**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 3

по дисциплине **«Алгоритмы и структуры данных»**

на тему: **«**Сортировка**»**

Выполнил: студент гр. ИП-32

Суховенко Э. С.

Принял: преподаватель М. А. Процкая

Гомель 2022

**Цель работы:** Изучить основные методы сортировки. Сравнить эффективность методов.

Разработать проект для обработки дерева поиска, каждый элемент которого содержит структуру данных из сформировать случайным образом массив записей из 100000 штук. Скопировать его трижды. Провести сортировку по одному ключу, двум ключам и трем ключам. Оценить эффективность каждой сортировки по времени и количеству перестановок и сравнений. Провести четвертую сортировку, включив проверку на каждом шаге, что массив уже отсортирован. Оценить эффективность включения такой оценки. Организовать поиск по трем указанным ключам. Оценить эффективность поиска.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 12 | Data, float, str | У, ВВ, УУВ | Пузырёк слева направо | бинарный |

**Алгоритм решения задачи:**

Пузырьковая сортировка:

1. опишем функцию swap, которая принимает 2 индекса и массив и меняет местами элементы находящиеся на этих индексах

2. проходим в цикле все элементы в цикле и сравниваем соседние, если левый элемент больше правого, а сортировка по возрастанию то выполняем функцию swap

3. выполняем алгоритм n раз, где n – длина массива

**Листинг программы:**

using System;

namespace AISDLab3

{

public static class BinarySearch

{

public static void Search(Item[] array, Item item)

{

int low = 0;

int high = array.Length - 1;

while (low <= high)

{

int mid = (low + high) / 2;

if (Item.CompareToThreeKeys(item, array[mid]) == -1)

high = mid - 1;

else if (Item.CompareToThreeKeys(item, array[mid]) == 1)

low = mid + 1;

else if (Item.CompareToThreeKeys(item, array[mid]) == 0)

{

Console.WriteLine("Поиск успешен");

Console.WriteLine("Элемент {0} найден на позиции {1}\n", item, mid + 1);

return;

}

}

Console.WriteLine("Поиск не успешен");

}

}

}

using System;

using System.Text;

using System.Threading;

namespace AISDLab3

{

public static class GenerationData

{

public static Item[] GenerateDate(int size)

{

var workArray = new Item[size];

for (var i = 0; i < size; i++)

{

Thread.Sleep(30);

workArray[i] = new Item

{

Date = RandomDay(),

Float = RandomFloat(),

Str = RandomString(RandomNumber(5, 8), false)

};

}

workArray[2].Date = workArray[3].Date;

workArray[1].Date = workArray[4].Date;

workArray[1].Float += 20;

workArray[6].Date = workArray[7].Date;

workArray[6].Float = workArray[7].Float;

return workArray;

}

private static string RandomString(int size, bool lowerCase)

{

StringBuilder builder = new StringBuilder();

Random random = new Random();

char ch;

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

{

ch = Convert.ToChar(Convert.ToInt32(Math.Floor(26 \* random.NextDouble() + 65)));

builder.Append(ch);

}

if (lowerCase)

return builder.ToString().ToLower();

return builder.ToString();

}

private static float RandomFloat()

{

Random random = new Random();

return (float)random.NextDouble() \* random.Next(2, 100);

}

private static DateTime RandomDay()

{

Random random = new Random();

DateTime start = new DateTime(1995, 1, 1);

int range = (DateTime.Today - start).Days;

return start.AddDays(random.Next(range));

}

private static int RandomNumber(int min, int max)

{

Random random = new Random();

return random.Next(min, max);

}

}

}

using System;

namespace AISDLab3

{

public class Item

{

public DateTime Date { get; set; }

public float Float { get; set; }

public string Str { get; set; }

public override string ToString() => $"Date: {Date}\t Float: {Float}\t Str: {Str}";

public static int InverseCompare(int result) => result == 0 ? 0 : result == 1 ? -1 : 1;

// У

public static int CompareToOneKey(Item left, Item right) => InverseCompare(DateTime.Compare(left.Date, right.Date));

// ВВ

public static int CompareToTwoKeys(Item left, Item right)

{

if (DateTime.Compare(left.Date, right.Date) != 0)

return DateTime.Compare(left.Date, right.Date);

if (left.Float.CompareTo(right.Float) != 0)

return left.Float.CompareTo(right.Float);

return 0;

}

// УУВ

public static int CompareToThreeKeys(Item left, Item right)

{

if (DateTime.Compare(left.Date, right.Date) != 0)

return InverseCompare(DateTime.Compare(left.Date, right.Date));

if (left.Float.CompareTo(right.Float) != 0)

return InverseCompare(left.Float.CompareTo(right.Float));

return left.Str.CompareTo(right.Str);

}

}

}

using System;

using System.Diagnostics;

namespace AISDLab3

{

class Program

{

static void Main()

{

Console.WriteLine("Введите размер массива:");

int size = int.Parse(Console.ReadLine());

Item[] workArray = GenerationData.GenerateDate(size);

Item[] secondArray = workArray;

Item[] thirdArray = workArray;

var stopwatch = new Stopwatch();

Console.WriteLine("Исходный массив: ");

Print(workArray);

Console.WriteLine();

Console.WriteLine("Сортировка по одному ключу: ");

stopwatch.Start();

Item[] firstResult = SortService.SortOneKey(workArray);

stopwatch.Stop();

Print(firstResult);

Console.WriteLine("Время сортировки по одному ключу: " + (double)stopwatch.ElapsedTicks / 1000 + " ms\n");

Console.WriteLine("Сортировка по двум ключам: ");

stopwatch.Restart();

Item[] secondResult = SortService.SortTwoKey(secondArray);

stopwatch.Stop();

Print(secondResult);

Console.WriteLine("Время сортировки по двум ключам: " + (double)stopwatch.ElapsedTicks / 1000 + " ms\n");

Console.WriteLine("Сортировка по трем ключам: ");

stopwatch.Restart();

Item[] thirdResult = SortService.SortThreeKey(thirdArray);

stopwatch.Stop();

Print(thirdResult);

Console.WriteLine("Время сортировки по трем ключам: " + (double)stopwatch.ElapsedTicks / 1000 + " ms\n");

Console.WriteLine("Сортировка с проверкой: ");

stopwatch.Restart();

Item[] sortWithCheck = SortService.SortThreeKeyWithCheck(thirdArray);

stopwatch.Stop();

Print(sortWithCheck);

Console.WriteLine("Время сортировки с проверкой: " + (double)stopwatch.ElapsedTicks / 1000 + " ms\n");

Console.WriteLine("Бинарный поиск: ");

stopwatch.Restart();

BinarySearch.Search(workArray, workArray[2]);

stopwatch.Stop();

Console.WriteLine("Время нахождения элемента: " + (double)stopwatch.ElapsedTicks / 1000 + " ms\n");

}

public static void Print(Item[] items)

{

foreach (var item in items)

{

Console.WriteLine(item);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AISDLab3

{

public class SortService

{

public static Item[] SortOneKey(Item[] workArray)

{

var list = workArray.ToList();

list.Sort(Item.CompareToOneKey);

return list.ToArray();

}

public static Item[] SortTwoKey(Item[] workArray)

{

var list = workArray.ToList();

list.Sort(Item.CompareToTwoKeys);

return list.ToArray();

}

public static Item[] SortThreeKey(Item[] workArray)

{

var list = workArray.ToList();

list.Sort(Item.CompareToThreeKeys);

return list.ToArray();

}

public static Item[] SortThreeKeyWithCheck(Item[] workArray)

{

var copiedArray = (Item[]) workArray.Clone();

for (int i = 0; i < copiedArray.Length; i++)

{

int minInd = i;

for (int j = i + 1; j < copiedArray.Length; j++)

{

if (Item.CompareToThreeKeys(copiedArray[i], copiedArray[j]) == 1)

{

minInd = j;

}

}

if (minInd != i)

{

Item temp = copiedArray[i];

copiedArray[i] = copiedArray[minInd];

copiedArray[minInd] = temp;

}

bool sorted = false;

for (int j = 0; j < copiedArray.Length - 1; j++)

{

if (Item.CompareToThreeKeys(copiedArray[i], copiedArray[i + 1]) <= 0)

{

sorted = true;

break;

}

}

if (sorted)

{

break;

}

}

return copiedArray;

}

}

}

*Асимптотическая оценка сложности алгоритма:*

O(n^2) – вне зависимости от компаратора для пузырьковой сортировки.

*Наилучшую, наихудшую и среднюю оценки:*

Все оценки равны O(n^2), если только не была описана оптимизация на проверку предварительно отсортированного массива за O(n), в таком случае лучшая оценка будет O(n)

*Псевдокод операции на упрощённом языке:*

bubbleSort(arr, comp):

for (let i = 0; i < arr.length; ++i) {

for (let j = 0; j < arr.length - 1; ++j) {

if (comp(arr[j], arr[j + 1]) == 1) {

[arr[j], arr[j + 1]] = [arr[j + 1], arr[j]]

}

}

}

*Выкладки расчета асимптотической оценки:*

Первый цикл проходит O(n) шагов, второй точно также, на каждом шагу первого второй делает O(n) операций. Итого: O(n^2)

*Выкладки или результаты тестирования на различных по объему набору данных:*

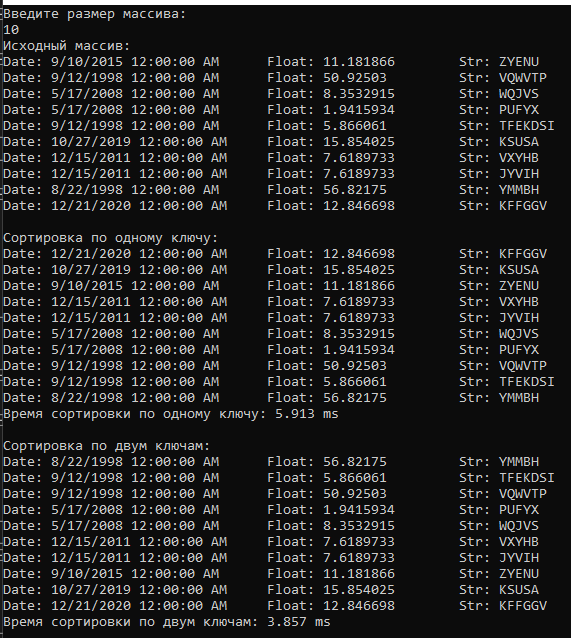
****

Рисунок 1

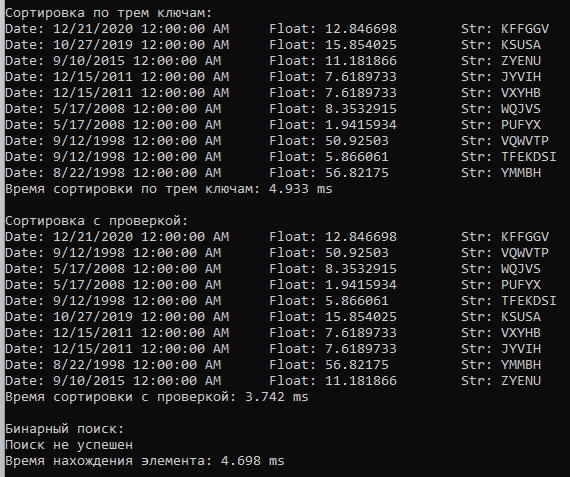


Рисунок 2

**Вывод:** изучил основы работы с нелинейными динамическими структурами (бинарными деревьями).