректными входными	«Таблица загружена»

		данными	
2	Вывести таблицу	2	{Таблица}
3	Вывести пустую тулицу	2	«Пустая таблица»
4	Добавить запись	3 {корректные данные}	«Значение добавлено»
5	Удалить запись	4 {Корректный номер записи}	«Удаление совершено»
6	Загрузить исходную таблицу в буфер	5	«Таблица загружена»
7	Отсортиовать по оценкам таблицу с помощью quick sort	6	«Сортировка завершена»
8	Отсортиовать по оценкам таблицу с помощью bubble sort	7	«Сортировка завершена»
9	Отсортиовать по оценкам массив ключей с помощью quick sort	8	«Сортировка завершена»
10	Отсортиовать по оценкам массив ключей с помощью bubble sort	9	«Сортировка завершена»
11	Вывести массив ключей, массива не пуст	10	{Массив ключей}
12	Вывести массив ключей, массива пуст	11	«Пустой массив»
12	Вывести таблицу по массиву ключей	11	{Таблица в порядке указанном в массиве ключей}
13	Поиск студентов, живущих в съемном жилье, с заданным годом поступления, найденные значения есть	12	{Таблица с найденными значениями}
15	Сохранить таблицу	13 {Корректное название файла}	«Таблица сохранена»
16	Замерный эксперимент	14	«Измерение завершен»
17	Выйти	15	Программа завершает работу

Таблица 2: Негативные тесты

№	Описание	Целое число	Результат
	Неверный диапазон значения команды	0	
	Неверное значение команды (не число)	a	
	Неправильное имя файла (длина превышает 50)	A 51 раз	«Не удалось считать имя файла»
	Неправильный формат файла	{файл с неправильными данными}	«Неправильный формат файла»
	Неверный диапазон поля	Тип жилья: 15	«Неверное значение типа»
	Неверное значение поля	Тип жилья: ag	«Не удалось считать тип»

Замерный эксперимент

Сравнивались быстрая сортировка пузырьком с флагом и быстрая сортировка. Замеры проводились на 500 элементах (значения сортируемого поля сгенерированы случайно, но одинаковы для каждого случая), для каждого случая проводилось 10 измерений и бралось среднее значение.

Рассматриваемый случай	Количество элементов в таблице	Время сортировки (с)	Затраченная память
Qsort таблиц	500	0.000079	52000
Bubble sort таблица	500	0.024266	52000
Qsort массив ключей	500	0.000070	60000
Bubble sort массив ключей	500	0.004872	60000

На 500 элементах использование сортировки массива ключей ускоряет время сортировки на 8.86% в случае использования qsort и на 79.93% в случае использования bubble sort.

Для определения дальнейшей зависимости проведены аналогичные замеры от 1000 до 10000 с шагом 1000.

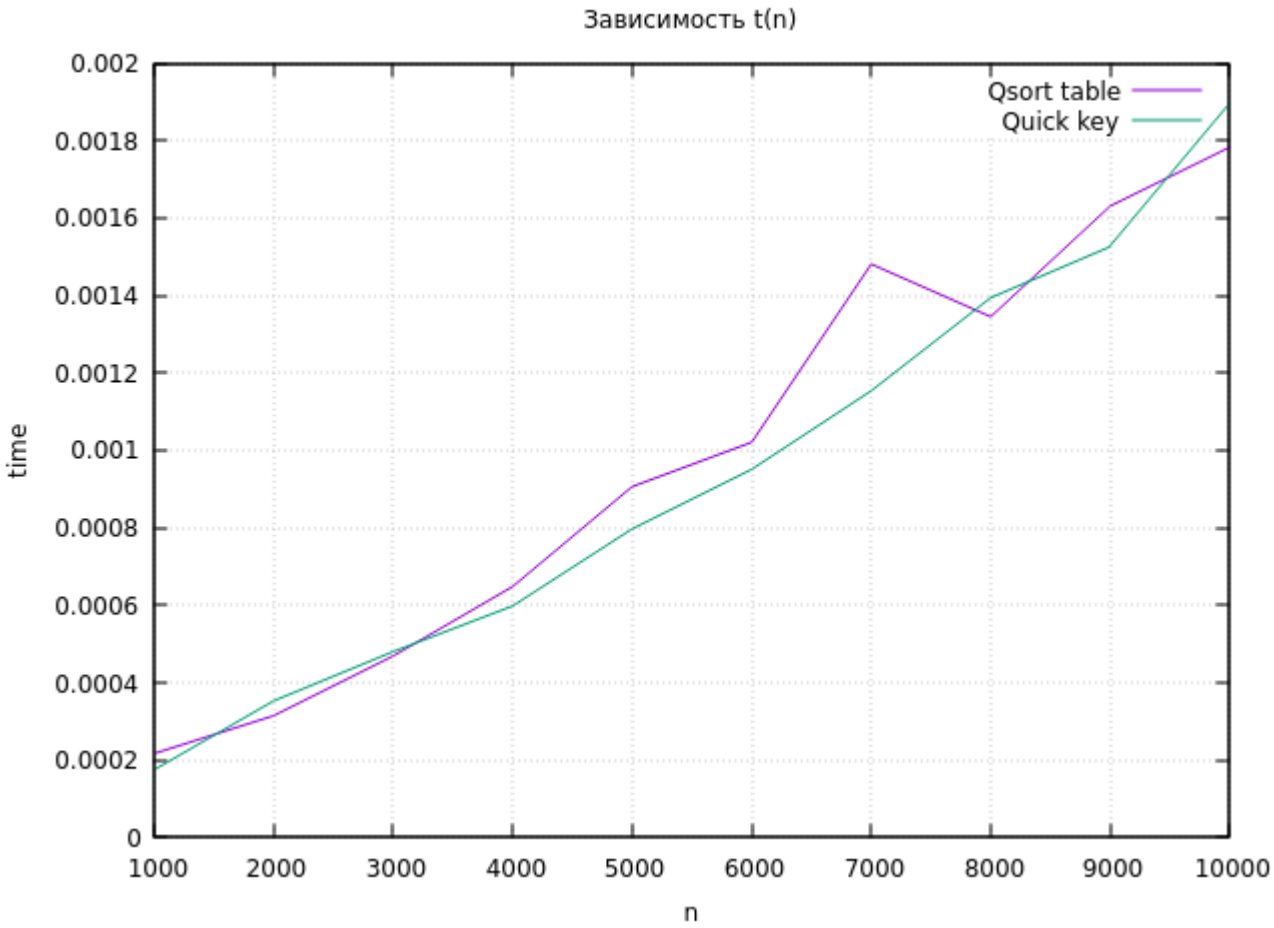

Рисунок 1: Зависимость времени сортировки с помощью qsort

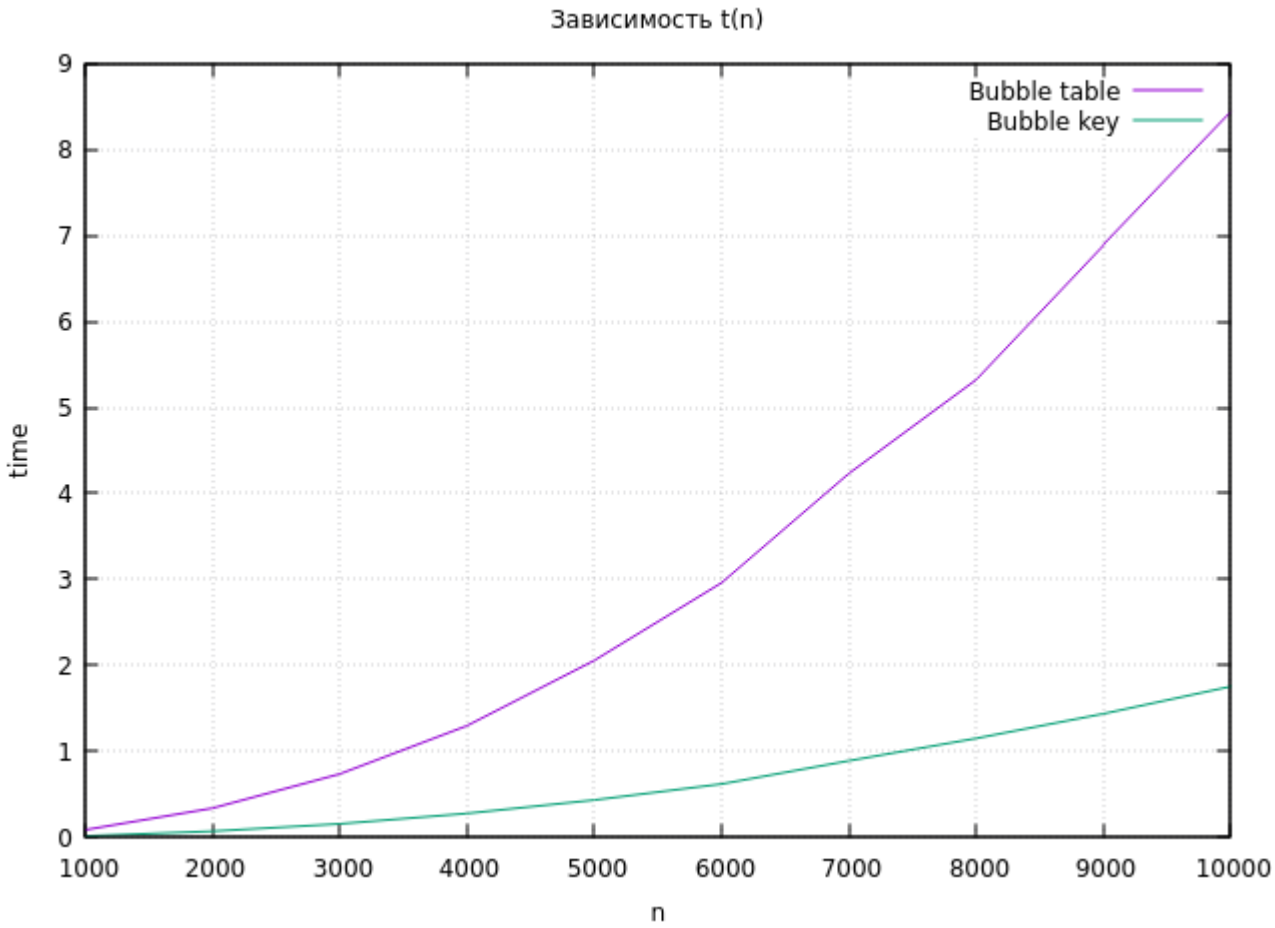

Рисунок 2: Зависимость времени сортировки с помощью bubble sort

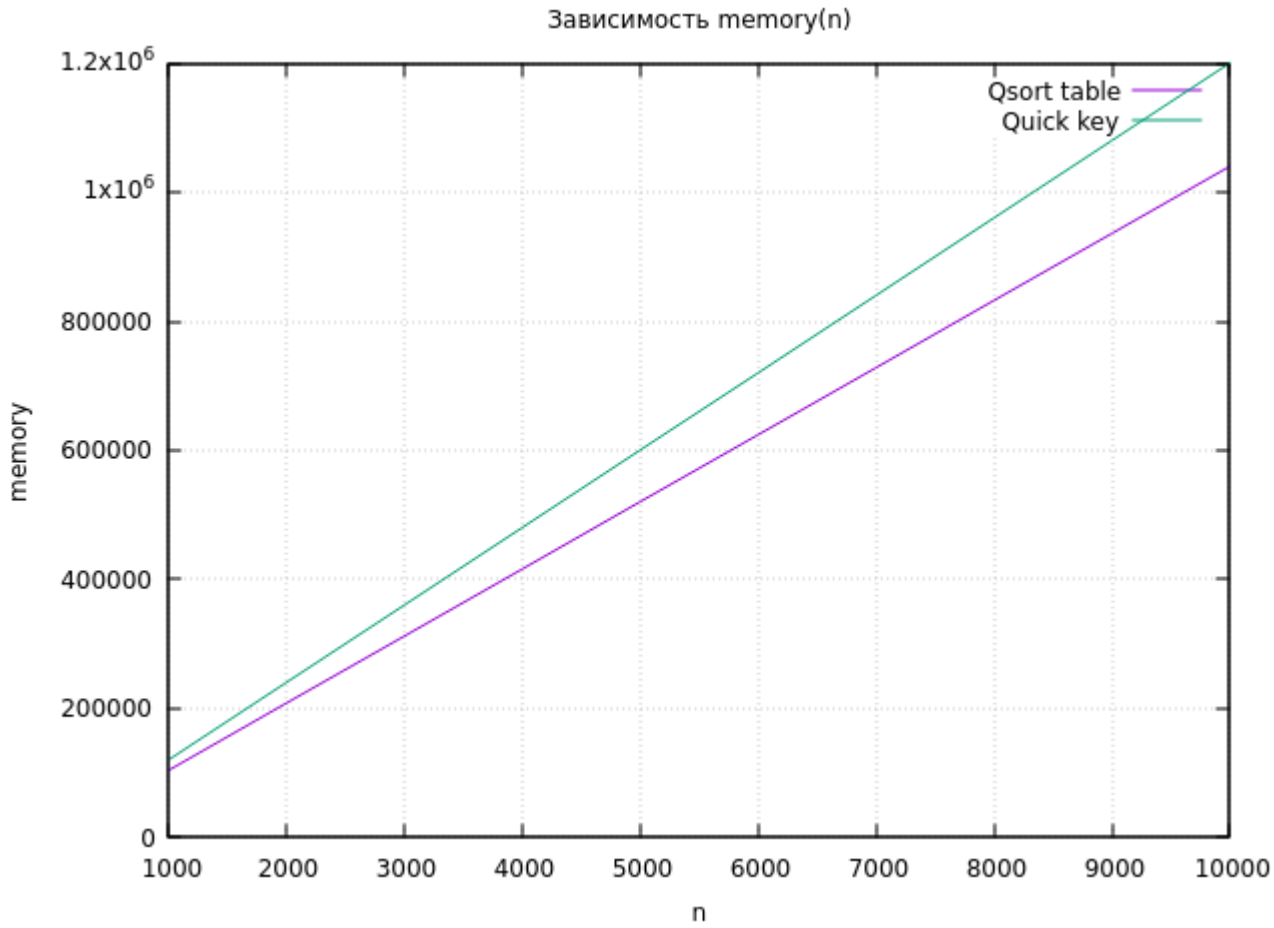

Рисунок 3: Зависимость памяти qsort

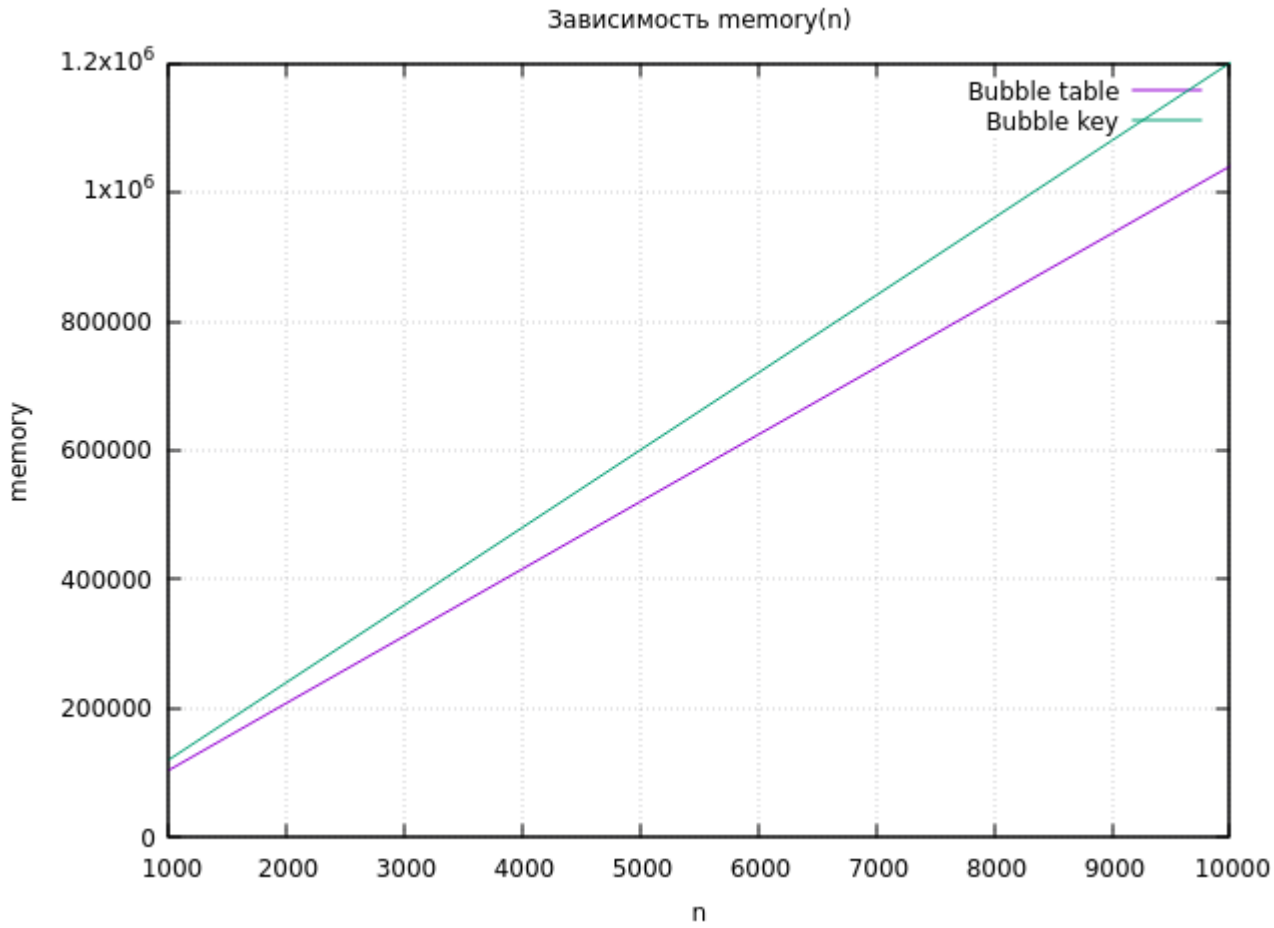

Рисунок 4: Зависимость памяти bubble sort

Контрольные вопросы

1. Выделяется общий блок памяти для всех полей вариантной части (берется максимально возможный размер вариантной части).
2. Поведение будет не определено, данные корректно считать не получится.
3. За правильностью выполнения операций следит программист.
4. Таблица ключей — массив (структура), содержащая индексы определенного элемента каждой записи исходной таблица. Таблица ключей позволяет сократить время при сортировке и поиске записей в исходной таблице.
5. Таблица ключей позволяет уменьшить время сортировки таблицы, т. к. перестановка записей занимает меньшее время. Минусы: требует дополнительной памяти, эффективность не сильно возрастет, если в таблице мало полей.

6. При сортировке самой таблице лучше использовать алгоритмы, требующие наименьшее количество перестановок. При сортировке массива ключей лучше использовать сортировки с наименьшей сложностью.

Вывод

Одним из способов уменьшения размера данных при обработке является использование объединений. При работе с большими данными использование таблицы ключей и выбор подходящего алгоритма сортировки существенно сокращают время обработки данных. Не смотря на увеличение затрачиваемой памяти эти методы сильно повышают скорость и эффективность обработки данных.