Институт приборостроения, автоматизации и информационных технологий

Кафедра информационных систем и цифровых технологий

Дисциплина «Алгоритмы и структуры данных»

Отчет к лабораторной работе № 7

«Сортировка во внешней памяти многопутевым сбалансированным слиянием»

Выполнил:

Василения Иван Валерьевич

Мельниченко Артём Олегович

Принял:

Рыженков Д. В.

Орёл, 2024г

|  |  |  |
| --- | --- | --- |
| Количество путей слияния | Количество слияний | Время |
| 2 | 13 | 198,938 |
| 3 | 8 | 99,0688 |
| 4 | 7 | 81,5452 |
| 5 | 6 | 73,962 |
| 6 | 5 | 85,5906 |
| 7 | 5 | 84,8344 |
| 8 | 5 | 72,8226 |
| 9 | 4 | 75,0808 |
| 10 | 4 | 85,7972 |

Листинг файла main.cpp

#include <iostream>

#include <fstream>

#include <random>

#include <ctime>

#include <vector>

#include <string>

#include <chrono>

#include "Record.h"

using namespace std;

void createF0(int amount) {

ofstream outfile("f0.bin", ios::binary);

for (int i = 0; i < amount; i++)

{

Record record;

record.key = rand();

outfile.write((char\*)&record, sizeof(Record));

}

outfile.close();

}

int printFile(const char\* filename) {

int cntr = 0;

cout << "FILE: " << filename << endl;

ifstream infile(filename, ios::binary);

Record record;

while (infile.read((char\*)&record, sizeof(Record)) && !infile.eof()) {

cout << record.key << endl;

cntr++;

}

infile.close();

cout << "--------------------------------" << endl;

return cntr;

}

int bMerge(int mergeWays) {

int mergeCount = 0;

ifstream f0("f0.bin", ios::binary); // Основная последовательность

vector<fstream>outFiles(mergeWays \* 2);

vector<int> indexMap(mergeWays \* 2);

int L = 0; // Кол-во полученных отрезков

for (int i = 0; i < mergeWays; i++) // Открытие N файлов на вывод

{

outFiles[i].open("out" + to\_string(i) + ".bin", ios::binary | ios::out | ios::trunc);

}

int minKey = INT\_MIN;

int fileIndex = 0;

Record record;

while (f0.read((char\*)&record, sizeof(Record)) && !f0.eof()) { // Начальная иниц. выходных файлов

if (record.key < minKey) { // Конец отрезка

fileIndex = (fileIndex + 1) % mergeWays;

L++;

}

minKey = record.key;

outFiles[fileIndex].write((char\*)&record, sizeof(Record));

}

for (int i = 0; i < mergeWays; i++) // Закрытие файлов

{

outFiles[i].close();

}

f0.close(); // Больше не нужен

for (int i = 0; i < mergeWays \* 2; i++) // Начальная иниц. индексной карты

{

indexMap[i] = i;

}

while (L != 1) { // Слияние

mergeCount++;

int af = min(L, mergeWays); // Активные файлы

int ao; //количество активных файлов с активными отрезками;

for (int i = 0; i < af; i++)

{

outFiles[indexMap[i]].open("out" + to\_string(indexMap[i]) + ".bin", ios::binary | ios::in);

// Открыть для чтения t[0]..t[af] файлы

}

for (int i = mergeWays; i < mergeWays \* 2; i++)

{

outFiles[indexMap[i]].open("out" + to\_string(indexMap[i]) + ".bin", ios::binary | ios::out | ios::trunc);

// Открыть для записи t[N+1]..t[2N] файлы

}

vector<int>ta(mergeWays); // Массив индексов активных файлов, в которых имеются активные отрезки

for (int i = 0; i < mergeWays; i++)

{

ta[i] = i; // инициализировать ta[1].. ta[af] индексами из ta[]

}

int j = mergeWays; //j – индекс очередного выходного файла;

L = 0;

while (af > 0) // Проход

{

L++;

ao = mergeWays;

vector<Record>records(mergeWays); // Массив для считанных записей

vector<int>lastKeys(mergeWays); // Массив для значений последнего считанного ключа файла

for (int i = 0; i < mergeWays; i++)

{

records[i].key = INT\_MAX;

lastKeys[i] = INT\_MIN;

}

vector<int> localta(mergeWays);

for (int i = 0; i < mergeWays; i++)

{

localta[i] = ta[i];

}

while (ao != 0) { //слияние первых активных отрезков в f[t[j]]

int m = -1; // Индекс файла с минимальным ключом

int minKey = INT\_MAX;

for (int i = 0; i < mergeWays; i++) // По итогу цикла получим m

{

if (ta[i] >= 0 && localta[i] >= 0 && records[i].key == INT\_MAX) { // Файл i активен, необходимо чтение

outFiles[indexMap[i]].read((char\*)&records[i], sizeof(Record));

if (outFiles[indexMap[i]].eof()) { // Файл кончился

ta[i] = -1; // Файл больше не активен

localta[i] = -1; // Файл больше не активен

af--;

ao--;

records[i].key = INT\_MAX; // Это для корректного нахождения минимума

}

else if (lastKeys[i] > records[i].key) { // Считан ключ меньше чем в

ao--; // прошлый раз => отрезок кончился

localta[i] = -1;

records[i].key = INT\_MAX;

outFiles[indexMap[i]].seekg(-1 \* sizeof(Record), ios::cur); // т.к. мы считали лишнюю запись, нужно считать снова позже

}

lastKeys[i] = records[i].key;

}

if (records[i].key < minKey) { // Между делом находим минимальный ключ и индекс его файла

minKey = records[i].key;

m = i;

}

}

if (m == -1) break; // Все отрезки закончились

outFiles[indexMap[j]].write((char\*)&records[m], sizeof(Record));//скопировать запись из f[ta[m]] в f[t[j]]

records[m].key = INT\_MAX; // Для того, чтобы на следующей итерации

// из файла с индексом m считалась новая запись

}

//outFiles[indexMap[j]].close();

j = ((j + 1) % mergeWays) + mergeWays;

}

// Переключить индексную карту

for (int i = 0; i < mergeWays; i++)

{

swap(indexMap[i], indexMap[i + mergeWays]);

}

for (int i = 0; i < mergeWays \* 2; i++)

{

outFiles[indexMap[i]].close();

}

}

/\*string name = "out" + to\_string(indexMap[0]) + ".bin";

int total = printFile(name.c\_str());

cout << "TOTAL: " << total << endl;\*/

return mergeCount;

}

int main()

{

setlocale(LC\_ALL, "RUS");

std::locale::global(std::locale("ru\_RU.UTF-8"));

std::cout.imbue(std::locale());

srand(static\_cast<unsigned int>(std::time(0)));

//createF0(10000);

std::ofstream file("data.csv");

if (!file.is\_open()) {

std::cerr << "Не удалось создать файл: " << "data.csv" << std::endl;

return -1;

}

file << "Количество путей слияния" << ';';

file << "Количество слияний" << ';';

file << "Время" << endl;

for (int i = 2; i <= 10; i++)

{

auto start = chrono::steady\_clock::now();

int merges = bMerge(i);

auto end = chrono::steady\_clock::now();

chrono::duration<double, std::milli> elapsed\_milliseconds = end - start;

double duration = elapsed\_milliseconds.count();

cout << "Merge ways: " << i << endl;

cout << "Time: " << duration << " миллисекунд." << endl;

cout << "Merges: " << merges << endl;

cout << "========================================" << endl;

file << i << ';';

file << merges << ';';

file << duration << endl;

}

file.close();

return 0;

}