МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА - Российский технологический университет"
РТУ МИРЭА

Отчет по выполнению практического задания №5

Тема: Алгоритмы внешних сортировок

Дисциплина: «Структуры и алгоритмы обработки данных**»**

Выполнил студент Антонов А.Д.

Группа ИКБО-01-20

Москва 2021

Содержание

1. Сортировка массива методом естественного слияния 3

2. Сортировка массива методом многофазного слияния 3

3. Разработка алгоритма сортировки прямого слияния..... 4

3.1. Постановка задачи 4

3.2. Описание алгоритма прямого слияния 4

3.3. Алгоритм прямого слияния.....5

3.4. Код алгоритма и основной программы.....9

3.5. Результат тестирования 13

4. Разработка алгоритма сортировки естественного слияния 14

4.1. Постановка задачи 14

4.2. Описание алгоритма естественного слияния 14

4.3. Алгоритм естественного слияния.....15

4.4. Код алгоритма и основной программы.....16

4.5. Результат тестирования 22

5. Определение эффективного алгоритма..... 22

ВЫВОДЫ 23

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 23

1. Сортировка массива методом естественного слияния

Исходный файл f: **2 3 17 7 8 9 1 4 6 9 2 3 1 18**

	Распределение	*Слияние*
1 проход	*f1*: 2 3 17 ` 1 4 6 9 ` 1 18 *f2*: 7 8 9 ` 2 3	*f*: 2 3 7 8 9 17 1 2 3 4 6 9 1 18
2 проход	*f1*: 2 3 7 8 9 17 ` 1 18 *f2*: 1 2 3 4 6 9 `	*f*: 1 2 2 3 3 4 6 7 8 9 9 17 1 18
3 проход	*f1*: 1 2 2 3 3 4 6 7 8 9 9 17 *f2*: 1 18	*f*: 1 1 2 2 3 3 4 6 7 8 9 9 17 18

Рис.1 Пример естественного слияния

После этого сортировка будет завершена, т.к. в массив была записана серия размером с изначальным размер массива.

2. Сортировка массива методом многофазного слияния

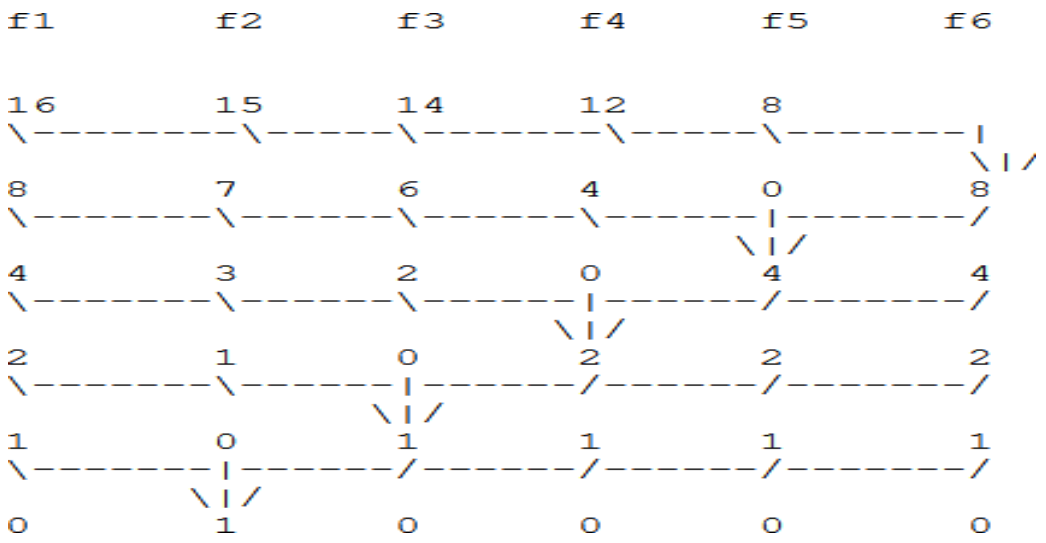

Рис.2 Пример многофазного слияния

3. Разработка алгоритма сортировки прямого слияния

3.1. Постановка задачи

Разработать алгоритм сортировки прямым слиянием, провести анализ практической сложности алгоритма на файлах, заполненных различным количеством записей.

3.2. Описание алгоритма прямого слияния

Для решения задачи сортировки эти три этапа выглядят так:

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.

1.1. — 2.1. Рекурсивное разбиение задачи на меньшие происходит до тех пор, пока размер массива не достигнет единицы (любой массив длины 1 можно считать упорядоченным).

3.1. Соединение двух упорядоченных массивов в один.

Основную идею слияния двух отсортированных массивов можно объяснить на следующем примере. Пусть мы имеем два уже отсортированных по возрастанию подмассива. Тогда:

3.2. Слияние двух подмассивов в третий результирующий массив.

На каждом шаге мы берём меньший из двух первых элементов подмассивов и записываем его в результирующий массив. Счётчики номеров элементов результирующего массива и подмассива, из которого был взят элемент, увеличиваем на 1.

3.4. «Прицепление» остатка.

Когда один из подмассивов закончился, мы добавляем все оставшиеся элементы второго подмассива в результирующий массив.

3.3. Алгоритм прямого слияния

- Массив делится пополам — на левую и правую половины.
- Элементы разбиваются на группы.
- На первой итерации это двойки элементов (1-й элемент левой половины + 1-й элемент правой половины, 2-й элемент левой половины + 2-й элемент правой половины и т.д.), на второй итерации — четвёрки элементов (1-й и 2-й элементы левой половины + 1-й и 2-й элементы правой половины, 3-й и 4-й элементы левой половины + 3-й и 4-й элементы правой половины и т.д.), на третьей — восьмёрки и т.д.
- Элементы каждой группы из левой половины являются отсортированным подмассивом, элементы каждой группы из правой половины также являются отсортированным подмассивом.
- Производим слияние отсортированных подмассивов из предыдущего пункта.
- Возвращаемся в пункт 1. Цикл продолжается до тех пор, пока размеры групп меньше размера массива

3.4. Код алгоритма и основной программы

Алгоритм сортировки прямым слиянием представлен следующими функциями:

Функция разделения массива на части:

```
7   void merge(int j, int r, int m, int n, int *a, int &comps, int &swaps) {
8       if (j + r < n) {
9           if (m == 1) {
10              comps++;
11              if (a[j] > a[j + r]) {
12                  swaps++;
13                  swap( &: a[j],  &: a[j + r]);
14              }
15          } else {
16              m = m / 2;
17              merge(j, r, m, n, a,  &: comps,  &: swaps);
18              if (j + r + m < n) {
19                  merge( j: j + m, r, m, n, a,  &: comps,  &: swaps);
20              }
21              merge( j: j + m,  r: r - m, m, n, a,  &: comps,  &: swaps);
22          }
23      }
24  }
```

Функция сортировки массива

```
26  void merge_sort(int *a, int n) {
27      int comps = 0, swaps = 0;
28      int m = 1;
29      while (m < n) {
30          int j = 0;
31          while (j + m < n) {
32              merge(j, m, m, n, a,  &: comps,  &: swaps);
33              j = j + m + m;
34          }
35          m = m + m;
36      }
37  }
38
```

```
39  void print_arr(vector<vector<string>> array) {
40      for (auto & i : array) {
41          for (int j = 0; j < 9; j++) {
42              cout << i[j];
43          }
44          cout << endl;
45      }
46  }
```

```
48  void input_a(vector<vector<string>> array) {
49      ofstream a( s: "test_samples/a.txt");
50      for (int i = 0; i < array.size(); ++i) {
51          for (int j = 0; j < 9; j++) {
52              a << array.at(i)[j];
53          }
54          if (i != array.size() - 1) a << "\n";
55      }
56      a.close();
57  }
58
59  void input_b(vector<vector<string>> array) {
60      ofstream b( s: "test_samples/b.txt");
61      for (int i = 0; i < array.size(); ++i) {
62          for (int j = 0; j < 9; j++) {
63              b << array.at(i)[j];
64          }
65          if (i != array.size() - 1) b << "\n";
66      }
67      b.close();
68  }
69
70  void input_c(vector<vector<string>> array) {
71      ofstream c( s: "test_samples/c.txt");
72      for (int i = 0; i < array.size(); ++i) {
73          for (int j = 0; j < 9; j++) {
74              c << array.at(i)[j];
75          }
76          if (i != array.size() - 1) c << "\n";
77      }
78      c.close();
79  }
```

```
vector<vector<string>> output_a(int n) {
    vector<vector<string>> arr;
    ifstream file( s: "test_samples/a.txt");
    string buff;
    for (int i = 0; i < n; ++i) {
        vector<string> str;
        for (int j = 0; j < 5; j++) {
            file >> buff;
            str.push_back(buff);
            if (j != 4) str.push_back(" ");
        }
        arr.push_back(str);
    }
    file.close();
    return arr;
}

vector<vector<string>> output_b(int n) {
    vector<vector<string>> arr;
    ifstream file( s: "test_samples/b.txt");
    string buff;
    for (int i = 0; i < n; ++i) {
        vector<string> str;
        for (int j = 0; j < 5; j++) {
            file >> buff;
            str.push_back(buff);
            if (j != 4) str.push_back(" ");
        }
        arr.push_back(str);
    }
    file.close();
    return arr;
}
```

```
vector<vector<string>> output_c(int n) {
    vector<vector<string>> arr;
    ifstream file( s: "test_samples/c.txt");
    string buff;
    for (int i = 0; i < n; ++i) {
        vector<string> str;
        for (int j = 0; j < 5; j++) {
            file >> buff;
            str.push_back(buff);
            if (j != 4) str.push_back(" ");
        }
        arr.push_back(str);
    }
    file.close();
    return arr;
}

vector<vector<string>> output_students(int n) {
    vector<vector<string>> arr;
    ifstream file( s: "test_samples/out.txt");
    string buff;
    for (int i = 0; i < n; ++i) {
        vector<string> str;
        for (int j = 0; j < 5; j++) {
            file >> buff;
            str.push_back(buff);
            if (j != 4) str.push_back(" ");
        }
        arr.push_back(str);
    }
    file.close();
    return arr;
}
```

3.5. Результат тестирования

Для тестирования алгоритма были разработаны тесты согласно варианту личного задания (Вариант №1). Сортировка производилась над записями учета данных студентов. Карточка пользователя (строка) содержала следующие параметры: номер зачетной книжки, фамилия, имя, дата рождения, номер телефона, дата поступления. Сортировка осуществлялась относительно номера зачетной книжки. На рис. 3 представлен фрагмент подготовленных записей в файле out.txt для тестирования. В таблице 1 результаты тестирования сортировки прямого слияния на разном количестве записей.

```
123456 Пупкин Василий 01.01.1970 88005553535 26.08.2020
666666 Швабовский Артемий 02.06.2002 84991234567 26.08.2020
432987 Евтюх Аргений 13.09.2001 89803759472 25.08.2019
327478 Сэмпл Текст 22.11.2001 12345678900 11.11.2011
423795 Куууу Физик 30.04.2000 75093457905 12.05.2020
753954 Товарищ Майор 228.282.255.0 011111111 08.08.2008
234809 Ульрих Ларс 6A.P4.64НЫ 00000000001 04/04.2004
324798 Какая Разница Все Равно Строки
```

Рис. 3 Фрагмент подготовленных записей для тестирования

Таблица 1. Сводная таблица результатов тестирования прямого слияния

n	T(n)	T_T=f(C+M)	$T_п$=$C_ф$+$M_ф$
8	Records: 8 Time: 62 ms	$O(n \cdot \log n)$	Comps: 18 Moves: 54 Total: 72
64	Records: 64 Time: 312 ms		Comps: 292 Moves: 774 Total: 1066
512	Records: 512 Time: 1537 ms		Comps: 3860 Moves: 9225 Total: 13085
4096	Records: 4096 Time: 13607 ms		Comps: 43072 Moves: 98316 Total: 141388
32768	Records: 32768 Time: 101438 ms		Comps: 442132 Moves: 983404 Total: 1425536

4. Разработка алгоритма сортировки естественного слияния

4.1. Постановка задачи

Разработать алгоритм сортировки слиянием, провести анализ практической сложности алгоритма на файлах, заполненных различным количеством записей.

4.2. Описание алгоритма естественного слияния

Алгоритм слияния можно использовать и для сортировки массивов, если последовательно применить его несколько раз ко все более длинным упорядоченным последовательностям. Для этого:

1. В исходном наборе выделяются две подряд идущие возрастающие подпоследовательности (серии)

2. Эти подпоследовательности (серии) сливаются в одну более длинную упорядоченную последовательность так, как описано выше

3. Шаги 1 и 2 повторяются до тех пор, пока не будет достигнут конец входного набора

4. Шаги 1 –3 применяются к новому полученному набору, т.е. выделяются пары серий, которые сливаются в еще более длинные наборы, и.т.д. до тех пор, пока не будет получена единая упорядоченная последовательность.

4.3 Алгоритм естественного слияния

1. Исходный файл f разбивается на два вспомогательных файла f1 и f2. Распределение происходит следующим образом: поочередно считываются записи ai исходной последовательности (неупорядоченной) таким образом, что если значения ключей соседних записей удовлетворяют условию f(ai)<=f(ai+1), то они записываются в первый вспомогательный файл f1. Как только встречаются f(ai)>f(ai+1), то записи ai+1 копируются во второй вспомогательный файл f2. Процедура повторяется до тех пор, пока все записи исходной последовательности не будут распределены по файлам.

2. Вспомогательные файлы f1 и f2 сливаются в файл f, при этом серии образуют упорядоченные последовательности.

3. Полученный файл f вновь обрабатывается, как указано в шагах 1 и 2.

4. Повторяя шаги, сливаем упорядоченные серии до тех пор, пока не будет упорядочен целиком весь файл (Рис.2). Символ "`" обозначает признак конца серии.

4.3. Код алгоритма и основной программы

Алгоритм сортировки естественным слиянием представлен следующими функциями:

```
8   void print_arr(vector <vector<string>> crr) {
9       for (auto & i : crr) {
10          for (int j = 0; j < 9; j++) {
11              cout << i[j];
12          }
13          cout << endl;
14      }
15  }
```

```
17  void input_a(vector <vector<string>> arr) {
18      ofstream a( s: "test_samples/a.txt");
19
20      for (int i = 0; i < arr.size(); ++i) {
21          for (int j = 0; j < 9; j++) {
22              a << arr.at(i)[j];
23          }
24          if (i != arr.size() - 1) a << "\n";
25      }
26      a.close();
27  }
28
29  void input_b(vector <vector<string>> brr) {
30      ofstream b( s: "test_samples/b.txt");
31      for (int i = 0; i < brr.size(); ++i) {
32          for (int j = 0; j < 9; j++) {
33              b << brr.at(i)[j];
34          }
35          if (i != brr.size() - 1) b << "\n";
36      }
37      b.close();
38  }
39
40  void input_c(vector <vector<string>> crr) {
41      ofstream c( s: "test_samples/c.txt");
42      for (int i = 0; i < crr.size(); ++i) {
43          for (int j = 0; j < 9; j++) {
44              c << crr.at(i)[j];
45          }
46          if (i != crr.size() - 1) c << "\n";
47      }
48      c.close();
49  }
```

```
vector <vector<string>> output_a(int n) {
    vector <vector<string>> arr;
    ifstream file( s: "test_samples/a.txt");
    string buff;
    for (int i = 0; i < n; ++i) {
        vector <string> str;
        for (int j = 0; j < 5; j++) {
            file >> buff;
            str.push_back(buff);
            if (j != 4) str.push_back(" ");
        }
        arr.push_back(str);
    }
    file.close();
    return arr;
}

vector <vector<string>> output_b(int n) {
    vector <vector<string>> arr;
    ifstream file( s: "test_samples/b.txt");
    string buff;
    for (int i = 0; i < n; ++i) {
        vector <string> str;
        for (int j = 0; j < 5; j++) {
            file >> buff;
            str.push_back(buff);
            if (j != 4) str.push_back(" ");
        }
        arr.push_back(str);
    }
    file.close();
    return arr;
}
```

```
vector <vector<string>> output_c(int n) {
    vector <vector<string>> arr;
    ifstream file( s: "test_samples/c.txt");
    string buff;
    for (int i = 0; i < n; ++i) {
        vector <string> str;
        for (int j = 0; j < 5; j++) {
            file >> buff;
            str.push_back(buff);
            if (j != 4) str.push_back(" ");
        }
        arr.push_back(str);

    }
    file.close();
    return arr;
}

vector <vector<string>> output_students(int n) {
    vector <vector<string>> arr;
    ifstream file( s: "test_samples/out.txt");
    string buff;
    for (int i = 0; i < n; ++i) {
        vector <string> str;
        for (int j = 0; j < 5; j++) {
            file >> buff;
            str.push_back(buff);
            if (j != 4) str.push_back(" ");
        }
        arr.push_back(str);
    }
    file.close();
    return arr;
}
```

```cpp
void merge_files(int n1, int n2, int k) {
    vector <vector<string>> brr = output_b(n1);
    vector <vector<string>> crr = output_c(n2);
    vector <vector<string>> arr;
    int i = 0, j = 0;
    int q = k;
    while (i < n1 && q <= max(n1, n2)) {
        while ((i < q || j < q) && ((i + j) < (n1 + n2))) {
            cout << q << endl;
            cout << i << " " << j << endl;
            if (i < min(n1, n2) && j < min(n1, n2)) {
                if (brr.at(i)[0] <= crr.at(j)[0]) {
                    arr.push_back(brr.at(i));
                    i++;
                } else {
                    arr.push_back(crr.at(j));
                    j++;
                }
                if (i == q && j < q) {
                    for (j; j < q; j++) arr.push_back(crr.at(j));
                } else if (j == q && i < q) {
                    for (i; i < q; i++) arr.push_back(brr.at(i));
                } else if (i >= min(n1, n2)) {
                    for (i; i < q; i++) arr.push_back(brr.at(i));
                } else if (j >= min(n1, n2)) {
                    for (i; i < q; i++) arr.push_back(brr.at(i));
                }
            }
        }
        q += k;
    }
    input_a( arr: arr);
}
```

```
155  void divide_file(vector <vector<string>> arr, int n, int k) {
156      int u = 0;
157      vector <vector<string>> b, c;
158      for (int i = 0; i < n; ++i) {
159          if (u < k) {
160              u++;
161              b.push_back(arr.at(i));
162          } else {
163              u++;
164              if (u == 2 * k) u = 0;
165              c.push_back(arr.at(i));
166          }
167      }
168      input_b( brr: b);
169      input_c( crr: c);
170  }
171
172
173  void merge_sort(int n) {
174      vector <vector<string>> arr;
175      arr = output_students(n);
176      int k = 1;
177      while (k < n) {
178          divide_file( arr: arr, n, k);
179          int u = 0;
180          int q1 = 0, q2 = 0;
181          for (int i = 0; i < n; i++) {
182              if (u < k)
183                  { u++; q1++; }
184              else
185                  { q2++; u++; if (u == 2 * k) u = 0; }
186          }
187          merge_files(q1, q2, k);
188          arr = output_a(n);
189          k *= 2;
190      }
191  }
```

```cpp
int main() {
    unsigned int start_time = clock();
    setlocale( _Category: LC_ALL,  _Locale: "rus");
    int n = 16;
    // cin >> n;
    cout << "Length: " << n << endl;
    merge_sort(n);
    unsigned int end_time = clock();
    unsigned int search_time = end_time - start_time;
    cout << search_time << "ms";
    return 0;
}
```

Подготовленные данные

```
123456 Пупкин Василий 01.01.1970 88005553535 26.08.2020
666666 Швабовский Артемий 02.06.2002 84991234567 26.08.2020
432987 Евтюх Аргений 13.09.2001 89803759472 25.08.2019
327478 Сэмпл Текст 22.11.2001 12345678900 11.11.2011
423795 Кууу Физик 30.04.2000 75093457905 12.05.2020
753954 Товарищ Майор 228.282.255.0 011111111 08.08.2008
234809 Ульрих Ларс 6A.P4.64HЫ 00000000001 04/04.2004
974328 Адыгейский Сыр 99.99.9999 37498372094 32.32.3232
655464 Юсеков Бир T4.PK.08 94759837482 00.00.0000
895435 Клубень Андрей 43.65.2344 54353463446 65.78.1234
324798 Какая Разница Все Равно Строки
530850 Тык Тык Тык Тык Тык
957348 Оо Оо Оо Моя Оборона
345897 Невер Гонна Гив Ю Ап
573485 Есть Билеты На Концерт Мастейна?
489774 Ладно Дальше Можно Рандомом Сгенерить
```

4.4. Результат тестирования

В таблице 2 результаты тестирования сортировки прямого слияния на разном количестве записей.

Таблица 2. Сводная таблица результатов тестирования естественного слияния

n	T(n)	T_T=f(C+M)	$T_п$=$C_ф$+$M_ф$
8	Records: 8 Time: 51 ms	$O(n \cdot \log n)$	Comps: 12 Moves: 22 Total: 34
64	Records: 64 Time: 216 ms		Comps: 490 Moves: 559 Total: 1049
512	Records: 512 Time: 1612 ms		Comps: 31915 Moves: 32431 Total: 64346
4096	Records: 4096 Time: 34987 ms		Comps: 198434 Moves: 412374 Total: 610808
32768	Records: 32768 Time: 425854 ms		Comps: 2056466 Moves: 2013485 Total: 4069951

5. Определение эффективного алгоритма

На основании результатов тестирования в таблице 1 и таблице 2 видно, что наиболее эффективным является алгоритм прямого слияния: на небольшом количестве записей он уступает алгоритму естественного слияния, но в случае с количеством записей больше 1000 – алгоритм прямого слияния эффективней. Это может быть связано с тем, что для работы алгоритма естественного слияния с файлами используется алгоритм сортировки вставками.

ВЫВОДЫ

В ходе данной практической работы разработали алгоритмы сортировки прямого слияния и естественного слияния. Составили тесты для алгоритмов и произвели их тестирование для определения наиболее эффективного алгоритма. Наиболее эффективным оказался алгоритм прямого слияния: на небольшом количестве записей он уступает алгоритму естественного слияния, но в случае с количеством записей больше 1000 – алгоритм прямого слияния эффективней. Это может быть связано с тем, что для работы алгоритма естественного слияния с файлами используется алгоритм сортировки вставками.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Лекционный материал по структурам и алгоритмам обработки данных Гданского Н.И.

2. Динамический массив // Википедия [Электронный ресурс]. https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2. – (дата обращения: 18.04.2021)

3. Естественное слияние последовательностей. // Shpargalum [Электронный ресурс]. - https://shpargalum.ru/shpora-gos-povtas/strukturyi-i-algoritmyi-obrabotki-dannyix/estestvennoe-sliyanie-posledovatelnostej.-vneshnyaya-sortirovka.html /. – (дата обращения: 18.04.2021)