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 № 3

по курсу «Анализ Алгоритмов»

на тему: «Трудоемкость сортировок»

Студент ИУ7-53Б _____ Лысцев Н. Д.
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ Волкова Л. Л.
(Подпись, дата) (И. О. Фамилия)

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сортировка – процесс перегруппировки последовательности объектов в некотором порядке. Это одна из фундаментальных операций в алгоритмике и компьютерных науках, играющая ключевую роль в эффективной обработке данных.

Целью данной лабораторной работы является исследование трех алгоритмов сортировки: блочной сортировки, сортировки слиянием и поразрядной сортировки.

Для достижения поставленной цели необходимо решить следующие задачи:

1) Изучить и описать три алгоритма сортировки: блочной, слиянием и поразрядной.

2) Создать программное обеспечение, реализующее следующие алгоритмы:

   – алгоритм блочной сортировки;

   – алгоритм сортировки слиянием;

   – алгоритм поразрядной сортировки.

3) Провести анализ эффективности реализаций алгоритмов по памяти и по времени.

4) Провести оценку трудоемкости алгоритмов сортировки.

5) Обосновать полученные результаты в отчете к выполненной лабораторной работе.

1 Аналитический раздел

В данном разделе будут рассмотрены алгоритм блочной сортировки, сортировки слиянием и поразрядной сортировки.

1.1 Алгоритм блочной сортировки

Блочная сортировка – алгоритм сортировки, в котором сортируемые элементы распределяются между конечным числом отдельных блоков так, чтобы все элементы в каждом следующем по порядку блоке были всегда больше (или меньше), чем в предыдущем. Каждый блок затем сортируется отдельно, либо рекурсивно тем же методом, либо другим. Затем элементы помещаются обратно в массив [1].

1.2 Алгоритм сортировки слиянием

Сортировка слиянием – алгоритм сортировки, который упорядочивает списки (или другие структуры данных, доступ к элементам которых можно получать только последовательно, например — потоки) в определённом порядке. Эта сортировка — хороший пример использования принципа «разделяй и властвуй» [2].

Алгоритм действий в сортировке слиянием:

1) Сортируемый массив разбивается на две части примерно одинакового размера;

2) Каждая из получившихся частей сортируется отдельно, например — тем же самым алгоритмом;

3) Два упорядоченных массива половинного размера соединяются в один.

1.3 Алгоритм поразрядной сортировки

Поразрядная сортировка – алгоритм сортировки, который выполняется за линейное время. Сравнение производится поразрядно: сначала сравниваются значения одного крайнего разряда, и элементы группируются по результатам

этого сравнения, затем сравниваются значения следующего разряда, соседнего, и элементы либо упорядочиваются по результатам сравнения значений этого разряда внутри образованных на предыдущем проходе групп, либо переупорядочиваются в целом, но сохраняя относительный порядок, достигнутый при предыдущей сортировке. Затем аналогично делается для следующего разряда, и так до конца [3].

Вывод

В данном разделе были рассмотрены алгоритм блочной сортировки, сортировки слиянием и поразрядной сортировки.

2 Технологический раздел

В данном разделе будут перечислены средства реализации, листинги кода и функциональные тесты.

2.1 Требования к программному обеспечению

К программе предъявляется ряд требований:

- на вход подаётся вектор элементов;
- все элементы вектора - целые неотрицательные числа (это необходимо для возможности сравнения сортировок между собой);
- на выходе в том же векторе находятся отсортированные по возрастанию элементы исходного.

2.2 Средства реализации

В качестве языка программирования для этой лабораторной работы был выбран $C++$ [4] по следующим причинам:

- в $C++$ есть встроенный модуль $ctime$, предоставляющий необходимый функционал для замеров процессорного времени;
- в стандартной библиотеке $C++$ есть оператор $sizeof$, позволяющий получить размер переданного объекта в байтах. Следовательно, $C++$ предоставляет возможности для проведения точных оценок по используемой памяти.

В качестве функции, которая будет осуществлять замеры процессорного времени, будет использована функция $clock\_gettime$ из встроенного модуля $ctime$ [5].

2.3 Сведения о модулях программы

Программа состоит из шести модулей:

1) `algorithms.cpp` — модуль, хранящий реализации алгоритмов сортировки;

2) `processTime.cpp` — модуль, содержащий функцию для замера процессорного времени;

3) `memoryMeasurements.cpp` — модуль, содержащий функции, позволяющие провести сравнительный анализ использования памяти в реализациях алгоритмов сортировки;

4) `timeMeasurements.cpp` — модуль, содержащий функции, позволяющие провести сравнительный анализ использования времени в реализациях алгоритмов сортировки;

5) `main.cpp` — файл, содержащий точку входа в программу;

6) `task7` — модуль, содержащий набор скриптов для проведения замеров программы по времени и памяти и построения графиков по полученным данным.

2.4 Реализации алгоритмов

В листингах ...

Листинг 2.1 – Реализация алгоритма блочной сортировки

```
void bucketSort(vector<int> &arr)
{
    int n = arr.size();
    int minVal = *min_element(arr.begin(), arr.end());
    int maxVal = *max_element(arr.begin(), arr.end());
    int bucketRange = (maxVal - minVal) / n + 1;
    int bucketIndex, i, j, index = 0;

    vector<vector<int>> buckets(n);

    for (i = 0; i < n; i++)
    {
        bucketIndex = (arr[i] - minVal) / bucketRange;
        buckets[bucketIndex].push_back(arr[i]);
    }

    for (i = 0; i < n; i++)
        sort(buckets[i].begin(), buckets[i].end());

    for (i = 0; i < n; i++)
        for (j = 0; j < buckets[i].size(); j++)
            arr[index++] = buckets[i][j];
}
```

Листинг 2.2 – Реализация алгоритма сортировки слиянием (начало)

```
static void _merge(vector<int> &arr, int low, int high, int mid)
{
    int i, j, k, a;
    int lengthLeft = mid - low + 1;
    int lengthRight = high - mid;

    vector<int> arrLeft(lengthLeft), arrRight(lengthRight);

    for (a = 0; a < lengthLeft; a++)
        arrLeft[a] = arr[low + a];

    for (a = 0; a < lengthRight; a++)
        arrRight[a] = arr[mid + 1 + a];

    i = 0;
    j = 0;
    k = low;

    while (i < lengthLeft && j < lengthRight)
    {
        if (arrLeft[i] <= arrRight[j])
        {
            arr[k] = arrLeft[i];
            i++;
        }
        else
        {
            arr[k] = arrRight[j];
            j++;
        }
        k++;
    }
```

Листинг 2.3 – Реализация алгоритма сортировки слиянием (конец)

```
    while (i < lengthLeft)
    {
        arr[k] = arrLeft[i];
        k++;
        i++;
    }

    while (j < lengthRight)
    {
        arr[k] = arrRight[j];
        k++;
        j++;
    }
}

static void _mergeSort(vector<int> &arr, int low, int high)
{
    if (low < high)
    {
        _mergeSort(arr, low, (low + high) / 2);
        _mergeSort(arr, (low + high) / 2 + 1, high);

        _merge(arr, low, high, (low + high) / 2);
    }
}

void mergeSort(vector<int> &arr)
{
    _mergeSort(arr, 0, arr.size() - 1);
}
```

Листинг 2.4 – Реализация алгоритма поразрядной сортировки (конец)

```
static void countSort(vector<int> &arr, int exp)
{
    int n = arr.size();
    int i;

    vector<int> output(n), count(10, 0);

    for (i = 0; i < n; i++)
        count[(arr[i] / exp) % 10]++;

    for (i = 1; i < 10; i++)
        count[i] += count[i - 1];

    for (i = n - 1; i >= 0; i--)
    {
        output[count[(arr[i] / exp) % 10] - 1] = arr[i];
        count[(arr[i] / exp) % 10]--;
    }

    for (i = 0; i < n; i++)
        arr[i] = output[i];
}

void radixSort(vector<int> &arr)
{
    int max = *max_element(arr.begin(), arr.end());
    int exp;

    for (exp = 1; max / exp > 0; exp *= 10)
        countSort(arr, exp);
}
```

2.5 Функциональные тесты

В таблице 2.1 приведены тестовые данные, на которых было протестировано разработанное ПО. Все тесты были успешно пройдены.

Таблица 2.1 – Функциональные тесты

Массив	**Блочная**	**Слиянием**	**Поразрядная**
1 2 3 4 5 6	1 2 3 4 5 6	1 2 3 4 5 6	1 2 3 4 5 6
6 5 4 3 2 1	1 2 3 4 5 6	1 2 3 4 5 6	1 2 3 4 5 6
41 56 67 10 34 2	2 10 34 41 56 67	2 10 34 41 56 67	2 10 34 41 56 67
54 33 0 55 33 7 14	0 7 14 33 33 54 55	0 7 14 33 33 54 55	0 7 14 33 33 54 55
4 4 4 4 4 4	4 4 4 4 4 4	4 4 4 4 4 4	4 4 4 4 4 4
10	10	10	10
{}	Сообщение об ошибке	Сообщение об ошибке	Сообщение об ошибке

Вывод

В данном разделе были реализованы и протестированы 3 алгоритма сортировки: алгоритм блочной сортировки, алгоритм сортировки слиянием и алгоритм поразрядной сортировки.

3 Исследовательский раздел

В данном разделе будут проведены сравнения реализаций алгоритмов умножения матриц по времени работы и по затрачиваемой памяти.

3.1 Технические характеристики

Технические характеристики устройства, на котором проводились исследования:

– операционная система: Ubuntu 22.04.3 LTS x86_64 [6];

– оперативная память: 16 Гб;

– процессор: 11th Gen Intel® Core™ i7-1185G7 @ 3.00 ГГц × 8.

3.2 Время выполнения алгоритмов

Время работы алгоритмов измерялось с использованием функции *clock_gettime* из встроенного модуля *ctime*.

Замеры времени для каждого размера матрицы проводились 1000 раз. На вход подавались случайно сгенерированные матрицы заданного размера.

3.3 Использование памяти

Вывод

В данном разделе были проведены замеры времени работы, а также расчеты используемой памяти реализаций алгоритмов сортировки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Блочная сортировка [Электронный ресурс]. — Режим доступа: `https://en.wikipedia.org/wiki/Bucket_sort` (дата обращения: 21.11.2023).
2. Сортировка слиянием [Электронный ресурс]. — Режим доступа: `https://en.wikipedia.org/wiki/Merge_sort` (дата обращения: 21.11.2023).
3. Поразрядная сортировка [Электронный ресурс]. — Режим доступа: `https://en.wikipedia.org/wiki/Radix_sort` (дата обращения: 21.11.2023).
4. Справочник по языку C++ [Электронный ресурс]. — Режим доступа: `https://learn.microsoft.com/ru-ru/cpp/cpp/cpp-language-reference?view=msvc-170` (дата обращения: 28.09.2022).
5. clock_getres [Электронный ресурс]. — Режим доступа: `https://pubs.opengroup.org/onlinepubs/9699919799/functions/clock_getres.html` (дата обращения: 28.09.2022).
6. Ubuntu 22.04.3 LTS (Jammy Jellyfish) [Электронный ресурс]. — Режим доступа: `https://releases.ubuntu.com/22.04/` (дата обращения: 28.09.2022).