

МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение высшего образования*

***«МИРЭА – Российский технологический университет»***

**РТУ МИРЭА**

Отчет по выполнению практического задания № 4

**Тема:**

«Определение эффективного алгоритма сортировки на основе эмпирического и асимптотического методов анализа»

Дисциплина: «Алгоритмы внешних сортировок»

Выполнил студент: Моисенко М. О.

Фамилия И.О.

Группа: ИКБО-00-22

Номер группы

Москва – 2023

СОДЕРЖАНИЕ

[1 ПОСТАНОВКА ЗАДАЧИ 3](#_Toc132248790)

[1.1 Условие задания 1 3](#_Toc132248791)

[1.2 Условие задания 2 3](#_Toc132248792)

[2 РЕШЕНИЕ ЗАДАНИЯ 1 3](#_Toc132248793)

[2.1 Описание и реализация алгоритма внешней сортировки прямого слияния 3](#_Toc132248794)

[2.2 Демонстрация работы программы 7](#_Toc132248795)

[2.3 Тестирование работы алгоритма 8](#_Toc132248796)

[3 РЕШЕНИЕ ЗАДАНИЯ 2 9](#_Toc132248797)

[3.1 Описание и реализация алгоритма внешней сортировки естественного слияния 9](#_Toc132248798)

[3.2 Тестирование алгоритма внешней сортировки естественного слияния 16](#_Toc132248799)

[4 ВЫВОД 17](#_Toc132248800)

[5 СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 17](#_Toc132248801)

# ПОСТАНОВКА ЗАДАЧИ

Цель:освоить приёмы сортировки данных из файлов.

## Условие задания 1

Разработать программу и применить алгоритм внешней сортировки прямого слияния к сортировке файла данных индивидуального варианта (приложение 1) по значению ключевого поля (ключ в структуре записи варианта – подчеркнутое поле). Мой вариант обладает номером 9: «Список экспортируемых товаров. Об отдельном товаре хранятся данные: Наименование товара, Страна импортирующая товар, Количество(в штуках).»

## Условие задания 2

Разработать программу и применить алгоритм сортировки естественного слияния к сортировке файла с данными варианта (файл уже должен быть подготовлен в задании 1.

# РЕШЕНИЕ ЗАДАНИЯ 1

## Описание и реализация алгоритма внешней сортировки прямого слияния

Алгоритм состоит из двух этапов: разделение и слияние.

Согласно размеру порции из исходного файла выбирается последовательность элементов и записывается в созданный файл. Когда количество записанных в файл элементов становится равным числу порции, запись происходит в другой файл по тому же критерию остановки записи. Эти действия повторяются до момента, когда все элементы исходного файла не будут переписаны в два других.

Слияние двух файлов в один происходит по следующему правилу: считываются по одной порции из каждого из файлов и, посредством сравнения элементов порций обоих файлов в файл записывается отсортированная комбинация двух порций. Продолжается до тех пор, пока не будет достигнут конец каждого из файлов.

Сами процессы разделения и слияния продолжаются до тех пор, пока размер порции не будет равен количеству записей.

Требование для сортировки: количество сортируемых элементов должно быть степенью двойки, а элементы строго соответствовать шаблону, указанному в индивидуальном задании.

Так как сортировка реализуется не на числовом массиве, а на файле с базой данных, то необходимо реализовать функцию, которая будет извлекать из строки базы необходимый ключ сортировки. Реализация этой функции показана в листинге 1.

Листинг 1 – реализация функции нахождения и выдачи ключевого элемента

string Translate(string l)
{
 string country;
 int s = 0;
 for (int i=0; i < l.length(); i++)
 {
 if (l[i] == '|')
 s = s + 1;
 else if (s == 1)
 country = country + l[i];
 else if (s == 2)
 return country;
 }
}

Функция разделения и функция слияния представлены в листинге 2. Вся работа выполнена на языке C++

Листинг 2 – реализация функций разделения и слияния внешней сортировки прямого слияния

void Separate(int size)
{
 ifstream base("C:/Users/User/CLionProjects/prac\_4/cmake-build-debug/Base.txt");
 ofstream fA("A.txt");
 ofstream fB("B.txt");

 int count = 0;
 string l;
 int i;
 bool perem = true;
 bool endlA = true;
 bool endlB = true;
 if (base.is\_open() && fA.is\_open() && fB.is\_open())
 {
 while (getline(base, l))
 {
 fA << l << endl;
 for (i = 0; i < size - 1 && getline(base, l); i++)
 {
 fA << l << endl;
 }
 for (i = 0; i < size && getline(base, l); i++)
 {
 fB << l << endl;
 }
 }
 base.close();
 fA.close();
 fB.close();
 }

}
int Sliv(int size)
{
 ofstream base("C:/Users/User/CLionProjects/prac\_4/cmake-build-debug/Base.txt");
 ifstream fA("A.txt");
 ifstream fB("B.txt");

 bool first = true;
 int i, j;
 int sizeA = 0;
 int sizeB = 0;
 int port = 0;
 string lA;
 string lB;
 string a;
 string b;

 if (base.is\_open() && fA.is\_open() && fB.is\_open())
 {
 getline(fA, lA);
 getline(fB, lB);
 a = Translate(lA);
 b = Translate(lB);
 while (fA && fB)
 {
 if (sizeA < size && sizeB < size)
 {
 if (a <= b)
 {
 base << lA << endl;
 sizeA++;
 port++;
 getline(fA, lA);
 a = Translate(lA);
 }
 else
 {
 base << lB << endl;
 sizeB++;
 getline(fB, lB);
 b = Translate(lB);
 }
 }
 else if (sizeB < size)
 {
 base << lB << endl;
 sizeB++;
 getline(fB, lB);
 b = Translate(lB);
 }
 else if (sizeA < size)
 {
 base << lA << endl;
 sizeA++;
 port++;
 getline(fA, lA);
 a = Translate(lA);
 }
 else
 {
 sizeA = 0;
 sizeB = 0;
 }
 }
 while (fA)
 {
 base << lA << endl;
 sizeA++;
 port++;
 getline(fA, lA);
 a = Translate(lA);
 }
 while (fB) {
 base << lB << endl;
 sizeB++;
 getline(fB, lB);
 b = Translate(lB);
 }
 base.close();
 fA.close();
 fB.close();
 }
 return port;
}

 Функция, объединявшая в себе все написанные раннее функции показана в листинге 3. Именно ее время работы измерялось в моей работе далее.

Листинг 3 – функция внешней сортировки прямого слияния

void directMergeSort()
{
 Separate(1);
 for (int i = 1; i < Sliv(i); i \*= 2)

 {
 Separate(i \* 2);
 }
}

## Демонстрация работы программы

В качестве примера работы программы я возьму базу данных, состоящую из 16 записей. На рисунке 3 изображена база данных до работы алгоритма и на рисунке 4 представлена версия после выполнения программы. Так как неключевые значения не играют роли в работе сортировки, я их оставил одинаковыми для создания акцента на ключевых элементах.



Рисунок 1 – База данных до работы алгоритма



Рисунок 1 – База данных после работы алгоритма

## Тестирование работы алгоритма

Результаты тестирования программы при различных размерах входных данных представлены в таблице 1.

Таблица 1 - Результаты тестирования внешней сортировки прямым слиянием

|  |  |
| --- | --- |
| **Кол-во строк в исходном файле** | **Время работы программы, мс** |
| 8 | 21 |
| 16 | 43 |
| 32 | 45 |
| 64 | 62 |
| 128 | 86 |

Согласно результатам, вычислительная сложность алгоритма O(n \* log2n).

# РЕШЕНИЕ ЗАДАНИЯ 2

## Описание и реализация алгоритма внешней сортировки естественного слияния

Первым делом исходный файл разбивается на два фиксированной длины, которые сортируются методом внутренней сортировки (в данном случае используется сортировка простым выбором). Затем данные копируются обратно в исходный файл.

Алгоритм сортировки естественным слиянием разбивает исходный файл на два по сериям. В каждой серии значения уже отсортированы, при этом размер серии не фиксирован. Конец серии определяется выполнениям условия: i-ый элемент меньше i+1-го. С каждой строки данные считываются в две переменные: в одну будет считываться ключевое поле, а во вторую – вся строка. Процесс разделения и слияния продолжается до тех пор, пока размер серии не будет равен количеству записей.

Алгоритм реализован на языке С++. Функция сортировки выбором продемонстрирована в листинге 4. Функции разделения фиксированного размера и разделения отсортированного исходного файла представлены в листинге 5, функция слияния представлена в листинге 6. И функция, выполняющая внешнюю сортировку прямого слияния показана в листинге 7.

Листинг 4 – реализация функций разделения и слияния внешней сортировки прямого слияния

void SelectionSort(string \*mas, string \*massStr, int n)
{
 int min;
 string str, s;

 for (int i = 0; i < n - 1; i++)
 {
 min = i;
 for (int j = i + 1; j < n; j++)
 {
 if (mas[j] < mas[min])
 min = j;
 }
 if (min != i)
 {
 s = mas[i];
 mas[i] = mas[min];
 mas[min] = s;

 str = massStr[i];
 massStr[i] = massStr[min];
 massStr[min] = str;
 }
 }
}

Листинг 5 – реализация функций разделения и слияния внешней сортировки прямого слияния

void seriesFixedDivideFiles(int size)
{
 // открытие файлов
 ifstream base("C:/Users/User/CLionProjects/prac\_4/cmake-build-debug/Base.txt");
 ofstream fA("A.txt");
 ofstream fB("B.txt");

 string \*massId = new string[size];
 string \*massStr = new string[size];
 int iMass = 0;

 int i;
 string l, obj;
 bool nextB = true;

 if (base.is\_open() && fA.is\_open() && fB.is\_open())
 {

 getline(base, l);
 obj = Translate(l);

 while (base)
 {

 massId[iMass] = obj;
 massStr[iMass] = l;
 iMass++;

 if (iMass == size)
 {

 SelectionSort(massId, massStr, iMass);

 if (nextB)
 {
 for (i = 0; i < iMass; i++)
 {
 fA << massStr[i] << endl;
 }
 nextB = false;
 } else
 {
 for (i = 0; i < iMass; i++)
 {
 fB << massStr[i] << endl;
 }
 nextB = true;
 }
 iMass = 0;

 }

 getline(base, l);
 obj = Translate(l);
 }

 if (iMass > 0)
 {
 SelectionSort(massId, massStr, iMass);
 if (nextB)
 {
 for (i = 0; i < iMass; i++)
 {
 fA << massStr[i] << endl;
 }
 nextB = false;
 } else
 {
 for (i = 0; i < iMass; i++) {
 fB << massStr[i] << endl;
 }
 nextB = true;
 }

 }

 base.close();
 fA.close();
 fB.close();
 }

 delete[] massId;
 delete[] massStr;
}
void seriesDivideFiles() {
 ifstream base("C:/Users/User/CLionProjects/prac\_4/cmake-build-debug/Base.txt");
 ofstream fA("A.txt");
 ofstream fB("B.txt");

 string l, lnext;
 string id, idNext;
 bool fileB = true;

 if (base.is\_open() && fA.is\_open() && fB.is\_open())
 {

 getline(base, l);
 id = Translate(l);
 getline(base, lnext);
 idNext = Translate(lnext);

 while (base)
 {
 if (fileB)
 fA << l << endl;
 else
 fB << l << endl;

 if (id > idNext)
 {
 fileB = !(fileB);
 }

 id = idNext;
 l = lnext;

 getline(base, lnext);
 idNext = Translate(lnext);
 }

 if (fileB)
 fA << l << endl;
 else
 fB << l << endl;

 base.close();
 fA.close();
 fB.close();
 }
}

Листинг 6 – реализация функций разделения и слияния внешней сортировки прямого слияния

bool naturalMerge()
{

 ofstream base;
 base.open("C:/Users/User/CLionProjects/prac\_4/cmake-build-debug/Base.txt");

 ifstream fA;
 fA.open("A.txt");
 bool lastA = false;
 bool lastInFileA = false;

 ifstream fB;
 fB.open("B.txt");
 bool lastB = false;
 bool lastInFileB = false;

 string lA, lB, lANext, lBNext;
 string idA, idB, idAN, idBN;

 if (base.is\_open() && fA.is\_open() && fB.is\_open()) {

 getline(fA, lA);
 idA = Translate(lA);
 getline(fA, lANext);
 idAN = Translate(lANext);
 getline(fB, lB);
 idB = Translate(lB);
 getline(fB, lBNext);
 idBN = Translate(lBNext);

 if (!fA)
 {
 base << lB << endl;
 base << lBNext << endl;
 while (getline(fB, lB))
 {
 base << lB << endl;
 }
 base.close();
 fA.close();
 fB.close();
 return true;
 }
 if (!fB)
 {
 base << lA << endl;
 base << lANext << endl;
 while (getline(fA, lA))
 {
 base << lA << endl;
 }
 base.close();
 fA.close();
 fB.close();
 return true;
 }

 while (true)
 {
 if (!lastB && !lastA)
 {
 if (idA <= idB)
 {
 if (idA > idAN)
 {
 lastA = true;
 }
 base << lA << endl;

 idA = idAN;
 lA = lANext;
 getline(fA, lANext);
 idAN = Translate(lANext);

 if (lastInFileA)
 {
 base << lB << endl;
 if (!lastInFileB)
 base << lBNext << endl;
 break;
 }

 } else
 {
 if (idB > idBN)
 {
 lastB = true;
 }
 base << lB << endl;

 idB = idBN;
 lB = lBNext;
 getline(fB, lBNext);
 idBN = Translate(lBNext);

 if (lastInFileB)
 {
 base << lA << endl;
 if (!lastInFileA)
 base << lANext << endl;
 break;
 }
 }
 } else if (!lastB)
 {
 if (idB > idBN)
 {
 lastB = true;
 }
 base << idB << lB << endl;
 idB = idBN;
 lB = lBNext;
 fB >> idBN;
 getline(fB, lBNext);
 if (lastInFileB)
 {
 base << idA << lA << endl;
 if (!lastInFileA)
 base << idAN << lANext << endl;
 break;
 }
 } else if (!lastA)
 {
 if (idA > idAN)
 {
 lastA = true;
 }
 base << idA << lA << endl;
 idA = idAN;
 lA = lANext;
 fA >> idAN;
 getline(fA, lANext);
 if (lastInFileA)
 {
 base << lB << endl;
 if (!lastInFileB)
 base << lBNext << endl;
 break;
 }
 } else
 {
 lastA = false;
 lastB = false;
 }

 if (!fA)
 lastInFileA = true;
 if (!fB)
 lastInFileB = true;
 }

 while (getline(fA, lA))
 base << lA << endl;
 while (getline(fB, lB))
 base << lB << endl;

 base.close();
 fA.close();
 fB.close();
 }
 return false;
}

Листинг 7 – реализация функций разделения и слияния внешней сортировки прямого слияния

void naturalMergeSort(int n)
{

 seriesFixedDivideFiles(n);

 ofstream base;
 base.open("C:/Users/User/CLionProjects/prac\_4/cmake-build-debug/Base.txt");

 ifstream fA;
 fA.open("A.txt");

 ifstream fB;
 fB.open("B.txt");

 string line;

 if (base.is\_open() && fA.is\_open() && fB.is\_open())
 {
 while (getline(fA, line))
 {
 base << line << endl;
 }
 while (getline(fB, line))
 {
 base << line << endl;
 }

 base.close();
 fA.close();
 fB.close();
 }

 do
 {
 seriesDivideFiles();
 } while (!naturalMerge());
}

## Тестирование алгоритма внешней сортировки естественного слияния

Результаты тестирования программы при различных размерах входных данных представлены в таблице 2.

Таблица 2 - Результаты тестирования внешней сортировки естественным слиянием

|  |  |
| --- | --- |
| **Кол-во строк в исходном файле** | **Время работы программы, мс** |
| 8 | 10 |
| 16 | 17 |
| 32 | 28 |
| 64 | 32 |
| 128 | 55 |

Согласно результатам, вычислительная сложность алгоритма O(n \* log2n).

# ВЫВОД

Освоены алгоритмы внешних. Реализованы сортировки прямым и естественным слиянием для файла, структура которого выстроена согласно индивидуальному варианту. Согласно тестированиям, алгоритмы сортировок работают верно. Проанализированы времена работы сортировок при разном количестве строк в исходном файле. Согласно результатам, сортировка естественным слиянием в среднем случае работает немного быстрее сортировки прямым слиянием. Функции роста обоих алгоритмов равны.

# СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Структуры данных и проектирование программ : Пер. с англ. / Р. Круз. — М.: БИНОМ. Лаборатория знаний, 2017. — 766 с.

2. Полный справочник по C++ : Пер. с англ. / Г. Шилдт. — М.: ООО "И.Д.Вильямс", 2016. — 796 с.: ил. — Предм. указ.: с. 787-796